Machine Learning Point Cloud Enhancement for Self-Driving Systems

Nicolás Salomón
Fundación Fulgor
Córdoba, Argentina
nicolassalomon96@gmail.com

Claudio A. Delrieux
Departamento de Ingeniería Eléctrica y Computadoras, Universidad Nacional del Sur
Bahía Blanca, Argentina
cad@uns.edu.ar

Damián A. Morero
Laboratorio de Comunicaciones Digitales, Universidad Nacional de Córdoba
Córdoba, Argentina
dmorero@gmail.com

Leandro E. Borgnino
Departamento de Electrónica, Universidad Nacional de Córdoba
Córdoba, Argentina
leandro.borgnino@unc.edu.ar

Abstract

Autonomous driving, once a distant vision, is now reshaping the automotive industry. However, LiDAR sensors-crucial for perception-remain costly and computationally demanding. This work explores the use of advanced interpolation techniques to enhance lower-cost LiDAR sensors while ensuring computational efficiency, aiming to match the performance of high-end 64-channel LiDARs.

Our approach is based on three key aspects: (1) the analysis and representation of LiDAR data, (2) the implementation of an interpolation technique using 1D convolutional layers and fully connected layers to process sliding window data, and (3) a comparative evaluation against state-of-the-art interpolation methods, leveraging PR curves to assess performance in object detection tasks on point clouds. Additionally, we present a preliminary analysis of power consumption and discuss potential hardware implementations.

The proposed interpolation method achieved detection and classification improvements ranging from 1.92% to 30.98%, depending on the object and detection type (3D or bird's eye view). Importantly, it also reduced inference times compared to alternative techniques, demonstrating its computational efficiency. These results highlight the potential of our approach to enable cost-effective, high-performance autonomous driving systems.

Keywords: LiDAR, Artificial Intelligence, point clouds, range images, image interpolation, object detection.

1 Introduction

Autonomous driving systems are one of the most relevant research topics nowadays, both in the automotive and the Computer Science fields. Their development offers a wide variety of improvements, ranging from

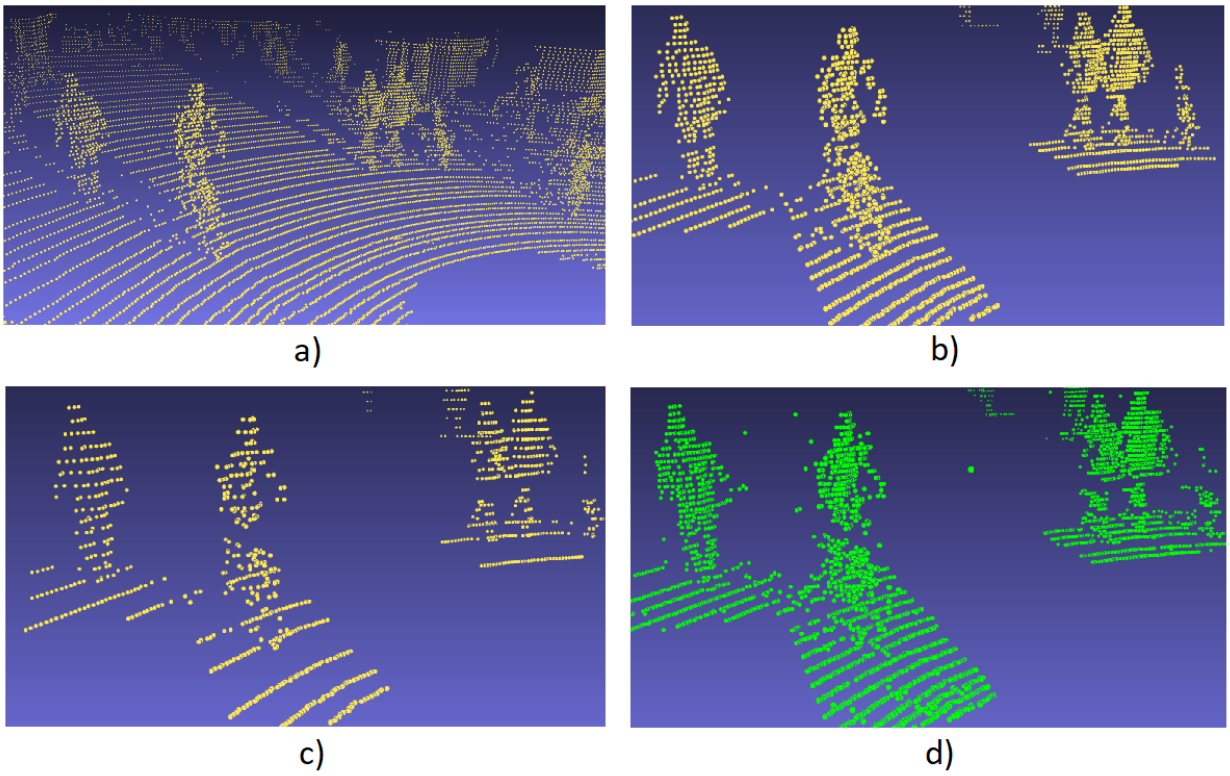


Figure 1: Interpolated point clouds. a) and b) Original (64-channel). c) Original (32-channel). d) Interpolated 64-channel.

general safety, by limiting human error in driving, to the optimization of processes and resources by reducing time and increasing performance in all tasks associated with transportation.

None of this would be possible without the presence of multiple sensors, responsible for mapping and generating a genuine representation of the environment surrounding the vehicle. LiDAR (*Light Detection And Ranging*) is one of the most reliable and highest resolution sensors for most navigation tasks, and one of the most expensive too. In particular, 3D LiDAR can sense within large volumes, with high resolution for object detection such as cars, pedestrians or cyclists, generating a three-dimensional point cloud of the surrounding space (see Fig. 1 a). However, this is the case only for high resolution sensors, associated with LiDARs of 64-channels or more, which generate higher density point clouds and allow capturing more specific details, at the cost of a notably higher price (the higher number of channels, the more expensive the sensor will be). This last factor currently limits its use for commercial autonomous vehicles, making low-channel LiDAR information processing a very attractive research and improvement topic.

On the other hand, the current rise of Machine Learning (ML) and Deep Learning (DL) algorithms applied to data processing and computer vision is expanding the scope in several fields. The synergy between the data obtained by LiDAR sensors, together with current data analysis techniques and Artificial Intelligence (AI), could result in solving the cost problem raised above. Furthermore, the development of current graphics engines (Unreal Engine, Unity, among others) allow the creation of practical autonomous vehicle simulators with numerous sensors, considerably reducing the cost associated with the acquisition of data for the study and development of systems that require this information.

In this work we contribute to the development of more accurate and efficient tools for capturing and analyzing 3D data, without leading to a significant increase in costs. The main idea is to interpolate data from a 32-channel LiDAR to resemble the behavior of a 64-channel sensor, oriented to the autonomous driving related contexts. Using AI techniques together with real and simulated datasets, we were able to train an intelligent upsampling model able to generate results similar to those presented in Fig. 1. This work paves the way to explore the potential of AI towards the combination of current technologies and AI methods, redefining the capabilities of remote sensing, and opening new possibilities for research and its practical applications.

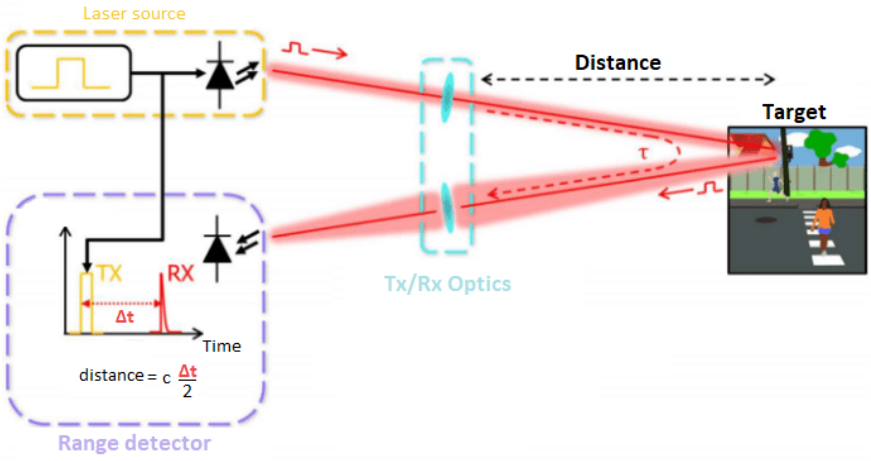


Figure 2: Operation of a time-of-flight LiDAR

2 Preliminary studies

This section provides a brief overview on the main concepts related to the proposed interpolation technique, such as point clouds, range images, and 3D object detection networks. Understanding the foundational concepts and methodologies within these areas is crucial for developing robust and efficient algorithms for analyzing and interpreting 3D spatial data.

