ADAPTIVE ACQUISITION OF AIRBORNE LIDAR POINT CLOUD BASED ON DEEP REINFORCEMENT LEARNING

Chengxuan Huang[a], *Dalei Wu*[b*], *and Yu Liang*[b]

[a]University of California, Davis, California, USA 95616
cxxhuang@ucdavis.edu
[b]University of Tennessee at Chattanooga, Tennessee, USA 37403
dalei-wu@utc.edu, yu-liang@utc.edu

ABSTRACT

Human experience involvement in existing operations of airborne Light Detection and Ranging (LIDAR) systems and off-line processing of collected LIDAR data make the acquisition process of airborne LIDAR point cloud less adaptable to environment conditions. This work develops a deep reinforcement learning-enabled framework for adaptive airborne LIDAR point cloud acquisition. Namely, the optimization of the airborne LIDAR operation is modeled as a Markov decision process (MDP). A set of LIDAR point cloud processing methods are proposed to derive the state space, action space, and reward function of the MDP model. A DRL algorithm, Deep Q-Network (DQN), is used to solve the MDP. The DRL model is trained in a flexible virtual environment by using simulator AirSim. Extensive simulation demonstrates the efficiency of the proposed framework.

1. INTRODUCTION

The basic operation of a LIDAR is to determine ranges (variable distance) by targeting an object or a surface with a laser and measuring the time for the reflected light to return to the receiver. Collected LIDAR data creates a 3-D point cloud model of the scene. Within the realm of remote sensing, airborne LIDARs have found popular uses in multiple applications, such as building detection, road extraction, disaster management, power line patrol, and land cover mapping [1, 2]. Most of existing airborne LIDAR systems require skilled humans to set up operational configurations. Also, processing LIDAR point cloud and extracting information of interest are challenging and involve a series of sophisticated steps. As a result, most airborne LIDAR systems process data in off-line steps after field scanning, making LIDAR data acquisition and processing time-consuming and less adaptable to field conditions. Furthermore, many applications require sensing within inaccessible or hazardous environments for human. Therefore, adaptive airborne LIDAR data acquisition and processing is preferred.

The decision-making process of operating an autonomous airborne LIDAR can be modeled as a finite-horizon Markov decision process (MDP) with finite state and action spaces, but the curse of extremely high dimensionality of state space makes it computationally infeasible to derive optimal action using standard finite-horizon MDP algorithms [3]. To overcome this challenge, in this work deep reinforcement learning (DRL) is considered to handle the large dimensionality of the state space and learn the optimal policy at the same time. As a computational methodology for automatic decision-making of intelligent agents in uncertain environments, DRL has progressed tremendously in the past decade [4]. The concern of DRL is how the agent ought to take actions from a given state of an environment so as to maximize some notion of cumulative reward. The full potential of DRL requires an agent to interact directly with the environment to attain a flow of real-world experiences.

Some research has been done on adaptive mobile LIDAR systems adopting DRL. Few work has been focused on airborne LIDAR systems. In [5], an end-to-end DRL-based autonomous driving method was proposed for navigating unregulated urban intersections using raw LIDAR point clouds. It was reported that the method was capable of handling imperfect partial observations such as occlusions. In [6], an end-to-end DRL method was studied for collision avoidance for mobile robots operating in dense and crowded environments by using multiple perception sensors such as a 2-D LIDAR along with a depth camera.

In this work, we propose a DRL framework for adaptive acquisition of airborne LIDAR point cloud. The key contributions are summarized as follows:

- We formulate the optimization of airborne LIDAR point cloud acquisition as a sequential decision-making problem which is further modeled as a Markov decision process (MDP).

*Corresponding Author

This work was supported by the National Science Foundation under grant numbers 1647175 and 1924278.

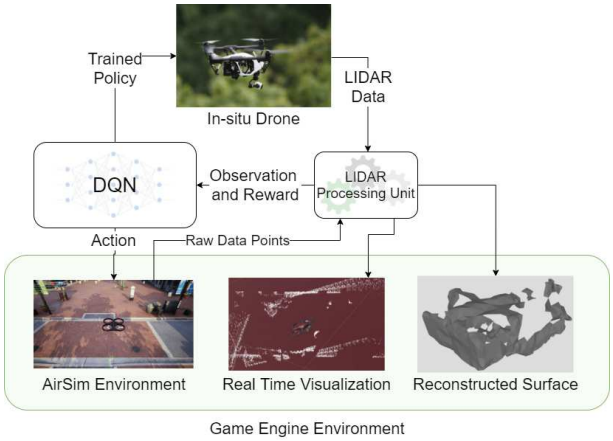

Fig. 1: The proposed airborne LIDAR point cloud acquisition framework.

