Fast Synthetic LiDAR Rendering via Spherical UV Unwrapping of Equirectangular Z-Buffer Images

Mohammed Hossnya,∗, Khaled Salehb, Mohammed Attiac, Ahmed Abobakra, Julie Iskandera

aDeakin University, Melbourne, Australia
bUniversity of Technology Sydney, Australia
cMedical Research Institute, Alexandria University, Egypt

Abstract

LiDAR data is becoming increasingly essential with the rise of autonomous vehicles. Its ability to provide 360° horizontal field of view of point cloud, equips self-driving vehicles with enhanced situational awareness capabilities. While synthetic LiDAR data generation pipelines provide a good solution to advance the machine learning research on LiDAR, they do suffer from a major shortcoming, which is rendering time. Physically accurate LiDAR simulators (e.g. BlenSor) are computationally expensive with an average rendering time of 14-60 seconds per frame for urban scenes. This is often compensated for via using 3D models with simplified polygon topology (low poly assets) as is the case of CARLA (Dosovitskiy et al., 2017). However, this comes at the price of having coarse grained unrealistic LiDAR point clouds. In this paper, we present a novel method to simulate LiDAR point cloud with faster rendering time of 1 sec per frame. The proposed method relies on spherical UV unwrapping of Equirectangular Z-Buffer images. We chose BlenSor (Gschwandtner et al., 2011) as the baseline method to compare the point clouds generated using the proposed method. The reported error for complex urban landscapes is 4.28cm for a scanning range between 2–120 meters with Velodyne HDL64-E2 parameters. The proposed method reported a total time per frame to 3.2 ± 0.31 seconds per frame. In contrast, the BlenSor baseline method reported 16.2 ± 1.82 seconds.

Keywords: LiDAR, Point Cloud, BlenSor, Synthetic

1. Introduction

The 2D perception tasks in self-driving cars, such as object detection and semantic segmentation, have been increasingly improved over the past few years. One of the main reasons for the improvement, is the recent advancements in deep neural network architectures and models, as well as the availability of large amounts of annotated data for such tasks (Cordts et al., 2015; Geiger et al., 2012; Ros et al., 2016). On the other hand, 3D perception tasks are still lagging behind. This is mainly because of the scarcity of available annotated datasets.

Recently, a promising approach, for tackling the data problem of the 3D perception tasks, was proposed (Dosovitskiy et al., 2017; Gschwandtner et al., 2011; Shah et al., 2017). It is the utilisation of photo-realistic simulators and simulated 3D ranges sensors, such as LiDARs, for generating virtually unlimited amounts of labelled 3D point cloud data (Fang et al., 2020; Gaidon et al., 2016; Griffiths and Boehm, 2019).

However, the available simulators are still suffering from a number of challenges. For example in CARLA simulator (Dosovitskiy et al., 2017), the meshes are simplified and/or low poly assets of urban traffic objects are used, in order to minimise the rendering time for point clouds of the scene. Thus, the resulting point clouds are missing
Figure 2: a) Proposed workflow on a reference 3D model. The F, S, T 3D extrusions represent front, side and top views, respectively. The scene is rendered using an equirectangular 360 deg camera. Only Z-buffer is obtained and visual fidelity effects turned off for efficiency. The equirectangular depth map is mapped to spherical coordinates $\theta, \phi$. The depth values at the corresponding Cartesian coordinates are used to set the distance off the simulated LiDAR point. The illustrated scan assumes $[-\frac{\pi}{2}, \frac{\pi}{2}]$ vertical FOV. Practical sensors choose a predefined set of vertical angles. b) An example showing how the unit sphere is carved into the point cloud. The sphere radius is multiplied by 50 meters for illustration.

a lot of the details and realism exist in other simulators which leads to the domain shift problem tackled in (El-madawi et al., 2019; Saleh et al., 2019, 2018a). Another 3D sensor simulation framework, BlenSor, which does not suffer from the aforementioned problem, still required extended time duration for rendering only one point cloud scan (roughly 60 secs). Therefore, in this paper, we are introducing a novel approach that combines the best of the two worlds of (CARLA and BlenSor), where our approach can realistically simulate 3D LiDAR point cloud scans using much lower rendering time (roughly 1 sec). Our approach relies on equirectangularly projecting the Z-buffer of the rendered scene; then UV unwrapping is used, to fold the scene onto a unit sphere; and finally, adjust the depth of each point on the sphere using the depth value from the Z-buffer.