Firstly, data acquired through LiDAR sensors are referred to as point clouds. These point clouds represent a collection of thousands or even hundreds of thousands of points, each with specific coordinates in a 3D space and an associated intensity value. They serve as fundamental data structures for capturing detailed geometric information about physical environments. The most widespread type of LiDAR (Light Detection and Ranging), called *Time of Flight (ToF) LiDAR*, operates by emitting laser pulses towards the environment and measuring the time it takes for the pulses to return after reflecting off objects. This process is known as time-of-flight measurement and it's shown in Fig 2. The sensor then uses this information to calculate the distance between the sensor and the objects in its vicinity, allowing it to map out the 3D structure of the surroundings. The distance R to the target is calculated as:

$$R = \frac{v}{2} * t_{oF} = \frac{c}{2} * t_{oF}, \tag{1}$$

where v is the pulse speed in the medium and t_{oF} is the round-trip travel time. It is usually assumed that $v = c$, where c is the speed of light in a vacuum, since the refractive index of air is very close to 1.

However, this representation tends to be computationally expensive due to the sparseness of points within the point cloud. To address this issue, range images are generated by transforming the data space from Cartesian to cylindrical coordinates, effectively transitioning from the three-dimensional plane (point cloud) to a two-dimensional representation.

Considering a LiDAR with n channels, each of the n laser beams will have a specific elevation angle θ^l where $l \in 1, 2, ..., n$. Similarly, at each time instant t, each laser beam has a unique azimuth angle ϕ^t. Therefore, each point in a point cloud is defined by a unique pair (θ^l, ϕ^t) along with a value r, which represents the distance between the sensor and the object where the laser beam hit [1]. Based on these three components, the 3D coordinates can be calculated through Eqs. (2)-(4).

$$x = r * \cos(\theta^l) * \sin(\phi^t) \tag{2}$$
$$y = r * \cos(\theta^l) * \cos(\phi^t) \tag{3}$$
$$z = r * \sin(\theta^l) \tag{4}$$

On the contrary, the projections from the cartesian to cylindrical plane, in order to return to the 2D space, can be made by using Eqs. (5)-(7).

$$r = \sqrt{x^2 + y^2 + z^2} \tag{5}$$
$$\theta = \arctan(y/x) \tag{6}$$
$$\phi = \arcsin(z/r) \tag{7}$$

In this process, distance and intensity values are encoded for each point in the cloud. This transformation is illustrated in Fig. 3.

Secondly, considering that the final goal of an autonomous driving system focuses on making decisions based on the environment it is located in, there are multiple architectures proposed for object detection, among which the most used are *Complex-YOLO* [2] and *Pointpillars* [3].

On one side, *Complex-YOLO* is an advanced adaptation of the popular YOLO (*You Only Look Once*) object detection algorithm, designed specifically to address challenges in accurately detecting objects in images. It maintains the general architecture of YOLO, which consists of a single convolutional neural network that simultaneously predicts bounding boxes and classes in a single pass over the input image, but adds complex properties that represent both the magnitude and phase of the features extracted from the image. This allows capturing spatial and structural information about objects compared to traditional methods that only consider the magnitude of features.

On the other side, *Pointpillars* is an architecture developed specifically for the context of autonomous vehicles and 3D perception systems. It focuses on computational efficiency and the ability to handle sparse and high-resolution point clouds, commonly arising in LiDAR sensors. This technique encodes the point cloud into vertical columns or pillars that can be used in any 2D convolutional detection model, in conjunction with an SSD type network (*Single Shot Detector*) to detect objects at different scales.

The latter architecture exhibits greater robustness against variations in point density and a better ability to detect small objects as compared with other detection networks, such as *Complex-YOLO*. Furthermore, in terms of computational efficiency, it is capable of operating in real time and tends to be more efficient due to its optimized design for point cloud processing, making it the most convenient choice for many three-dimensional object detection applications.

3 Related work

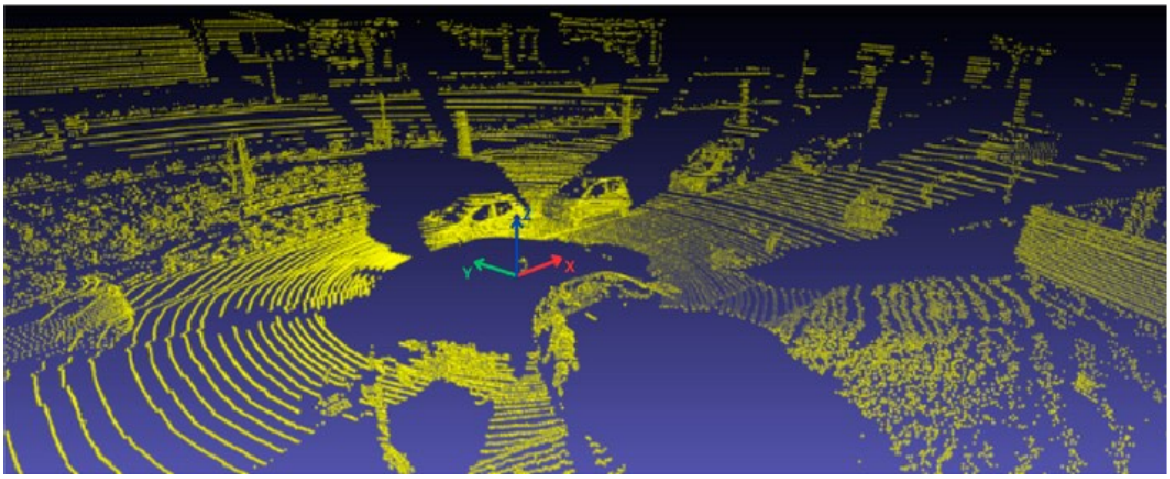
The development of autonomous driving systems, and the subsequent processing of data from LiDAR sensors, have been an active research area in recent years, with a particular focus on surrounding environment perception. Increasing the resolution of a point cloud using AI techniques has emerged as a crucial strategy to improve the accuracy and robustness of these systems.

There are multiple techniques proposed to achieve this objective. Some authors (see [4–7] for instance) present point cloud processing techniques using DL in 3D space. These techniques tend to add features to each point through multi-level feature extraction based on numerous convolutional layers with custom loss functions. However, this type of processing can often be too computationally expensive, due to the characteristic point cloud data dispersion. This is the reason why in works like [5] and [6] only test their algorithms performing the upsampling on single objects in indoor environments, and in [4] the authors only use patches of data from an freely available dataset, but not the entire point cloud.

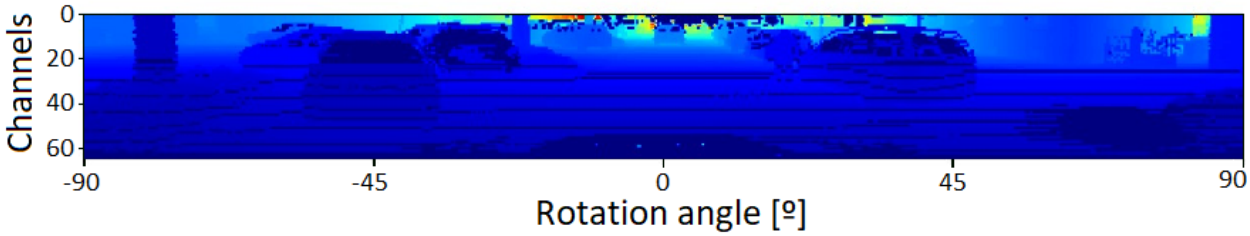
On the other hand, there are several proposals based on interpolation techniques in 2D through images that represent the three-dimensional space in a compressed way (range images), as seen in [8–11]. The use of this representation also enables to take advantage of the current advances in image Super-Resolution (SR) methods, which are usually based on trainable autoencoder architectures with convolutional and deconvolutional layers that allow reconstructing high resolution images based on low resolution ones [12,13].

In [8] the authors propose an encoder-decoder architecture with skip-connections for range image SR, along with Monte-Carlo dropout in order to reduce noise in the interpolated point cloud. The authors use only simulated data for the purpose of training their approach, but they don't share the results in object detection tasks. However, many of these interpolation techniques based on neural networks tend to oversmooth object edges, due to regularization effects or error functions used during training [14], leading to an overall loss of geometric definition and therefore of usefulness for object recognition. Furthermore, depending on the depth of these networks, they may not reach the standards in inference times to be applied in real autonomous navigation systems.

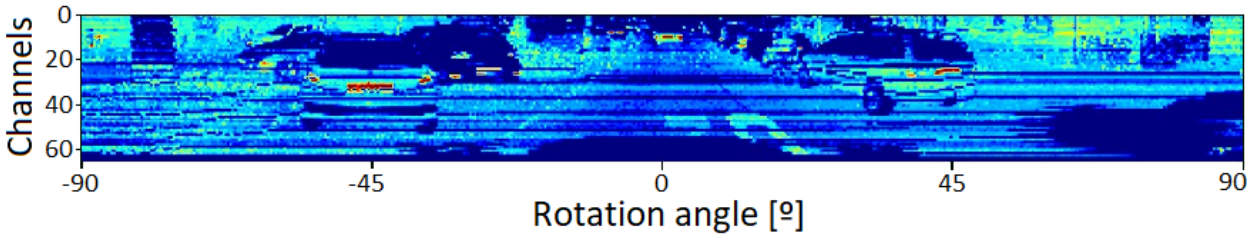
Changing the focus, [9] proposes to change neural networks to heavy interpolations using a custom interpolant function and converting the range image into a higher-dimension 2D range image with 4-Ch of the Cartesian coordinates (x, y, z) and intensity values. Finally, all the mentioned proposals use conventional interpolation techniques, such as bilinear or bicubic interpolation [15], as proof of concept and contrast to evaluate their results. Most of these simple techniques can't describe complex shapes and generate several noisy points around the object in the point cloud.