- We develop a set of LIDAR point cloud processing methods to formulate a DRL model. The state space, action space, and a novel reward function of the DRL model are derived.
- The DRL model is trained and tested in a flexible virtual environment by using autonomous vehicles simulator AirSim. Extensive simulation demonstrates the efficiency of the proposed framework.

2. OVERVIEW OF THE FRAMEWORK

This work aims to develop an autonomous LIDAR scanner carried by a drone to navigate an area of interest and collect point cloud data without prior knowledge of the environment. As shown in Figure 1, we propose to adopt reinforcement learning to allow the LIDAR agent to learn how to gather information efficiently without any collision with the environment.

The LIDAR agent is simulated and trained in a controlled virtual environment to offer a simplified interface to control the drone and retrieve LIDAR data. We used AirSim [7], a simulator for various vehicles built on Unreal Engine. Unreal Engine's level editor also provides visualization and raycasting, as well as the simulation/virtual environment, as shown in Figure 2. We used the "Downtown West Modular Pack" created by PurePolygons as the environment, which is available in the Epic Games Marketplace.

For each time step of the DQN, the virtual LIDAR agent sends its received LIDAR data to a LIDAR processing unit that analyzes the observation and calculates a reward for the agent. The LIDAR processing unit is built on the Computational Geometry Algorithms Library (CGAL). For each incoming LIDAR data point to be considered as new valid information, it needs to be at least somewhat distant from the rest of the already collected points. To process and represent the

Fig. 2: A virtual environment generated by AirSim.

observations from the agent, we propose two generic matrices that describe the surrounding of the agent. Each entry of the matrix represents a segment of the agent's surrounding. One matrix describes the density of points and the other describes the distance of the closest point in each segment. The observation also includes the position of the LIDAR agent since the agent is rewarded for acquiring the points that are closer from the origin. The observation and reward of each time step are sent to the replay memory module of the DQN. The DQN decides the action of the LIDAR agent of the next time step based on a balanced exploitation and exploration strategy and sends the action back to the agent in the AirSim environment.

It is worth noting that the trained DQN model could be deployed with a real airborne LIDAR in real-world environments. The point cloud acquired by the LIDAR can be processed within the LIDAR processing unit through rendering and surface reconstruction in a game engine for real time visualization.

Algorithm 1 Updating of point cloud $\mathcal{O}_t$

Require: Newly captured points: $\mathcal{P}_t$ at time step t, the point cloud at previous time step: $\mathcal{O}_{t-1}$, and the predetermined cut-off distance δ_{ob}

Initialize the valid newly captured point set: $\mathcal{V}_t := \{\}$;
for each newly collected point $\mathbf{p} \in \mathcal{P}_t$ **do**
  $min\_dist = argmin_{\mathbf{q} \in \mathcal{O}_{t-1} \cup \mathcal{V}_t} \|\mathbf{p} - \mathbf{q}\|$
  if $min\_dist \geq \delta_{ob}$ **then**
    $\mathcal{V}_t \leftarrow \mathcal{V}_t \cup \{\mathbf{p}\}$;
  end if
end for
$\mathcal{O}_t = \mathcal{O}_{t-1} \cup \mathcal{V}_t$;

3. NAVIGATION OPTIMIZATION BASED ON DRL

Given the environment and the airborne LIDAR agent, a policy needs to be constructed for the agent to act to maximize the information acquired about the environment.

Since the state and action spaces are finite, we can model

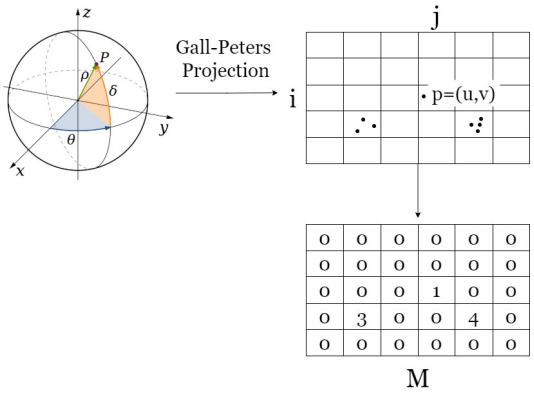

Fig. 3: Projection of points to the projection map

the optimization of the operation of the agent as a sequential decision-making problem, which can be further modeled as a Markov decision process (MDP). The MDP model is described as follows:

- State space $\mathcal{S}$: the set of the states about the LIDAR agent's operational status and the observed environment. Let $s_t = (\kappa_t, \Psi_t) \in \mathcal{S}$ represent the state of the agent and its observation about the environment at time t. Here κ_t represents the newly acquired LIDAR points, and Ψ_t represents the operating state of the agent, which may include the drone's orientation, position, velocity, and throttle of each fan.
- Action space $\mathcal{A}$: a set of actions of the agent. Let $a_t = (i_t, j_t, k_t)$ denote the agent's action of moving in at most one of three orthogonal directions in the three dimensional space at any time t. Thus, only up to one of i_t, j_t, and k_t can be ± 1. Then the position of the agent at time $t+1$ can be denoted as $P_t + a_t \cdot C_{step}$, where P_t represents the position of the agent at time t, and C_{step} is a constant that represents the drone's flight distance for each time step.
- State transition probability $Pr_t(s, a, s') = Pr(s_{t+1} = s' | s_t = s, a_t = a)$: the probability of transition from state s to state s' under action a.
- Reward $r_t(s, a, s')$: the immediate reward received by the agent after transitioning from state s to state s', due to action a, at time t.
- T: the horizon over which the drone will act.

Then, the goal of the optimization problem is to find a policy for the agent: a function $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that, given the state s_t, outputs an action a_t that maximizes the accumulative knowledge about the environment in the given finite horizon. Mathematically, we need to maximize $\mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t(s_t, a_t)]$, where $\mathbb{E}[\cdot]$ is the expectation taken over $s_{t+1} \sim Pr(s_{t+1} | s_t, a_t)$ and $\gamma (0 \leq \gamma \leq 1)$ is the discount factor of the reward r_t at different time steps. Due to the extreme curse of dimensionality in the state space $\mathcal{S}$ and the immense challenge of identifying transition probability $P(s_{t+1} | s_t, a_t)$, it is impractical to use exact methods such as linear programming and dynamic programming to solve the MDP problem. To address this challenge, we investigate a DRL framework where the agent is reinforced to learn a policy in the virtual environment.

3.1. State Definition

In this paper, we assume that the agent has no prior knowledge of the environment. Therefore, the state of the agent can only be derived by collected LIDAR points and the operating information itself, such as the position of the drone. As mentioned previously, the state is characterized by two parts: the information about the environment through the collected LIDAR points, κ, and the operating state of the agent, Ψ.

3.1.1. LIDAR Point Information Processing

Since the goal of the LIDAR agent is to gather as much information of the environment as possible, a definition of valid information is needed.

At the beginning of each episode, an empty set $\mathcal{O}_0$ will be initialized, which represents that there is no information gained at time 0. At each time t, the agent receives LIDAR data $\mathcal{P}_t$ from its surrounding. Algorithm 1 uses $\mathcal{P}_t$ and $\mathcal{O}_{t-1}$ as input, filters out points that are clustered, and outputs an updated point cloud $\mathcal{O}_t$. For each collected point $\mathbf{p} \in \mathcal{P}_t$, we check if there are any points in $\mathcal{O}_{t-1}$ that are in the vicinity of $\mathbf{p}$. If none, $\mathbf{p}$ is regarded as newly acquired information about the environment. We define a hyperparameter δ_{ob} as the smallest distance for point $\mathbf{p}$ to be away from the closest point in $\mathcal{O}_{t-1}$ so that $\mathbf{p}$ can be considered as newly acquired information.

In order to speed up the process of finding points, we use the k-d tree data structure to store $\mathcal{O}_t$. As a result, for each point $\mathbf{p}$, the computational complexity of finding the closest point in $\mathcal{O}_t$ to $\mathbf{p}$ is only $O(\log |\mathcal{O}_t|)$.

3.1.2. Observation Representation

In this paper, we propose a novel method of representing the point cloud collected by the agent. The observation of the agent should only consist of the points that are in the measurement range of the LIDAR laser. Furthermore, as shown in [8], most deep learning techniques applied to point clouds need to extract features from those point clouds, instead of using raw data sets, due to the nature of point cloud data, such as high dimensionality, sparseness of the data, and irregularity of its shapes. Such characteristics make point cloud data inefficient to be represented. Therefore, it is preferred to transform the point cloud data into lower dimensional spaces for better performance in deep learning.