The rest of this paper is organised as follows. Section 2 gives an overview about how rendering of LiDAR point cloud, in simulators, works. Section 3 introduces our novel approach for fast realistic LiDAR rendering. Section 4 presents experiments and results and finally, Section 5 concludes and introduces future work.

2. Simulated LiDAR Rendering

There are several solutions for simulating LiDAR output. BlenSor is perhaps the pioneering solution which started the new wave of LiDAR simulations. One success factor behind the popularity of BlenSor is due to its implementation as a Blender add-on which guaranteed a seamless integration within Blender 3D modelling tool. However, the popularity of BlenSor did actually spike due to its ability to simulate different depth imaging sensors including the structured light near infrared and time-of-flight depth sensors (e.g. Kinect and Kinect 2, respectively). The implementation of these two types of depth imaging sensors and the seamless integration with an open source 3D modelling and animation tool benefited the computer vision research community with the hype around RGB-D cameras. Despite its expensive rendering time, the BlenSor framework was proven effective in many application domains such as workplace safety (Abobakr et al., 2019b, 2017; Haggag et al., 2014, 2017b; Nahavandi and Hossny, 2017), fall detection (Abobakr et al., 2019a, 2018), and even animal detection (Haggag et al., 2017a, 2018a; Saleh et al., 2016, 2018b).

In general, LiDAR simulation relies on different ray casting algorithms (Roth, 1982). The main idea behind these algorithms is to extend a line from the location of the simulated LiDAR sensor and reporting back the the intersection point with polygons of 3D objects in the simulated scene. This method, however, suffers from exponentially increasing rendering time as the scene becomes more complex. In an urban simulation, for example, one frame would take 45-70 seconds to render, as shown in Figure 1. Rendering LiDAR point clouds of buildings and grounds are fairly straightforward because of their simple mesh topologies. A fair amount of the rendering time is actually spent on the finer elements in the scene, such as trees, lamp posts, and other props. However, rendering the point clouds for vehicles and pedestrians is the most computationally expensive, due to the number of polygons per object (3-8K polygons).

The number of polygons per object is a well known aspect for 3D artists and video game designers. This is why, in these industries, it is considered best practice to
reduce the number of polygons per object (low-poly) and compensate the degraded fidelity using material composition, shaders and textures. With the rise of deep reinforcement learning and machine learning models, targeted at autonomous driving, these methods were utilised in optimising the simulated LiDAR sensors in the new wave of urban driving simulators such as CARLA (Dosovitskiy et al., 2017) and AirSim (Shah et al., 2017). However, while these methods solve the problem for rendering aesthetically acceptable scenes, they are not acceptable for rendering point clouds.

The major breakthrough in the new real-time rendering engines such as Unity (Juliani et al., 2018) and Unreal Engine (Karis and Games, 2013) was that they did not rely solely on compensating the reduced look and feel of the low-poly 3D models with advanced materials. These engines also introduced several rendering shortcuts to reduce the computational cost of light bouncing, reflections, shadows and ambient occlusions. In fact, these extra aspects of the 3D rendering pipeline were the main motivation behind developing the ray casting rendering algorithms back in the 1980’s (Roth, 1982). With offline processing of these scene enhancing aspects (also known as baking), the 3D game engine can achieve real-time frame rates with aesthetic quality close enough to the ray casting results. These improvements and shortcuts then allowed deploying sophisticated scenes into the virtual reality realm while rendering 360 deg live imagery at 90 frames per second (FPS) (Jerald et al., 2014; Martinez-Gonzalez et al., 2018).

3. Proposed Synthetic LiDAR Rendering

From a LiDAR rendering perspective, having an accurate point cloud representation of the mesh topology of the 3D objects, is of the utmost importance. This alleviates the need for running the computationally expensive ray casting algorithms. What is actually needed is the depth map (z-buffer) which is calculated very efficiently using the perspective projection camera equations (Baker et al., 1997). In the proposed method, we make use of the 360 deg rendering to obtain a z-buffer of a 360 deg rendering of the scene.

An equirectangular unwrapping of a unit sphere simply maps the Cartesian coordinates \((x, y, z)\) onto a 2D surface with planar coordinates \((u, v)\) \(\in \mathbb{R}^2\) where

\[
x = r \sin \theta \cos \phi \\
y = r \sin \theta \sin \phi \\
z = r \cos \theta
\]

where \(r\) is the radius of the sphere. In this work, we are using unit sphere with \(r = 1\) to map the z-buffer of a 360 deg rendering of the scene.