((a)) Point cloud.



((b)) Range distance image.



((c)) Range intensity image.

Figure 3: LiDAR data representation. Range images field of view was restricted to [-90°, +90°] only for visualization purposes.

4 A novel upsampling technique

This section details all the processes involved in the design and implementation of the proposed system, along with a comprehensive explanation of the methods used in each task to increase the resolution in the point clouds using AI techniques. For access to the source code, please contact nicolas.salomon@uns.edu.ar.

The problem raised regarding the costs of high-resolution LiDAR sensors is addressed via the 2D representation of the data and its associated range images, transforming the 3D interpolation problem into a 2D one. However, the nature of these images must be considered, since they contain information related to distances or intensities and their spatial representation, rather than visual information (illumination, exposure, balance, among others), as happens in images captured by a conventional camera. The workflow to be followed by the proposed system is seen in Fig. 4 and can be divided into three fundamental steps:

- Data acquisition and dataset assembly.
- Interpolation architecture implementation.
- Training.

4.1 Data acquisition and dataset assembly

In the field of AI, the dataset constitutes one of the fundamental pillars on which algorithms are built and refined, becoming an essential part for the success of any system. Particularly, in the autonomous driving

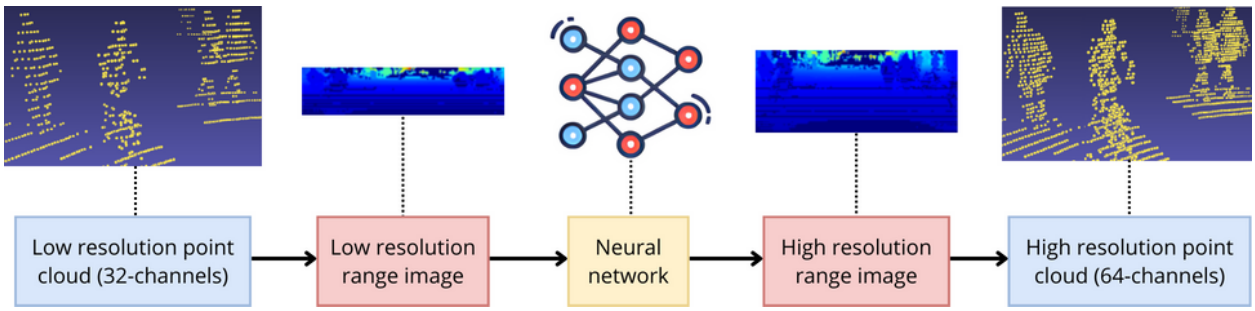


Figure 4: Main system architecture.

environment there is a high difficulty in generating a dataset that contains data from LiDAR sensors, due to its complexity and high cost. Therefore, we decided to address this difficulty from two different points: employ the KITTI (*Karlsruhe Institute of Technology and Toyota Technological Institute*) 3D object detection dataset [16] (see Fig. 6(a)) and, on the other hand, a synthetic dataset was generated, using the autonomous vehicle simulator CARLA (*Car Learning to Act*) [17] (see Fig. 6(b)) configured with the same parameters and characteristics than the LiDAR *Velodyne HDL-64E* used during the KITTI dataset data collection. CARLA allows the generation of a highly realistic 3D simulation environment with urban and suburban scenarios, fully customizable by adjusting parameters of the scene, vehicles and different virtual sensors, such as cameras, radars and LiDARs. It facilitates the simulation of the environment perception, modeling the dynamics of the vehicle in a realistic way. In contrast, the LiDAR computational model presents a simple development that only considers atmospheric attenuations, without taking into account important parameters such as the composition, surface and reflectivity of the objects present in the scene.

An important characteristic of the KITTI dataset is the distribution of labeled objects, which is inherently influenced by the nature of the environments where the data was collected (highways and urban streets). As shown in Figure 5, there is a significantly higher number of labeled objects belonging to the "Car" class, while other categories, such as "Pedestrian" and "Cyclist," are considerably less frequent.

This imbalance is a direct consequence of the data acquisition process in autonomous navigation applications, where vehicle-mounted sensors primarily capture a vehicular environment with fewer instances of pedestrians and cyclists. In this work, we used CARLA to generate a simplified scene, allowing an initial exploration of LiDAR point cloud upsampling. However, specifically increasing pedestrian and cyclist data requires a more detailed analysis, particularly regarding the physical models and behavioral patterns simulated within CARLA. While this represents an opportunity for future research, the present study focuses on developing

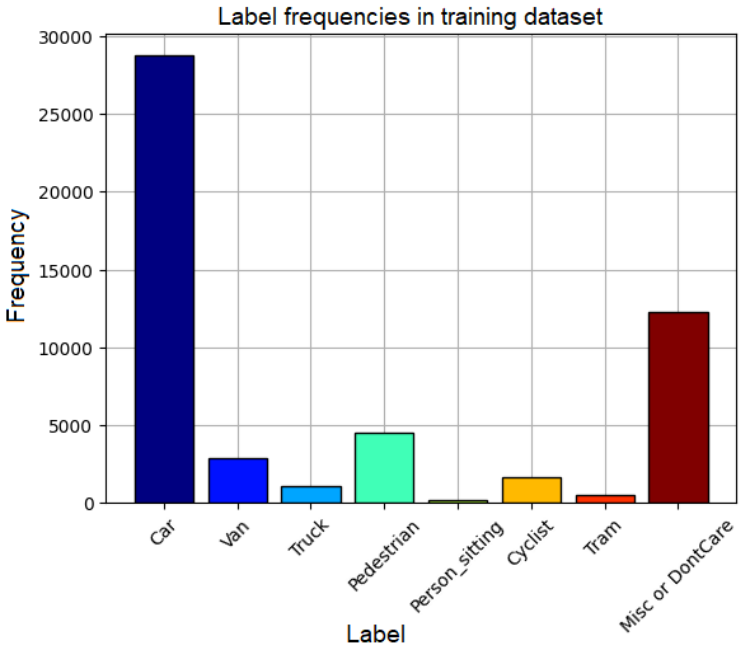
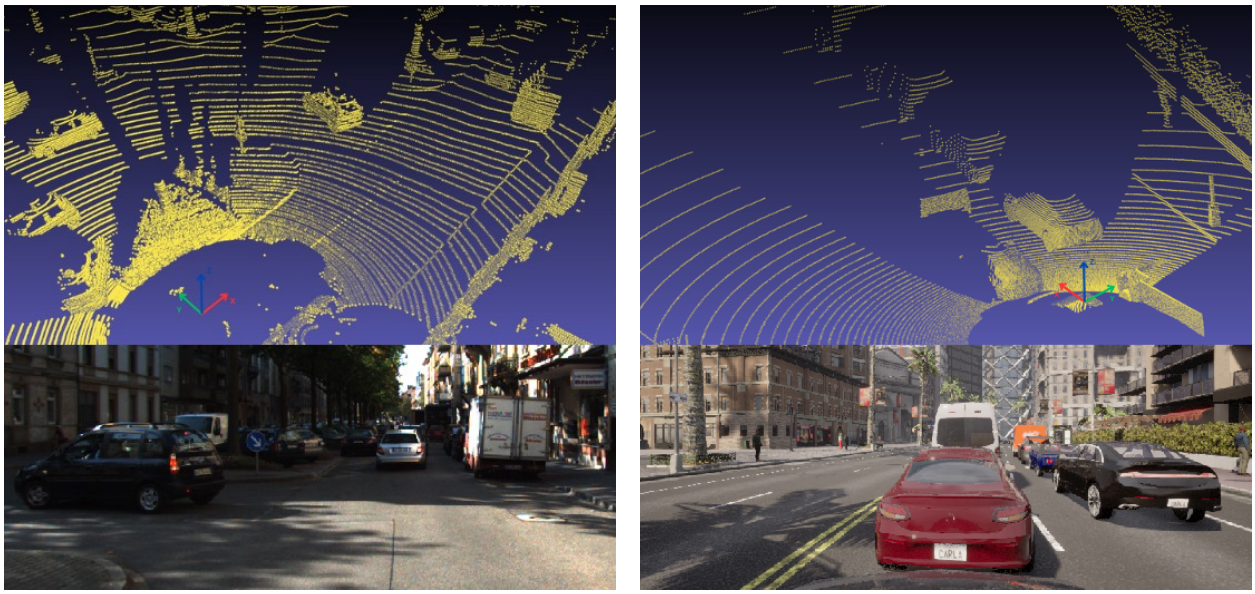


Figure 5: Label frequencies in the KITTI dataset.

and validating upsampling techniques, leaving dataset balancing as a potential improvement for future work.

Once the synthetic data was generated, a new database was created composed by the range images associated with both datasets, KITTI and CARLA. We achieved this by using Eqs. (5) to (7), which performs the coordinate transformation from cartesian to cylindical coordinates.

The input to our interpolation system is assumed to be a point cloud, with $n = H \times W$ total points with the form $p = \{x, y, z, i\}$, where H and W represent the height and width of the range image associated with the point cloud, x, y, z are its cartesian coordinates and i represents the light intensity received by the sensor for said point. In our study, range images of $H = 64$ and $W = 2048$ associated with the high-resolution data were used, and images of $H = 32$ and $W = 2048$ were used for the low-resolution data. It is important to highlight that the interpolation will be carried out on the vertical axis (height of the image) since this parameter is related to the total number of channels of the point cloud associated with the image.