In this paper, we construct two $\mathbb{R}^{n \times m}$ matrices to represent the points surrounding the LIDAR agent. The sphere

around the agent is segmented into $n \times m$ equal sized segments, and the points in each segments are used to calculate the entry of the matrices. For the first matrix, we represent the density of the surrounded points in each segment around the agent. Since there is a minimal distance between points, moving the LIDAR too close to an object (such as a wall) would result in a segment containing less points, and the drone would be in greater risk of colliding with the environment. Therefore, the agent is incentivized to find distances to the object that are optimal for point collection. The second matrix represents the closest point of the surrounded points in each region. The rationale behind this matrix is to make the agent avoid moving in certain directions when it detects some points that are too close to it.

To project the points onto the matrices, we employ the Gall-Peters projection [9] which is a rectangular map projection that preserves the size of each shape on the sphere.

We first project the surrounding points in 3D to a 2D rectangle. Let $E_t \subseteq \mathcal{O}_t$ be the points that are at most C_{range} away from the LIDAR agent. Assume the agent is at point $\mathbf{x}$. For each point $\mathbf{y} \in E_t$, let $(x', y', z') := \mathbf{y} - \mathbf{x}$. Then the point's position on the rectangle (u, v) can be derived as:

$$\begin{cases} u = & \arctan 2(y', x') \\ v = & 2z' \|\mathbf{y} - \mathbf{x}\|^{-1} \end{cases} \tag{1}$$

where $\arctan 2$ is the two-argument function that gives the unambiguous angle for the polar coordinates when converting from Cartesian coordinates.

To apply machine learning, we convert the rectangle into matrices so that the number of inputs are finite and constant. Therefore, we further partition the rectangle into partitions, where entries of the two matrices regarded as the observation of the agent will extract information from points in each partition. As shown in Figure 3, to construct the partitions, we need to determine which row and column each point will be in. Let the position of point p on the rectangle be (u, v). Then the index (i, j) of the point in the partitions is:

$$\begin{cases} i = & \left\lceil \frac{u+\pi}{2\pi/n} \right\rceil \\ j = & \left\lceil \frac{v+2}{4/m} \right\rceil \end{cases} \tag{2}$$

which indicates that point p is in the ith row and the jth column in the partition. As Figure 3 shows, this process results in a **projection map** $\mathbf{M}$ that has $n \times m$ partitions, where each partition contains the list of points. Then a **density matrix** $\mathbf{M}_d$ can be constructed as:

$$\mathbf{M}_d(i, j) = \frac{|\mathbf{M}(i, j)|}{C_{max}} \tag{3}$$

and a **closeness matrix** $\mathbf{M}_c$ can be constructed as:

$$\mathbf{M}_c(i, j) = \min(\{|p| : p \in \mathbf{M}(i, j)\}) \tag{4}$$

where C_{max} is a constant that can be calculated using the detecting range C_{range} of the LIDAR agent, the minimum closeness of each point d_{ob}, and the dimensions (n and m) of the matrices.

3.2. Action Definition

The set of actions include the movement of the drone in one of the three orthogonal directions in the three dimensional space. To reduce the action space, action values are discretized. In addition, we may restrain the drone from moving up and down and control the drone to stay at a constant height if it is unnecessary for the drone to move vertically in collecting LIDAR points.

Given the position of the drone $\mathbf{x}_t = (x_t, y_t, z_t) \in \Psi_t$ at time t, the position of the drone for the next time step can be generally derived as:

$$\mathbf{x}_{t+1} = \begin{cases} (x_t + C_{step}, y_t, z_t) & \text{or} \\ (x_t, y_t + C_{step}, z_t) & \text{or} \\ (x_t, y_t, z_t + C_{step}) & \text{or} \\ (x_t - C_{step}, y_t, z_t) & \text{or} \\ (x_t, y_t - C_{step}, z_t) & \text{or} \\ (x_t, y_t, z_t - C_{step}) & \text{or} \\ (x_t, y_t, z_t) \end{cases} \tag{5}$$

3.3. Reward Function

The agent is rewarded based on the newly observed information gain and penalized if it collides with the environment. Specifically, the reward function $r(s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is decided by the amount of newly acquired information $\mathcal{V}_t$ and the collision status of the agent. Thus, the reward function at time t is

$$r_t(s_t, a_t) = \sum_{\mathbf{p} \in \mathcal{V}_t} R(\mathbf{p}) + C_{col} \cdot \mathbf{1}_{\mathcal{S}}(\mathbf{x}_t) \tag{6}$$

where $\mathcal{V}_t$ is the newly acquired valid points and is defined in Algorithm 1; $R(\mathbf{p})$ is the scaled information gain function and will be defined below in Eq. (8); and $\mathbf{1}_{\mathcal{S}}$ is the collision indicator function and is defined as

$$\mathbf{1}_{\mathcal{S}}(\mathbf{x}_t) = \begin{cases} 1 & \text{if } \mathbf{x}_t \text{ contacts the environment} \\ 0 & \text{otherwise;} \end{cases} \tag{7}$$

C_{col} is a negative constant representing the penalty when the agent is collided with the environment.