3.2. LiDAR Parameters

The equirectangular UV-mapping allows us to simplify the mapping of LiDAR parameters by adjusting the resolution of the equirectangular depth map. The horizontal \(\delta_\phi\) and vertical resolution \(\delta_\theta\) are simply a scaling factor of the equirectangular depth-map’s width and height and is calculated based on the reported specifications of the LiDAR sensors as

\[
N = \frac{\text{LiDAR H.FOV}}{\delta_\phi} \\
M = \frac{\text{LiDAR V.FOV}}{\delta_\theta}
\]

where \(N\) is the number of rays cast horizontally and \(M\) is the number of the vertical laser sensors in the simulated LiDAR (i.e. number of channels). It is worth noting that practically, LiDAR sensors rarely distribute the vertical field of view equally. Instead, they have predefined inclinations of the laser emitters and their respective sensors.

3.3. Point Cloud Construction

As shown in Fig. 2, the rendering workflow starts with rendering an equirectangular depth map into a point cloud. For any 360 deg sensor \(\phi_{0..N}[\theta_0, \theta_N]\) with a horizontal field of view \((\theta_0, \theta_N)\) and a vertical field of view \((\phi_0, \phi_M)\) we need

\[
x = r \sin \theta \cos \phi \\
y = r \sin \theta \sin \phi \\
z = r \cos \theta
\]
an equirectangular panoramic rendering \( I_{M \times N}^{D,L}(u,v) \) with similar field of view parameters, where \( D \) and \( L \) denote to depth and label maps, respectively. When constructing the point cloud, we assume an \( M \times N \) 2D mesh grid \( G_{M \times N} \) as the Cartesian product of \( u = (\theta_0, \theta_M) \) and \( v = (\phi_0, \phi_N) \) as

\[
G_{M \times N} = \{ (\theta_0, \theta_M) \times (\phi_0, \phi_N) \} = \{ (\theta_0, \phi_0), \ldots, (\theta_u, \phi_v), \ldots, (\theta_N, \phi_N) \},
\]

for \( u, v \in (\theta_0, \theta_M), (\phi_0, \phi_N) \)

The 2D UV coordinates in the assumed grid are then used to construct and carve a unit sphere using Eq. 3 as

\[
x_{u,v} = I^D(u,v) \sin \theta_u \cos \phi_v, \\
y_{u,v} = I^D(u,v) \sin \theta_u \sin \phi_v, \\
z_{u,v} = I^D(u,v) \cos \theta_u, \\
l_{u,v} = I^L(u,v),
\]

where \( l_{u,v} \) is the label for all \( u,v \in G_{M \times N} \) and where \( I^{D,L} \) is the equirectangular depth and label maps, respectively. The proposed approach allows adjusting the angular resolution by modifying the dimensions \( M,N \) of the equirectangular depth and label maps. The vertical dimension \( M \) is particularly important to correct for the vertical misalignment problem.

### 3.4. Vertical Resolution and Angle Misalignment

In order to demonstrate the effect of equirectangular vertical resolution on the alignment of the vertical LiDAR channels, we designed a simple scene with the simulated sensors placed at the centre of a 200 m cube. We set the vertical FOV to 180 deg and number of channels to 8 and 64 to simulate coarse and fine point cloud scenarios.

The same setup and parameters were used for the BlenSor baseline. The controllable parameter in this experiment is the vertical dimension of the rendered equirectangular depth map \( M \) in Eq. 7. Figure 3 demonstrate the vertical misalignment problem associated with using lower vertical resolution (Fig. 3). The reason behind this problem is the low number of vertical lines in the equirectangular depth map which results in a polygonal UV-sphere and affects the point cloud projection accordingly. As shown in Fig. 3, the RMSE error drops exponentially as the vertical resolution increases. The error drop follows a negative power regression curve with 10.389\( x^{-2.0336} \) \( R^2 = 0.9533 \) and 77.561\( x^{-3.0226} \) \( R^2 = 0.8434 \) for 8- and 64-channel LiDAR setups, respectively. It is worth noting that this problem is more apparent with lower numbers of laser channels because the error yields an over estimation of the depth at the two extremes of the vertical FOV as shown in Fig. 3.