((a)) KITTI point cloud (up) and RGB image (down). ((b)) CARLA point cloud (up) and RGB image (down).

Figure 6: Comparison between KITTI and CARLA point clouds and RGB images.

4.2 Interpolation architecture implementation

Aiming to overcome the techniques proposed in [4, 8, 9] for object detection on the KITTI dataset, we propose an interpolation method based on 1D convolutions, fully connected layers, and a sliding window that moves through the image, obtaining the values of neighboring pixels to the one to be interpolated. This method seeks to combine established techniques with new strategies to improve accuracy and computational efficiency.

Our proposal addresses the challenges of interpolating a point cloud by leveraging the ability of 1D convolutions to capture local and global patterns in the data, as well as the power of fully connected layers to integrate contextual information across the neural network. A unique feature of our method is the use of a sliding window, which moves over the image to capture local information and enables adaptive interpolation based on local characteristics of the scene. This strategy improves the model's ability to generalize, and produces accurate results across a variety of conditions and data configurations. In addition, it has a reduced computational cost as it is a network with a fairly simple architecture.

The proposed topology can be seen in Fig. 7, which receives as input a low-resolution range image (32-channels). Then a sliding window of size $[2 \times 3]$ (2 rows by 3 columns) is used, which runs through the low-resolution image and in each step obtains the 6 pixels closest to the pixel to be interpolated, hence the dimension of the window. Subsequently, these 6 values serve as input to the network composed of 2 1-dimensional convolutional layers, followed by two fully connected layers with Leaky_ReLU and ReLU activation functions respectively.

At that point, it also incorporates the minimum value of the 6 values extracted by the sliding window, with the idea of helping the network to locate objects in the scene according to their distance from the sensor. At the output of our network, the interpolated value of the central pixel of the window is obtained. This operation is performed for each pixel to be interpolated.

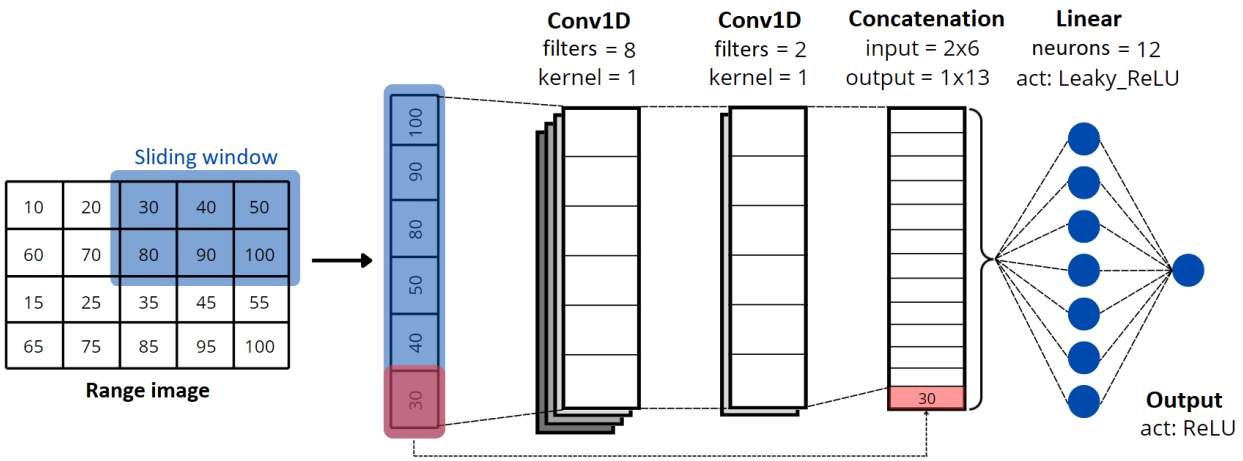


Figure 7: Proposed network architecture.

4.2.1 ***Loss function***

In the context of range image interpolation, it is essential to define an error function that allows evaluating the quality of the results and guiding the optimization process to improve the accuracy of the model. In this work, we employ two different metrics to address the two types of range images, necessary for a subsequent object detection network in point clouds: the distance range image, related to the spatial distance between the points in the cloud, and the intensity range image, related to the intensity of the signal received by the sensor.

To quantify the spatial fidelity of the interpolation in the distance range image, a well-known metric in point cloud comparison, the Chamfer Distance, is used. This metric calculates the discrepancy between points in a point cloud generated by the interpolation and the reference point cloud, by adding the least square distances between each point in one set and its closest point in the other set, such as as expressed in (8).

$$d_{\text{Chamfer}}(P,Q) = \sum_{p \in P} \min_{q \in Q} \|p - q\|_2^2 + \sum_{q \in Q} \min_{p \in P} \|p - q\|_2^2 \tag{8}$$

The Chamfer Distance provides a robust measure of the spatial precision of the interpolation, penalizing inappropriate offsets and geometric distortion. By minimizing this value, we seek to improve the spatial alignment between the interpolated point cloud and the reference one, resulting in a more precise and faithful interpolation to the original structure of the scene.

On the other side, since Chamfer Distance tends to quantify spatial precision, it would be inadequate as a loss function to compare intensity range images. Therefore, to evaluate the error of the network that uses this type of images, it was decided to use the Mean Absolute Error (*MAE*), which provides a measure of the precision in the reproduction of pixel intensities, without considering their spatial position. By minimizing the MAE, we seek to improve the accuracy in estimating range image intensity values.

4.2.2 ***Optimizer***

The choice of an appropriate optimization function is extremely relevant during the training of any ML architecture. In this work, the ADAM (*Adaptive Moment Estimation*) [18] algorithm was used as an optimization function, due to its ability to effectively adapt to complex optimization scenarios with multiple local minima, in conjunction with a learning rate equal to 10^{-2}.

4.2.3 ***Training***

The training process was carried out for 200 epochs in the case of the distance range images interpolating network, and 100 epochs for the intensity range images interpolating network. A batch size of 100 images was used, as this was the maximum allowed by the available hardware. The experiments were conducted on a system equipped with an NVIDIA RTX 3060 GPU, an Intel Core i5-11400F CPU, and 16GB of RAM, running Windows 10. Two different training sessions were carried out, varying the datasets used as follows:

i. **Dataset 1:** Training and validation sets composed of range images (distance and intensity) belonging only to the KITTI dataset.

ii. **Dataset 2:** Training and validation sets composed of range (distance and intensity) images from KITTI and CARLA.

Several topologies were tested aiming to achieve the lower interpolation error, varying the number of convolutional and linear layers from the final proposed network (see Fig. 7), and considering the Chamfer Distance as the loss function. The results obtained from these experiments are as follows:

- Decreasing the number of filters of the first convolutional layer to 4: achieving an error of 0.37.
- Adding a third 1-dimensional convolutional layer and changing the number of filters for each layer to 10, 8 and 6 respectively: achieving an error of 0.34.
- Adding a second hidden linear layer with 6 neurons: achieving an error of 0.44.

As shown in Fig. 8, the final proposed architecture improves on the others in terms of error function, achieving a Chamfer Distance error of 0.28.

5 Results

The objective of this section will be to show the results obtained and perform a contrast with other conventional interpolation methods, such as bilinear or bicubic interpolation, and the state of the art ([8,9]) against actual scenes point clouds, demonstrating the advantages of our proposal compared to the existing ones. With this goal in mind, the proposed interpolating network was trained according to the parameters detailed in Sec. 4.2. *Pointpillars* was used as the object detection network in point clouds to contrast and quantify the results of the different interpolation techniques. The parameters of this network were set as established by their authors, in this way the only variable that could reflect a difference between the results obtained was the interpolation network applied.

5.1 Interpolation network

As specified in 4.2, the interpolation networks were trained by varying the input datasets, in order to analyze the influence of using actual data in conjunction with synthetic data. The training loss plots are shown in Figs. 8 and 9. It is observed how the losses, in both the training and validation stages, begin with a relatively high value and decrease as the training epochs progress. This behavior shows us that the networks manage to learn patterns in the images, allowing them to interpolate more precisely, up to the point at which the error stabilizes, demonstrating that the network reached a minimum in its loss function. The same behavior is observed both for the network trained using only the KITTI dataset, and the one trained using a combination of both datasets, KITTI and CARLA.

It can be observed that the model trained exclusively on KITTI exhibits a faster initial convergence compared to the model trained on the KITTI + CARLA dataset. However, both configurations eventually reach similar final error values after 200 epochs.

Despite the seemingly comparable convergence, further experiments showed that using the interpolated point clouds from the KITTI + CARLA model as input for a downstream detection network, such as PointPillars, resulted in better overall performance (see Tables 1 and 2). This suggests that the additional diversity introduced by CARLA data, even if it does not significantly impact the final Chamfer Distance Error, provides richer features that improve object detection. The inclusion of synthetic data may help regularize the model, making it more robust to variations in real-world scenarios.

In Fig. 10 we show the interpolations carried out using the network trained only with the KITTI dataset (Fig. 10(a)) and with both the KITTI and CARLA datasets (Fig. 10(b)), in contrast to the original point cloud (Fig. 10(c)).