We propose a new evaluation method that measures the information gain of each new point. From the experiment, we observed that the drone prefers to go in one general direction because going any other direction may cause overlap between some of the scanned points and previous points. However, in this paper, we aim to have a holistic picture of the surrounding

Table 1: Configuration of DQN

Learning rate	0.0001
Batch size	64
Train frequency	5 episodes
Buffer size	10^3
Polyak update	1
Discount factor	0.99
Train frequency	10 episodes

environment, rather than a corridor of LIDAR points stretching in one direction. To that end, we introduce an exponentially scaled point evaluation system that assigns more reward to the points that are closer to the origin.

For any point $\mathbf{p}$, the reward given by that point is defined as:

$$R(\mathbf{p}) = \frac{\alpha}{\xi^{\|\mathbf{p}\|/\delta_{td}}} \tag{8}$$

where α denotes the unscaled reward for each point. In this paper, it is assumed that α is a constant that represents the information gain of a point without scaling. ξ denotes the regression factor, and δ_{td} denotes the threshold distance. For example, if $\xi = 2$ and $\delta_{td} = 10$, the information gain given by 1 point 10 units away is equivalent to the information gain given by 2 points 20 units away, which is also equivalent to that given by 1000 points 100 units away.

3.4. Optimal Q-value Approximation by DQN

With the defined state, action, and reward, the DRL process can be further described. The expected accumulated discounted reward of policy π is defined as $\eta(\pi)$:

$$\eta(\pi) = \mathbb{E}[\sum_{t=0}^{T} \gamma^t r_t(s_t, a_t)] \tag{9}$$

where $\mathbb{E}[\cdot]$ is the expectation taken over $s_{t+1} \sim Pr(s_{t+1}|s_t, a_t)$ and $\gamma(0 \leq \gamma \leq 1)$ is the discount factor of the reward at different time steps.

The goal of the learning algorithm is to determine the optimal policy π^* by estimating the optimal Q-function, which is defined as:

$$Q^*(s,a) = \mathbb{E}[\eta(\pi^*)|s_t = s, a_t = a]. \tag{10}$$

To approximate the optimal function, we use DQN with experience replay [10].

4. SIMULATION RESULTS

We evaluated the DRL-based airborne LIDAR navigation optimization method by conducting simulation using AirSim [7]. AimSim is a simulator platform for AI research to ex-

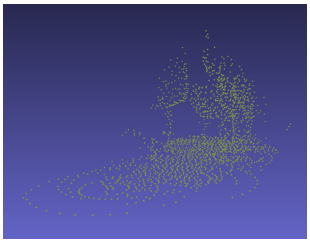

Fig. 4: Acquired point cloud during an episode of the early learning phase of the agent