#### 3.5. Noise Augmentation

The proposed method also simplifies noise augmentation. Because the proposed method relies solely on texture unwrapping, simulated LiDAR noise is simplified as 2D noise imposed on the equirectangular 360 deg depth image via 2D convolutional kernels implemented. This allows a plethora of available after-effects 2D convolutional kernels to be utilised during point cloud generation and processing. For example, noise in depth estimation are simulated via additive Gaussian noise \( N(0, \sigma) \) where the magnitude of the noise is controlled by the variance \( \sigma^2 \).

### 4. Experiments and Results

In order to validate the proposed method we chose the nearest neighbour (NN) octal-tree (OCTREE) for point cloud comparison. We chose BlenSor as
the base-line and generated two categories of 3D scenes. The first category is geometric shapes increasing in complexity characterised by number of polygons and the variance between polygon angles. The second category includes several urban city scenes.

4.1. Randomised Polyhedron ICO Sphere Test

We chose the polyhedron icosphere geometric shape for benchmarking because of its triangulation and non conformity with the UV coordinates. We generated seven icosphere with subdivision resolutions of 1, 2, 4, 8, 16, 32 and 64 yielding an exponential increase in number of triangles between 80 to 84,500 triangles. We then applied a uniformly distributed randomisation to the vertices to introduce discrepancies in the depth values. Each vertex was relocated randomly along its surface normal to ensure that mesh faces are not intersecting and do not produce negative normal vectors as the problem complexity increases. The same mesh topology was used with a linearly increasing radius from 1 to 128 meters. The 3D shapes are demonstrated in Figure 4-a. The simulated LiDAR sensor was placed at the origin of the polyhedron icosphere. We used Velodyne HDL-64E2 parameters for the sensor. Depth image dimensions were set to 2117 × 64 and the field of view was set to 2 deg −25 deg. The horizontal and vertical LiDAR resolution were set to 0.412, 0.09 deg, respectively. The same setup was used for both the proposed and the baseline solution (BlenSor).

The results summarised in Table 1 highlight an average error and the standard deviation range of the proposed solution in comparison to the BlenSor baseline. In scenes featuring low number of polygons, the error increases linearly from 1.5 cm at 2 meters distance to 10 cm at 120 meters as shown in Fig. 4-b-bottom. This error increases as the number of randomised polygons to range from 5.2 cm at 2 meters radius to 35.3 cm at 120 meters. It is worth noting that the error increase follows a logistic pattern as the number of polygons increases as shown in Fig. 4-b-top. This is because as the resolution of a 3D mesh increases, the area per mesh face decreases and results in many co-planar vertices and faces. Practically however, it is considered best practice to minimise the number of mesh faces while modelling a 3D scene which in return emphasises the efficacy of the proposed solution for 3D meshes up to

| ICO/Radius | (0, 32) | (32, 64) | (64, 96) | (96, 128) |
|------------|--------|---------|---------|---------|
| 1          | 1.451 ± 0.825 | 4.298 ± 0.861 | 7.119 ± 0.817 | 9.955 ± 0.884 |
| 2          | 1.477 ± 0.835 | 4.348 ± 0.865 | 7.129 ± 0.866 | 10.072 ± 0.927 |
| 4          | 1.572 ± 0.893 | 4.693 ± 0.966 | 7.723 ± 0.876 | 10.798 ± 1.063 |
| 8          | 1.905 ± 1.075 | 5.670 ± 1.159 | 9.305 ± 1.091 | 13.109 ± 1.290 |
| 16         | 2.742 ± 1.551 | 8.168 ± 1.706 | 13.662 ± 1.922 | 19.098 ± 1.913 |
| 32         | 4.154 ± 2.414 | 12.094 ± 2.324 | 20.361 ± 2.473 | 27.996 ± 2.626 |
| 64         | 5.198 ± 2.968 | 15.280 ± 3.032 | 25.248 ± 2.893 | 35.323 ± 3.075 |
2000 polygon as illustrated by the box plot in Fig. 4-d where the overall inferred probability density function lies between the inferred probability density functions of the polyhedron icospheres with 1620–5780 polygons.

One last point to note from the polyhedron icosphere experiment is the magnitude of the complexity relative to the urban scene. With the majority of assets in an urban scene exhibit quasi-planar surfaces, icospheres with offset randomised subdivisions higher than 8 (more than 2000 polygons) exist in urban scenes as tree leaves where a physical LiDAR sensor would perceive as an unstructured blob of point cloud with high error due to the wind. This point prompted us to design the next experiment to derive per-label error distribution and its associated efficacy relative to the autonomous driving use-case.