5.2 Object detection network

Once the interpolator network was trained, the *Pointpillars* detection network was trained for the following datasets (all range images have dimensions of 64×2048, unless specified otherwise):

1. Original KITTI dataset.
2. Original KITTI dataset with 32×2048 decimated range images.

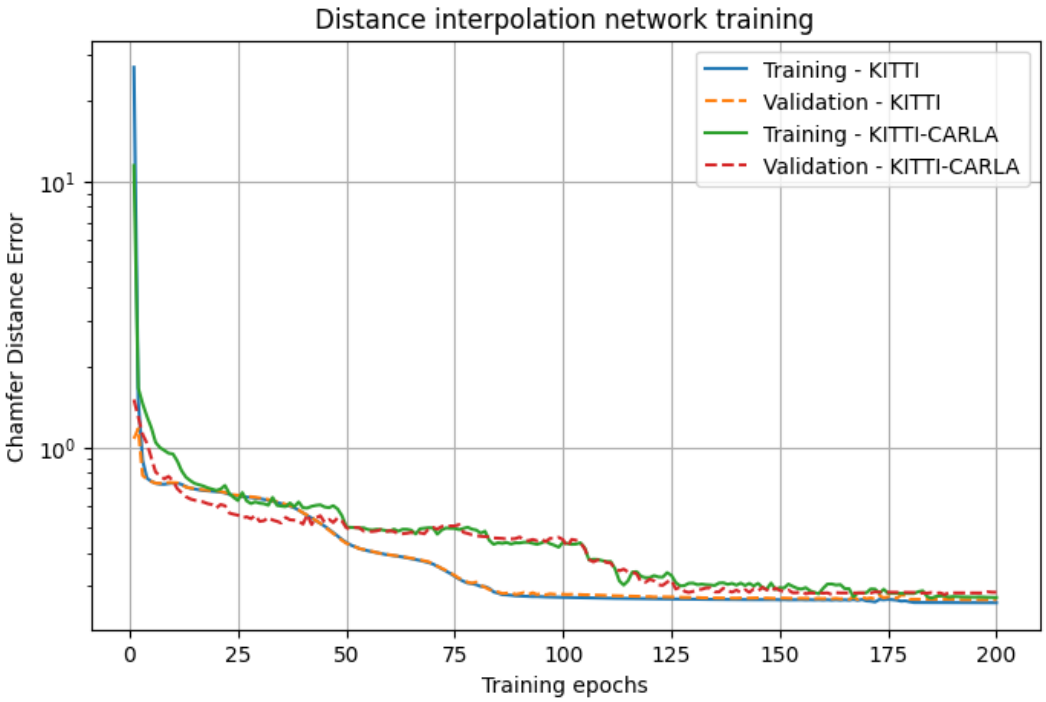


Figure 8: Loss function through epochs along distance range image interpolation networks training.

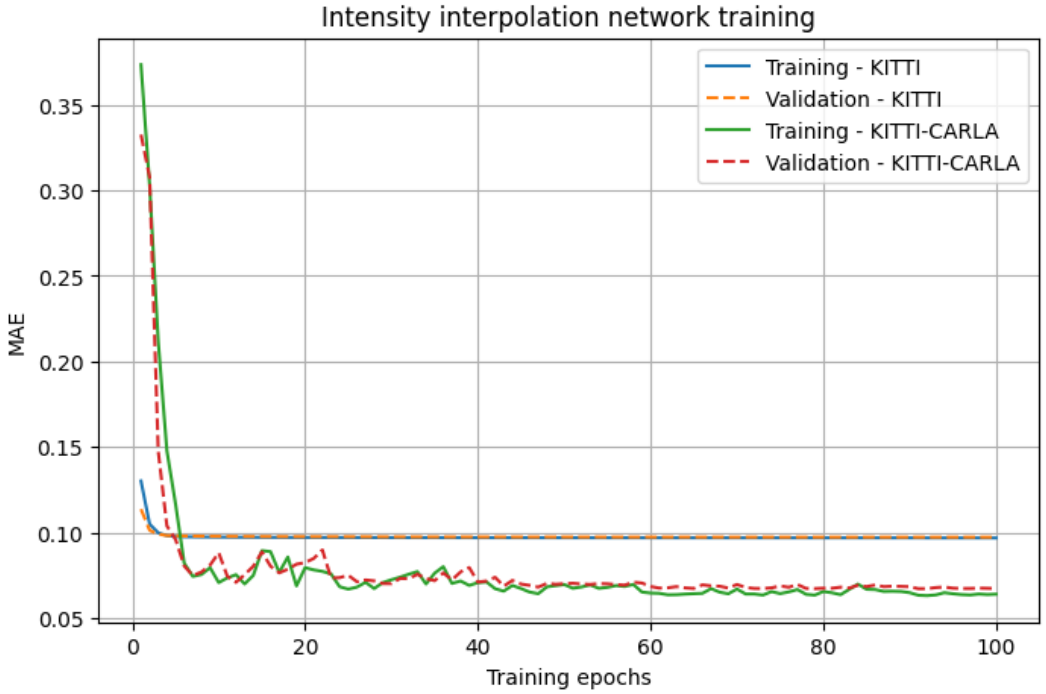
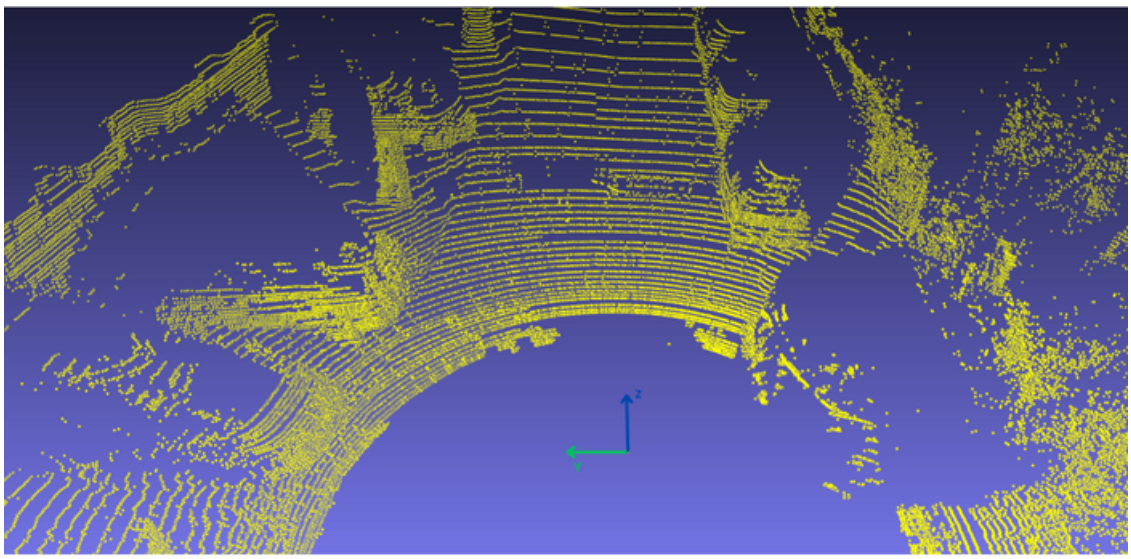
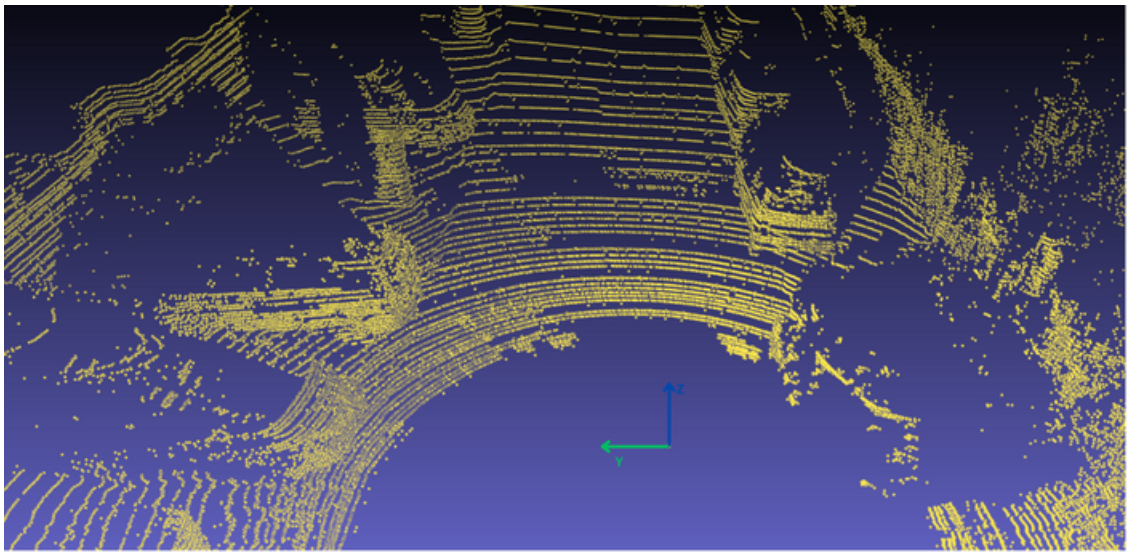


Figure 9: Loss function through epochs along intensity range image interpolation networks training.