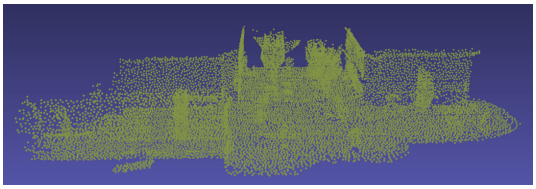

Fig. 5: Acquired point cloud during the agent training with high scale

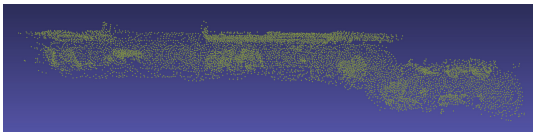

Fig. 6: Points collected without scaling

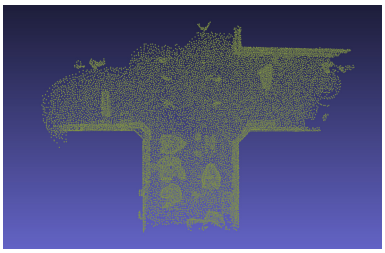

Fig. 7: Points collected with scaling

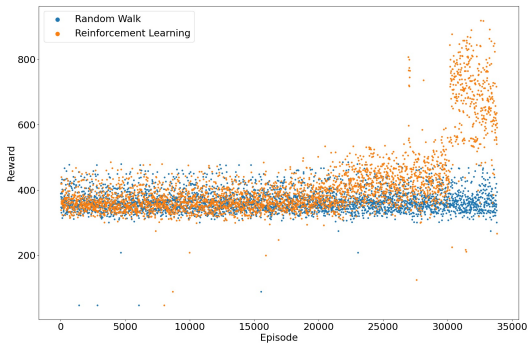

Fig. 8: Immediate rewards with different learning episode

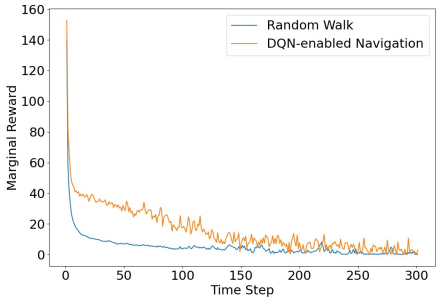

Fig. 9: Comparison of immediate rewards of random-walk and DQN-enabled navigation at different learning time steps

periment with deep learning, computer vision and reinforcement learning algorithms for autonomous vehicles such as drones and cars. AirSim provides APIs to retrieve data and control drones/vehicles in a platform-independent way. Table 1 shows the configuration of DQN.

As shown in Fig. 4, the agent can collect a small number of points during the earlier episodes due to colliding in the environment. When evaluating the agent after the training, the agent is able to avoid obstacles and collect plentiful LIDAR data of the environment, showing significant improvement. In contrast to Fig. 4 showing the point cloud acquired by the agent during an episode of the early learning phase, Fig. 5 shows the acquired point cloud during the agent training with high scale.

Before the scaled information gain function is employed, the reward function will count each newly added point as equal and valid information. As shown in Fig. 6, the agent moves in one direction and ensures that there are objects around it when possible to maximize the reward. The agent started from the right side of the figure and moved left until there is no tree around it, then it moved up to detect the wall, then continued to move left for the rest of the episode. However, we want the agent to provide a holistic view around the origin. Therefore, the scaled information gain function is introduced to disincentivize the agent from moving in only one direction. As shown in Fig. 7, the agent navigates around the origin and acquires a more comprehensive environment than that obtained in Fig. 6.

Figure 8 shows the immediate reward obtained with the proposed reinforcement learning method and random walk, respectively, at different learning episode. Figure 9 shows the comparison of the immediate rewards of random-walk and the DQN-enabled navigation with respect to learning steps. It can be observed that the DRL method has achieved better result than random walk.

5. CONCLUSION

We present a framework for adaptive acquisition of airborne LIDAR point cloud based on deep reinforcement learning (DRL). Novel LIDAR point cloud processing methods are developed to derive the state space, action space, and reward function of the DRL model. The model is trained and tested in a flexible virtual environment by using autonomous vehicles simulator AirSim. Extensive simulation demonstrates the efficiency of the proposed framework. As part of future work, we will evaluate the framework through field experiment.

6. REFERENCES

[1] D. Bastos, P. P. Monteiro, A. S. Oliveira, and M. V. Drummond, "An overview of lidar requirements and techniques for autonomous driving," in *Telecoms Conference (ConfTELE)*, 2021.

[2] S. Gargoum and K. El-Basyouny, "Automated extraction of road features using lidar data: A review of lidar applications in transportation," in *ICTIS*, 2017.

[3] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007.

[4] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[5] P. Cai, S. Wang, H. Wang, and M. Liu, "Carl-lead: Lidar-based end-to-end autonomous driving with contrastive deep reinforcement learning," *arXiv preprint arXiv:2109.08473*, 2021.

[6] J. Liang, U. Patel, A. J. Sathyamoorthy, and D. Manocha, "Realtime collision avoidance for mobile robots in dense crowds using implicit multi-sensor fusion and deep reinforcement learning," in *Proc. of the 19th AAMAS*, 2020.

[7] "AirSim," [Online] Available: https://microsoft.github.io/AirSim/ [Accessed on April 30, 2022].

[8] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020.

[9] J. Gall, "Use of cylindrical projections for geographical, astronomical, and scientific purposes," *Scottish Geographical Magazine*, vol. 1, pp. 119–123, 1885.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *NIPS Deep Learning Workshop*, 2013.