4.2. Simulated Urban City

The purpose of this experiment is to study the efficacy of the proposed solution in realistic urban scenes. For this experiment, we designed a procedural urban scene generator with 3-ways and 4-ways road intersections, 10 pedestrian models featuring different anthropometric features and activities, 4 cyclist gestures (Saleh et al., 2019), 20 vehicle models, 10 street props (e.g. mailbox, signs, road work, etc), 6 types of trees and 7 types of buildings. We placed a 64-channels simulated sensor on a motion path passing through the intersections. We set the sensor horizontal and vertical resolution to 0.17 and 0.42 degrees, respectively. Horizontal and vertical FOV were set to [0,360] and (−25, 2] degrees, respectively. The same setup and parameters were used for the BlenSor baseline with a “Generic Lidar” configuration. For the proposed method we chose a vertical resolution M = 1024 pixels. The RMSE error was measured using the nearest-neighbour octree method (Meagher, 1980).

The reported average RMSE error is 4.28 cm with statistical outliers and 2.32 cm after excluding statistical outliers. Because the data failed the normality tests (Anderson and Darling, 1952; D’Agostino, 1970; Shapiro and Wilk, 1965), we identified the outliers using the 10 sigmas Chebyshev bounds rule \(1 - \frac{1}{\sqrt{k}}, k = 10\) (Olkin and Pratt, 1958) which yields to 99% coverage of the reported error. The statistical outliers were mainly the points where the recorded error is larger than the maximum scanning distance of 120 meters. This is due to the fact that the point clouds of the proposed solution and the baseline BlenSor method are unstructured, not necessarily have the same number of points and are not sharing the same index (not paired). In fact these are the reasons we chose the OC-TREE nearest neighbour method for comparison.

Figure 5 visualises the results of this experiment. For error visualisation of the simulated point cloud, we squared the RMSE error (in centimetres) at each point cloud and then normalised by the correct distance between each point cloud and the simulated sensor. The result was then used to adjust the saturation of the point cloud shown in Fig. 5 which highlights the error as a gradient colour ranging from white (correct) to red (incorrect). We then examined the error distribution for different labels and scan ranges. As shown in Fig. 5-right, the majority of the significant error lies in the range (80, 120] meters. This is considered an acceptable error margin considering the physical limitations of the LiDAR technology with attenuated reflectivity rule (Lichti, 2008; Mittet et al., 2016) dictated by restricting the laser grade to Class-1 for human safety (Thomas et al., 2002). This error is attributed to two factors. First, the depth shadow errors accumulated from objects at range [2, 80] and second the resolution of the rendered equirectangular depth map. This is also reflected in Fig. 6-a where the error saturates around 4 cm with outliers and 2 cm without outliers. Practically, how-
ever, the effective scan range for real-life applications of LiDAR technology such as self-driving vehicles is below 90 m. For example, the planning horizon of most motion planners in self-driving vehicles is within the [6, 80] meters range \cite{Bansal2018, Zeng2019}, which is aligned with the recommended safe braking distances by NHTSA for vehicles cruising on speed range between [20, 70] miles per hour \cite{Richer2017}.

Figure 6-b shows the error distribution across different labels where the box plot shows the error distribution without outliers (left axis) and the black dots highlight the average error with outliers (right axis). Expectedly, the majority of error lies in both street props and cyclists. This is due to the thin structures of these objects such as fences and bicycle wheels. The error of trees point cloud did exhibit the highest variance as we explained in the previous subsection. Overall the recorded error for the majority of the point cloud for individual labels is less then 1 cm.

4.3. Performance Profiling and Mass Production

In the synthesised urban city experiment, the proposed method reported a sustainable 2.7 ± 0.21 seconds to render a 2117 × 1024 equirectangular depth/label map which produces a total of 135488 3D points (64 channels). An additional 0.5 ± 1.1 seconds were reported for the construction of point cloud. This brings the total time per frame to 3.2 ± 0.31 seconds per frame. In contrast, the BlenSor baseline method reported 16.2 ± 1.82 seconds. This provides a total of 80% increase in frame production which in return renders the proposed solution viable for mass production of accurate synthesised labelled point cloud datasets. Upon decreasing the vertical resolution to $M = 128$, the proposed method reported 2.16 seconds per frame which increases the production rate by 33% at the cost of an increased average RMSE error of 50% compared to $M = 1024$. These results include disk writing overhead and were obtained on using a 6-core CPU with no GPU support.