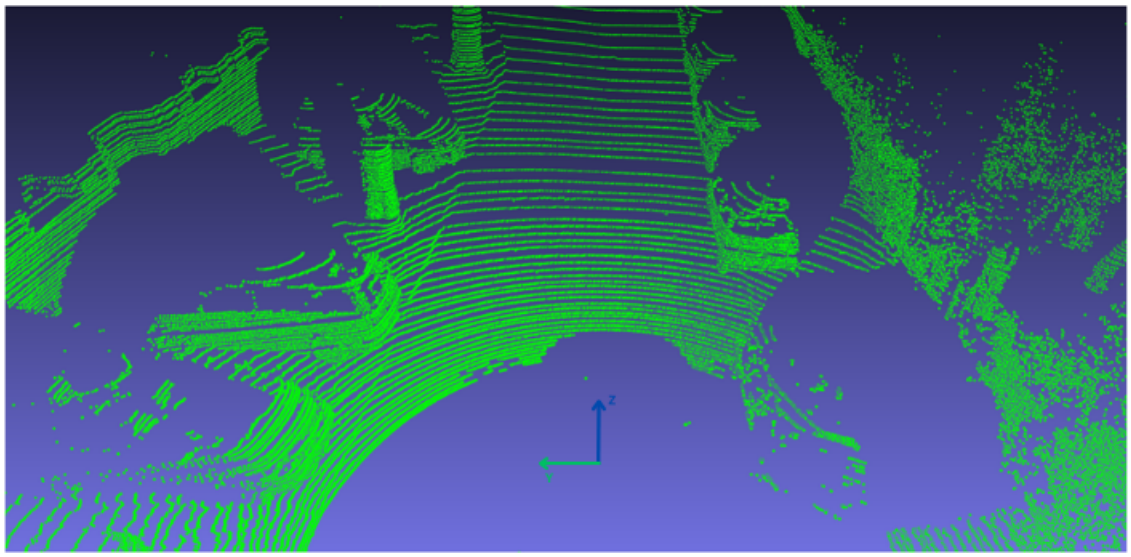
3. Dataset generated using bilinear interpolation.
4. Dataset generated using bicubic interpolation.
5. Dataset generated with the interpolation network proposed in [8].
6. Dataset generated with the interpolation network proposed in [9].
7. Dataset generated with the proposed interpolation network trained only with the KITTI dataset (for both distance and intensity range images).



((a)) Interpolated point cloud with trained network (KITTI).



((b)) Interpolated point cloud with trained network (KITTI-CARLA).



((c)) Ground Truth point cloud.

Figure 10: Comparison between original and interpolated point clouds from the KITTI test set.

8. Dataset generated with the proposed interpolation network trained with the KITTI and CARLA dataset (both for distance and intensity range images).

9. Dataset generated with the proposed distance interpolation network trained with the KITTI and CARLA dataset, and the proposed intensity interpolation network trained only with the KITTI dataset.

The last three sets generated vary in their interpolation networks used, since we aim to demonstrate that the computational modeling of the LiDAR in the CARLA simulator still does not reliably consider the real characteristics of the intensity channel of a real sensor.

The *AP* (*Average Precision*) was used as the metric to analyze the results, which were generated according to the KITTI evaluation protocol for 3D object detection (for more details, see [19]). Table 1 shows the performance for 3D object detection, while table 2 corresponds to the results for detection in bird's eye view. It is important to clarify that in [9] only the metrics obtained by the authors for the detection of objects in 3 dimensions are provided. With the intention of having values referring to the bird's eye view detection to contrast the results obtained with the architecture proposed in this work, a network based on [9] was trained and the bird's eye view detection values were taken to complete the Table 2.

On the other hand, the section "Proposed: KITTI + CARLA (distance)" of the tables 1 and 2 refers to the training carried out using the dataset generated with the proposed distance interpolation network, trained with the KITTI and CARLA dataset, and the proposed intensity interpolation network, trained only with the KITTI dataset.

Additionally, table 3 provides the inference times for the compared networks, using the same setup shown in 4.2.3. It should be noted that the authors in [9] do not declare the inference time of their proposal, so the inference time of the implementation carried out to obtain the detection values for bird's eye view was considered. In Fig. 11 and 12 the results obtained by interpolating a 32-channel point cloud using the proposed network are shown. This output serves as input to the pretrained object detection model *Pointpillars*. Additionally, the ground truth values are provided for the same scenes.

The Fig. 13 shows the results obtained by calculating Precision and Recall for each class (Car, Cyclist, and Pedestrian) and for each type of detection (3D and bird's-eye view).

5.3 Hardware Implementation Analysis

The goal is to design a neural network suitable for low-power hardware, such as the NVIDIA Jetson Nano or an FPGA, enabling efficient and scalable embedded applications. The Jetson Nano offers rapid and flexible development, while FPGAs provide real-time inference with lower power consumption, making them ideal for production environments.

Key points include:

- Network Size and Input Image Dimensions: The network has 215 parameters, requiring only 860 bytes of memory.
  Input tensors for 32×2048 range images of dimensions require 1.52 MB per image.

Table 1: Comparison table of results training *Pointpillars* for different datasets and using *AP* as an evaluation metric for 3D object detection.

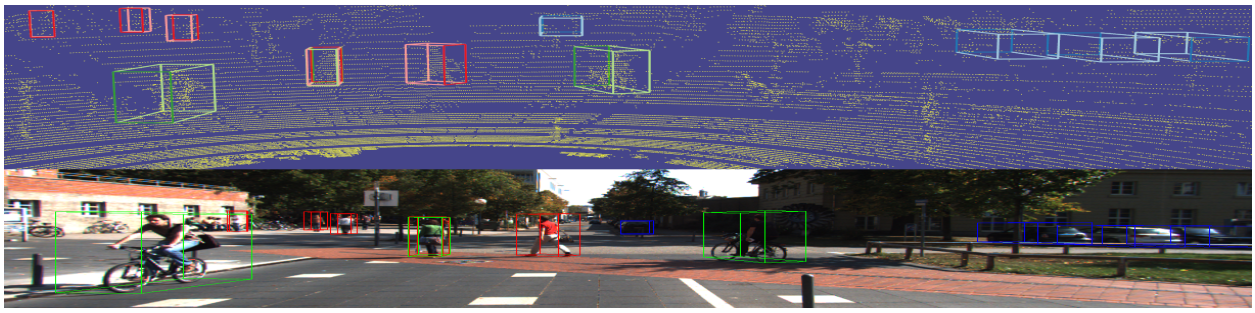
Class	**Reduced (32-channels)**			**Bilineal**			**Bicubic**		
	Easy	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
Car	69.41	55.13	54.23	71.70	53.82	48.31	47.73	35.42	31.88
Cyclist	65.29	45.34	42.26	64.13	40.62	37.79	32.60	20.68	18.77
Pedestrian	45.78	42.53	38.59	49.09	41.36	35.92	36.34	30.32	26.66
Class	**Shan, 2020 [8]**			**You, 2022 [9]**			**Proposed: KITTI**		
	Easy	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
Car	72.95	56.02	52.44	75.06	57.72	54.19	77.00	61.76	57.12
Cyclist	59.42	38.55	36.23	64.95	43.19	39.95	67.67	47.60	45.25
Pedestrian	47.09	40.11	35.51	39.96	35.18	31.71	50.67	43.94	39.98
Class	**Proposed: KITTI + CARLA**			**Proposed: KITTI + CARLA (distance)**			**Ground Truth (64-channels)**		
	Easy	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**	**Easy**	**Medium**	**Hard**
Car	77.39	64.21	58.24	**77.61**	**65.05**	**58.37**	84.36	72.59	67.23
Cyclist	70.30	48.69	45.92	**71.93**	**49.54**	**46.65**	79.43	61.14	56.88
Pedestrian	48.80	43.38	38.98	**52.58**	**45.90**	**41.51**	53.08	45.96	41.79

Table 2: Comparison table of results training *Pointpillars* for different datasets and using AP as an evaluation metric for bird's eye view object detection.

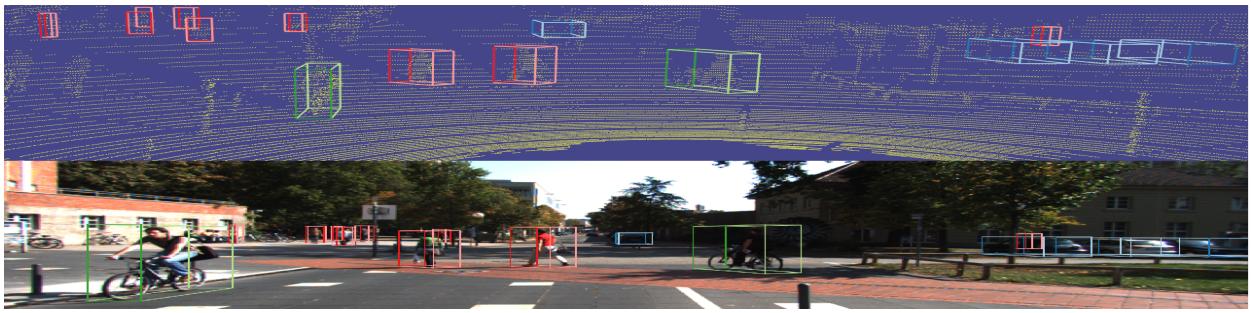
Class	Reduced (32-channels)			Bilineal			Bicubic		
	Easy	Medium	Hard	Easy	Medium	Hard	Easy	Medium	Hard
Car	86.66	76.70	76.05	87.41	69.08	64.61	73.32	55.32	49.56
Cyclist	69.49	49.13	45.61	68.75	43.57	41.00	34.80	22.22	20.87
Pedestrian	56.23	52.48	49.71	56.37	47.68	42.58	43.08	36.16	31.01
Class	**Shan, 2020 [8]**			**You, 2022 [9]**			**Proposed: KITTI**		
	Easy	Medium	Hard	Easy	Medium	Hard	Easy	Medium	Hard
Car	87.93	73.13	68.50	87.51	76.85	75.66	89.22	77.95	74.69
Cyclist	64.50	41.96	39.63	60.99	41.80	39.08	72.20	52.72	49.01
Pedestrian	52.24	44.66	39.13	51.75	46.32	42.91	58.37	51.13	46.09
Class	**Proposed: KITTI + CARLA**			**Proposed: KITTI + CARLA (distance)**			**Ground Truth (64-channels)**		
	Easy	Medium	Hard	Easy	Medium	Hard	Easy	Medium	Hard
Car	89.61	78.82	76.24	**89.43**	**78.83**	**76.42**	90.00	84.53	79.27
Cyclist	73.07	51.51	48.75	**76.56**	**54.64**	**50.49**	82.66	65.05	61.36
Pedestrian	58.08	51.21	45.99	**60.22**	**53.50**	**47.89**	61.63	55.22	49.30

Table 3: Comparison table of inference times between different techniques.

Interpolation technique	Inference time GPU: RTX 3060
Bilineal	0,11 [ms]
Bicubic	0,225 [ms]
Simulation-based Lidar Super-resolution for Ground Vehicles [8]	1,5 [s]
Up-Sampling Method for Low-Resolution LiDAR Point Cloud [9]	1,327 [ms]
Proposed interpolation network	1,106 [ms]

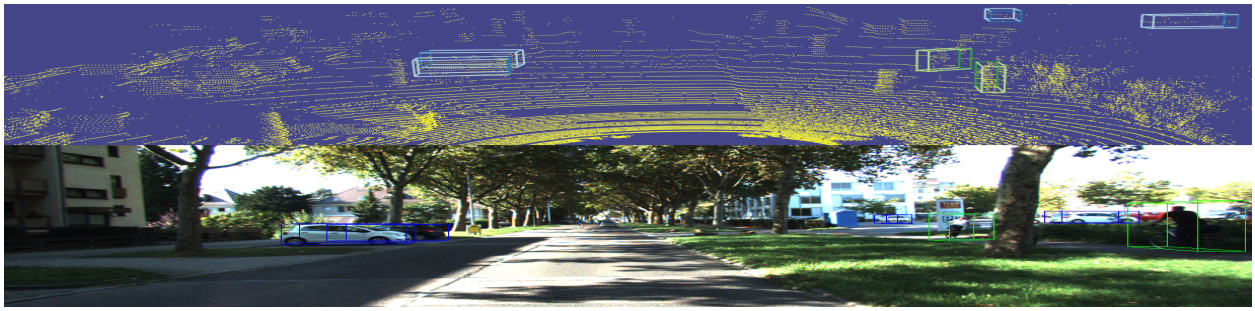


((a)) Detections made by interpolating the low resolution point cloud with the proposed network.

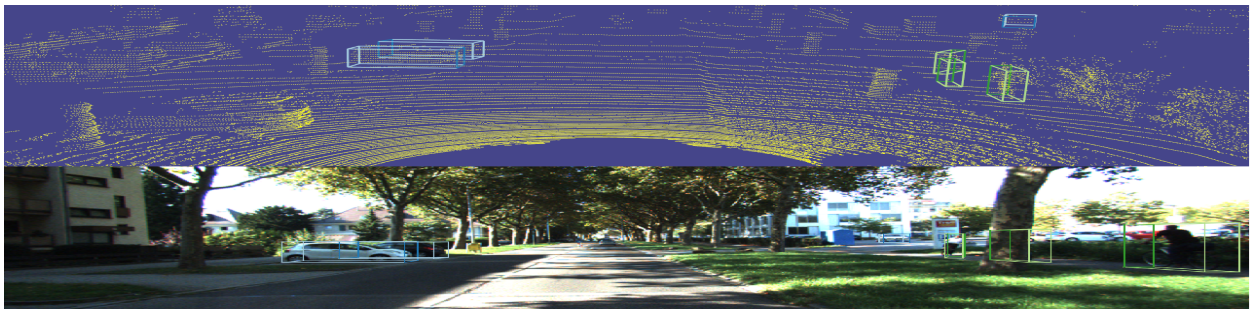


((b)) Ground Truth detections.

Figure 11: Comparison between ground truth detections and those made after interpolating the point clouds with the proposed network using the "000145" KITTI frame.



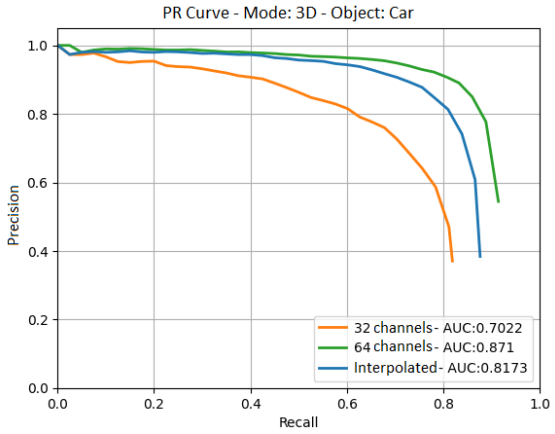
((a)) Detections made by interpolating the low resolution point cloud with the proposed network.



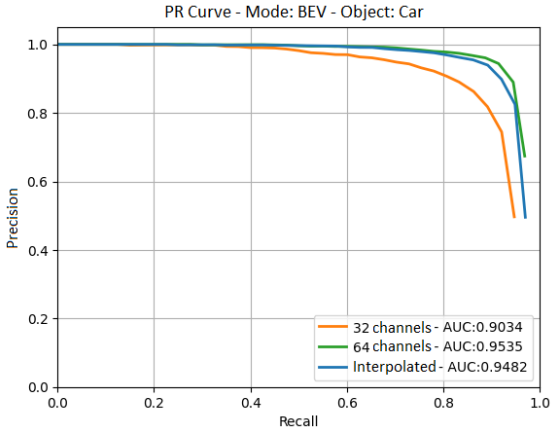
((b)) Ground Truth detections.

Figure 12: Comparison between ground truth detections and those made after interpolating the point clouds with the proposed network using the "000214' KITTI frame.

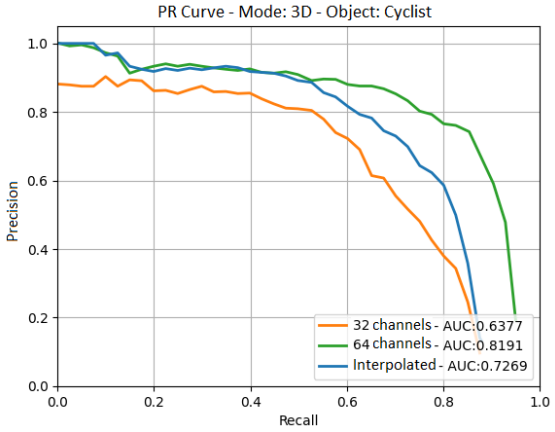
- Total memory required per inference: 2.36 MB, compatible with embedded devices like the Jetson Nano (4 GB RAM) and FPGAs with adequate external memory.
- Inference Time on CPU vs. GPU: GPU (NVIDIA RTX 3060) takes 1.106 ms per inference, while a CPU (Intel Core i5-11400F) needs 32.69 ms per inference.
  A typical Velodyne LiDAR operates at 20 Hz, generating data every 50 ms. The proposed network processes data well within this time, ensuring real-time integration viability.
- Operations per Inference: Each pixel interpolation requires 312 additions and 312 multiplications. For 63,488 windows, 39.6M operations (19.8M additions and 19.8M multiplications) are needed per inference.