5. Conclusions

In this paper, we presented a novel method for rendering LiDAR point cloud from synthetic scenes. The method is based on equirectangular UV unwrapping of the z-buffers obtained from 360 deg rendered scenes. The resulting point clouds recorded an average root mean squared error of 2 – 4 cm when compared to the baseline (BlenSor) for urban scenes. The majority of the error lies at the scan range from 80–120 meters. Additionally, the proposed method allows for more articulation in the simulated LiDAR point cloud. This is because the entire point cloud information can be encapsulated into a 2D texture of depth values (obtained from the z-buffer). This, in turn, allows us to apply different types of noise and image processing filters to the texture. These filters can also be applied to equirectangular label maps while using dithering to simulate mislabelling anomalies in point clouds.

One limitation with synthetic LiDAR point clouds in general is their lack of reporting intensity of the laser beams. This is being addressed in Unity using material compositions \cite{Juliani2018}. The resulting accuracy and rendering frame rate does show great potential for investigating this idea further. Intensity reporting and noise modelling and augmentation will be investigated and integrated to the proposed solution in the future.
References

Abobakr, A., Hossny, M., Abdelkader, H., Nahavandi, S., 2019a. Rgb-d fall detection via deep residual convolutional lstm networks. Abobakr, A., Hossny, M., Nahavandi, S., 2018. A skeleton-free fall detection system from depth images using random decision forest. IEEE Systems Journal 12, 2994–3005. Abobakr, A., Nahavandi, D., Hossny, M., Iskander, J., Attia, M., Nahavandi, S., Smets, M., 2019b.Rgb-d ergonomic assessment system of adopted working postures. Applied Ergonomics 80, 75–88. Abobakr, A., Nahavandi, D., Iskander, J., Hossny, M., Nahavandi, S., Smets, M., 2017. A kinect-based workplace postural analysis system using deep residual networks. Anderson, T.W., Darling, D.A., 1952. Asymptotic theory of certain “goodness of fit” criteria based on stochastic processes. Ann. Math. Statist. 23, 193–212. URL: https://doi.org/10.1214/aoms/1177729437

Baker, M., et al., 1997. Computer graphics, c version. Bansal, M., Krizhevsky, A., Ogale, A., 2018. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. arXiv preprint arXiv:1812.08079.

Cordts, M., Omran, M., Ramos, S., Scharwächter, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B., 2015. The cityscapes dataset, in: CVPR Workshop On The Future of Datasets in Vision. D’Agostino, R.B., 1970. Transformation to normality of the null distribution of g1. Biometrika 57, 679–681. Desolneux, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V., 2017. CARLA: An open urban driving simulator, in: Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16. Elmadawi, K., Abdelrazek, M., Elsobky, M., Éraqi, H.M., Zahran, M., 2019. End-to-end sensor modeling for lidar point cloud, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE, pp. 1619–1624.

Fang, J., Zhou, D., Yan, F., Zhao, T., Zhang, F., Ma, Y., Wang, L., Yang, R., 2020. Augmented lidar simulator for autonomous driving. IEEE Robotics and Automation Letters 5, 1931–1938.

Gaidon, A., Wang, Q., Cabon, Y., Vig, E., 2016. Virtual worlds as proxy for multi-object tracking analysis, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4340–4349.

Geiger, A., Lenz, P., Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite, in: Conference on Computer Vision and Pattern Recognition (CVPR).

Griffiths, D., Boehm, J., 2019. Synthcity: A large scale synthetic point cloud. arXiv preprint arXiv:1907.04768.

Gschwandtner, M., Kwitt, R., Uhl, A., Pree, W., 2011. Blensor: Blender sensor simulation toolbox, in: International Symposium on Visual Computing, Springer, pp. 199–208.

Haggag, H., Abobakr, A., Hossny, M., Nahavandi, S., 2017a. Semantic body parts segmentation for quadrupedal animals, pp. 855–860.

Haggag, H., Hossny, M., Haggag, S., Nahavandi, S., Creighton, D., 2014. Safety applications using kinect technology, pp. 2164–2169.