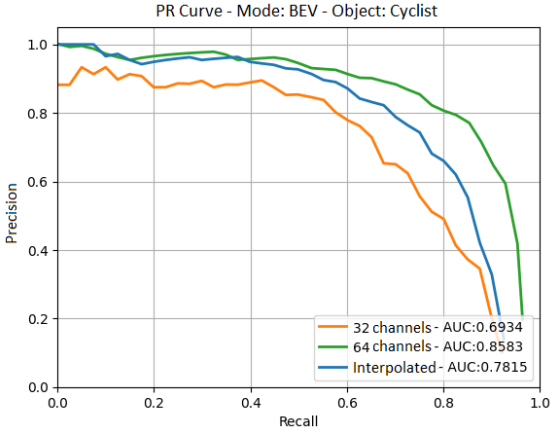
((a)) Class: Car - Detection type: 3D



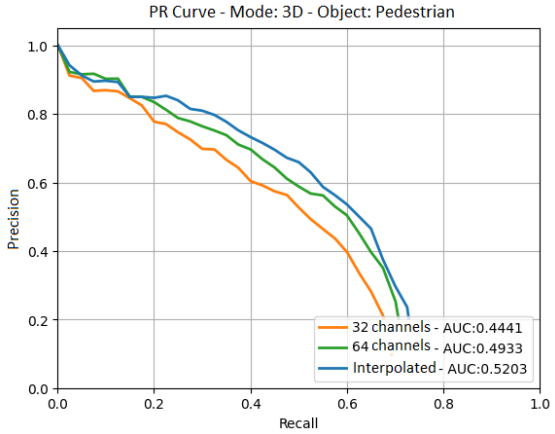
((b)) Class: Car - Detection type: BEV



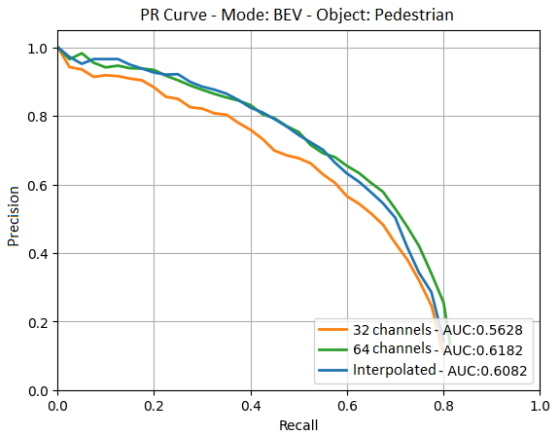
((c)) Class: Cyclist - Detection type: 3D



((d)) Class: Cyclist - Detection type: BEV



((e)) Class: Pedestrian - Detection type: 3D



((f)) Class: Pedestrian - Detection type: BEV

Figure 13: Precision-Recall curves for the different classes (Car, Cyclist, Pedestrian) and detection types (3D and bird's-eye view).

6 Discussion

On one hand, based on the results presented in Tables 1 and 2, certain assessments can be made regarding the benefits of the proposed interpolating network in contrast to the others, under the use of *Pointpillars* as a detection network. These advantages are detailed below:

- The hypothesis that the use of real data in conjunction with simulated data (KITTI + CARLA) provides an improvement in the performance of the system, was confirmed. However, it was also verified that the sensor modeling present in the CARLA simulator fails to faithfully represent the intensity channel of a real LiDAR. This is demonstrated since the use of the intensity interpolation network trained only with the KITTI dataset, in conjunction with the distance interpolation network trained with KITTI + CARLA, yielded better results than that trained using both datasets.
- Considering the network proposed in [9], which had the best performance compared to the others used as a contrast, improvements of 7.93%, 14.07%, 30.98% are observed for the detection of cars, bicycles and pedestrians in 3D, respectively. Under the same analysis, the improvements are 1.92%, 28.47%, 14.46% for the detection of cars, bicycles and pedestrians using the bird's eye view, respectively.
- In general, the proposed technique greatly outperformed the conventional interpolation techniques evaluated (bilinear and bicubic).
- The network proposed in [8] was also surpassed, which requires a very high inference time, in contrast to what was specified by the authors (10 ms), probably due to the computing power of the GPU used (RTX 3060). Even so, the time obtained by our proposed network is around 10 times less than the declared by the authors, thanks to its few parameters and simplicity.
- The object detection metrics obtained on the decimated dataset (32-channel) were surpassed, except for the case of pedestrians in the "Difficult" category in bird's eye view detection, where the value is slightly lower. This shows that the proposed network, in general, improves the performance of the low resolution sensor (32-channel).
- Improvements are generally greater for the "Medium" and "Difficult" categories of the dataset. This demonstrates the superiority of the proposed architecture in complex object detection situations, where objects may be partially hidden.
- Improvements are seen in inference times compared to other non-conventional techniques ([8, 9]).
- Figs. 11 and 12 show a performance very similar to the ground truth detection, these results are consistent with the superior metrics obtained.

On the other hand, based on the results shown in the precision-recall curves, it can be concluded:

- In all 3D detection cases, the blue curve (interpolator model) remains above the orange curve (baseline model, 32 channels) and gets even closer to the green curve (original model, 64 channels), resulting in a greater area under the curve (AUC), consistent with higher precision across almost the entire Recall range. This indicates that the interpolator model maintains a better balance between precision and Recall in the detection of cars, cyclists, and pedestrians.
- Although for bird's-eye view detection cases the blue curve always stays above the orange one, this occurs to a lesser extent compared to 3D detection.
- For pedestrians, and to a lesser extent for cyclists, it is observed that while the interpolator performs better compared to the baseline model, the difference is smaller compared to what is observed for cars. A likely reason for this is the significantly lower presence of pedestrians and cyclists in the dataset used.
- The superior performance in car detection, followed by cyclist detection, may suggest that LiDAR channel interpolation benefits larger objects with less complex shapes (such as cars and cyclists) more than pedestrians, who typically have lower point density or greater variability in position and shape.

Finally, in terms of hardware implementation, the combination of a low-cost, low-resolution LiDAR with a neural network to interpolate data and emulate a high-resolution sensor offers an efficient and affordable solution for embedded applications in production environments. This architecture allows the use of platforms

such as NVIDIA Jetson Nano or mid-range FPGAs, which are significantly cheaper than high-resolution LiDAR sensors.

FPGAs provide energy-efficient inference, while the Jetson Nano allows for flexibility in development and rapid prototyping. Overall, this approach is an accessible and scalable solution, ideal for commercial applications where balancing accuracy, cost, and energy efficiency is critical.

7 Conclusions

There is a growing interest in developing advanced autonomous navigation systems, that facilitate driving without human intervention. This progress is strongly influenced by the use of accurate sensors, especially LiDARs, which increase the reliability and precision in data collection and mapping of the environment surrounding the vehicle, with the associated expenses and requirements of computational power. Aiming to overcoming these disadvantages and to expand the usefulness of cheaper sensors, without leaving aside precision and safety standards, AI techniques were evaluated towards to generate similar results to those available with high-resolution LiDARs, but using low resolution, inexpensive LiDARs.

The proposed method was based on 1D convolutions, fully connected layers and a sliding window that moves along the range image associated with the point cloud, combining established techniques with new strategies to improve accuracy and computational efficiency. Our proposal took advantage of the ability of 1D convolutions to capture local and global patterns in the data, in conjunction with the power of fully connected layers to integrate contextual information across the neural network. In addition, datasets captured by actual LiDAR sensors (KITTI) and generated by a simulator (CARLA) were used.

The results obtained demonstrate the superiority of the proposed technique, compared to other techniques used as contrast, achieving improvements between 1.92% and 30.98% depending on the object and the type of detection (3D or bird's eye view). Moreover, computational efficiency, a crucial aspect of autonomous driving systems, was also addressed. The proposed technique managed to surpass inference times obtained by other techniques used as contrast. Finally, it was demonstrated that the sensor modeling present in CARLA simulator still requires adjustments regarding the generated intensity data, making it an area of special interest for future improvements.

References

[1] T. Wu, H. Fu, B. Liu, H. Xue, R. Ren, and Z. Tu, "Detailed analysis on generating the range image for lidar point cloud processing," *Electronics*, vol. 10, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/11/1224

[2] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, pp. 197–209.

[3] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 689–12 697.

[4] A. Savkin, Y. Wang, S. Wirkert, N. Navab, and F. Tombari, "Lidar upsampling with sliced wasserstein distance," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, p. 392â399, Jan. 2023. [Online]. Available: http://dx.doi.org/10.1109/LRA.2022.3214791

[5] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-net: Point cloud upsampling network," 2018.

[6] W. Zhang, H. Jiang, Z. Yang, S. Yamakawa, K. Shimada, and L. B. Kara, "Data-driven upsampling of point clouds," *Computer-Aided Design*, vol. 112, pp. 1–13, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010448518303956

[7] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "Pu-gan: A point cloud upsampling adversarial network," 10 2019, pp. 7202–7211.

[8] T. Shan, J. Wang, F. Chen, P. Szenher, and B. Englot, "Simulation-based lidar super-resolution for ground vehicles," *Robotics and Autonomous Systems*, vol. 134, p. 103647, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889020304875

[9] J. You and Y.-K. Kim, "Up-sampling method for low-resolution lidar point cloud to enhance 3d object detection in an autonomous driving environment," *Sensors*, vol. 23, no. 1, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/1/322

[10] G. Eskandar, S. Sudarsan, K. Guirguis, J. Palaniswamy, B. Somashekar, and B. Yang, "Hals: A height-aware lidar super-resolution framework for autonomous driving," 2022.

[11] Y. Kwon, M. Sung, and S.-e. Yoon, "Implicit lidar network: Lidar super-resolution via interpolation weight prediction," 05 2022, pp. 8424–8430.

[12] D. Kumar, R. Rajaan, D. Choudhary, and D. Sharma, "A comprehensive review and comparison of image super-resolution techniques," *International Journal of Advanced Engineering, Management and Science*, vol. 10, pp. 40–45, 01 2024.

[13] B. C. Maral, "Single image super-resolution methods: A survey," *ArXiv*, vol. abs/2202.11763, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:247084030

[14] S. Lin, C. Zhang, and Y. Yang, "A pluggable single-image super-resolution algorithm based on second-order gradient loss," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, vol. 3, no. 4, p. 100148, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772485923000650

[15] S. Fadnavis, "Image interpolation techniques in digital image processing: An overview," *International Journal Of Engineering Research and Application*, vol. 4, pp. 2248–962 270, 11 2014.

[16] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[18] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[19] A. Geiger, P. Lenz, and R. Urtasun, "3d object detection evaluation 2017," Available at https://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (2024/03/19).