Haggag, H., Hossny, M., Nahavandi, S., Haggag, O., 2017b. An adaptable system for rgb-d based human body detection and pose estimation: Incorporating attached props, pp. 1544–1549.

Jeraldi, J., Giokaris, P., Woodall, D., Hartboll, A., Chandak, A., Kunsta, S., 2014. Developing virtual reality applications with unity, in: 2014 IEEE Virtual Reality (VR), IEEE, pp. 1–3.

Juliani, A., Berges, V.P., Vckay, R., E., Gao, Y., Henry, H., Mattar, M., Lange, D., 2018. Unity: A general platform for intelligent agents. arXiv preprint arXiv:1809.02627.

Karls, B., Games, E., 2013. Real shading in unreal engine 4. Proc. Physically Based Shading Theory Practice 4.

Lichti, D.D., 2008. A method to test differences between additional parameter sets with a case study in terrestrial laser scanner self-calibration stability analysis. ISPRS Journal of Photogrammetry and Remote Sensing 63, 169–180.

Martinez-Gonzalez, P., Oprea, S., Garcia-Garcia, A., Jover-Alvarez, A., Orts-Escolano, S., Garcia-Rodriguez, J., 2018. Unreal-rox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. arXiv preprint arXiv:1810.06936.

Meagher, D.J., 1980. Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer. Electrical and Systems Engineering Department Rensselaer Polytechnic.

Mittet, M.A., Nouira, H., Roynard, X., Goullete, F., Deschaud, J.E., 2016. Experimental assessment of the quanergy m8 lidar sensor. Nahavandi, D., Hossny, M., 2017. Skeleton-free rul ergonomic assessment using kinect sensors. Intelligent Decision Technologies 11, 275–284.

Okin, I., Pratt, J.W., 1958. A multivariate tchebycheff inequality. Ann. Math. Statist. 29, 226–234. doi:10.1214/aoms/1177767220

Richer, C., Weissman, D., Ranaiaef, F., Gale, N., 2017. Los Angeles Vision Zero Technical Analysis: A Data-Driven Effort to Eliminate Traffic Fatalities. Technical Report.

Ros, G., Sellart, L., Materzynska, J., Vandez, D., Lopez, A.M., 2016. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 3234–3243.

Roth, S.D., 1982. Ray casting for modeling solids. Computer graphics and image processing 18, 109–144.

Saleh, K., Abobakr, A., Attia, M., Iskander, J., Nahavandi, D., Hossny, M., Nahavandi, S., 2019. Domain adaptation for vehicle detection from bird’s eye view lidar point cloud data, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, pp. 0–0.

Saleh, K., Abobakr, A., Nahavandi, D., Iskander, J., Attia, M., Hossny, M., Nahavandi, S., 2019. Cyclist intent prediction using 3d lidar sensors for fully automated vehicles, in: 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pp. 2020–2026. doi:10.1109/ITSC.2019.8917291.

Saleh, K., Hossny, M., Hossny, A., Nahavandi, S., 2018a. Cyclist detection in lidar scans using faster r-cnn and synthetic depth images, pp. 1–6. doi:10.1109/ITSC.2017.8317599 cited By 13.

Saleh, K., Hossny, M., Nahavandi, S., 2016. Kangaroo vehicle collision detection using deep semantic segmentation convolutional neural network. doi:10.1109/DICUA.2016.7797057 cited By 12.

Saleh, K., Hossny, M., Nahavandi, S., 2018b. Effective vehicle-based kangaroo detection for collision warning systems using region-based convolutional networks. Sensors (Switzerland) 18, doi:10.3390/s18060913 cited By 6.

Schroder, P., Sweldens, W., 1995. Spherical wavelets: Texture processing, in: Rendering Techniques 95. Springer, pp. 252–263.

Shah, S., Dey, D., Lovett, C., Kapoor, A., 2017. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, in: Field and Service Robotics. URL: https://arxiv.org/abs/1705.05055

Shapiro, S.S., Wilk, M.B., 1965. An analysis of variance test for normality (complete samples). Biometrika 52, 591–611.

Thomas, R.J., Rockwell, B.A., Marshall, W.J., Aldrich, R.C., Zimmerman, S.A., Rockwell Jr, R.J., 2002. A procedure for laser hazard classification under the z136.1-2000 american national standard for safe use of lasers. Journal of Laser Applications 14, 57–66.

Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R., 2019. End-to-end interpretable neural motion planner, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 8660–8669.