Robotics and Autonomous Systems 134 (2020) 103647

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Simulation-based lidar super-resolution for ground vehicles

Tixiao Shan [a], Jinkun Wang [b], Fanfei Chen [b], Paul Szenher [b], Brendan Englot [b,*]

[a] Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA
[b] Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, USA

ARTICLE INFO

Article history:
Received 23 March 2020
Accepted 7 September 2020
Available online 30 September 2020

Keywords:
Lidar super-resolution
Range sensing
Perception & driving systems

ABSTRACT

We propose a methodology for lidar super-resolution with ground vehicles driving on roadways, which relies completely on a driving simulator to enhance, via deep learning, the apparent resolution of a physical lidar. To increase the resolution of the point cloud captured by a sparse 3D lidar, we convert this problem from 3D Euclidean space into an image super-resolution problem in 2D image space, which is solved using a deep convolutional neural network. By projecting a point cloud onto a range image, we are able to efficiently enhance the resolution of such an image using a deep neural network. Typically, the training of a deep neural network requires vast real-world data. Our approach does not require any real-world data, as we train the network purely using computer-generated data. Thus our method is applicable to the enhancement of any type of 3D lidar theoretically. By novelly applying Monte-Carlo dropout in the network and removing the predictions with high uncertainty, our method produces high accuracy point clouds comparable with the observations of a real high resolution lidar. We present experimental results applying our method to several simulated and real-world datasets. We argue for the method's potential benefits in real-world robotics applications such as occupancy mapping and terrain modeling.

1. Introduction

Light detection and ranging (lidar) is an essential sensing capability for many robot navigation tasks, including localization, mapping, object detection and tracking. Lidar uses light in the form of pulsed laser to measure relative range to surrounding objects. Unlike most cameras, which only function with sufficient ambient light, lidar will function even at night, offering long-range visibility and a wide horizontal aperture. 2D lidar is usually cost-efficient and has been widely used in many indoor applications such as mapping, localization, and obstacle avoidance. Recently, with the rapid development of self-driving vehicles, demand for 3D lidar has grown significantly. Though a revolving 2D lidar can mimic a 3D lidar by continuously changing the scanning position, such systems are often inefficient. A typical 3D lidar has multiple channels that revolve at different heights, producing a 3D point cloud with ring-like structure. The number of channels in the sensor determines the vertical density of its point clouds. A denser point cloud from a lidar with more channels can capture the fine details of the environment; applications such as terrain modeling and object detection can benefit greatly from a higher resolution lidar. However, increasing the number of channels can be very costly. For example, the most popular 16-channel lidar, the Velodyne VLP-16, costs around $4000. The 32-channel HDL-32E and Ultra Puck, and the 64-channel HDL-64E cost around $30,000, $40,000 and $85,000 respectively.

In this paper, we propose what is to our knowledge the first dedicated deep learning framework for *lidar super-resolution*, which predicts the observations of a high-resolution (hi-res) lidar over a scene observed only by a low-resolution (low-res) lidar. We convert the resulting super-resolution (super-res) point cloud problem in 3D Euclidean space into a super-res problem in 2D image space, and solve this problem using a deep convolutional neural network. Unlike many existing super-res image methods that use high-res real-world data for training a neural network, we train our system using only computer-generated data from a simulation environment. This affords us the flexibility to train the system for operation in scenarios where real hi-res data is unavailable, and allows us to consider robot perception problems beyond those pertaining specifically to driving with passenger vehicles. We investigate the benefits of deep learning in a setting where much of the environment is characterized by sharp discontinuities that are not well-captured by simpler interpolation techniques. Furthermore, we use Monte-Carlo dropout [1,2] to approximate the outputs of a Bayesian Neural Network (BNN) [3], which naturally provides uncertainty estimation to support our inference task.

* Corresponding author.
E-mail addresses: shant@mit.edu (T. Shan), jwang92@stevens.edu (J. Wang), fchen7@stevens.edu (F. Chen), pszenher@stevens.edu (P. Szenher), benglot@stevens.edu (B. Englot).

https://doi.org/10.1016/j.robot.2020.103647
0921-8890/

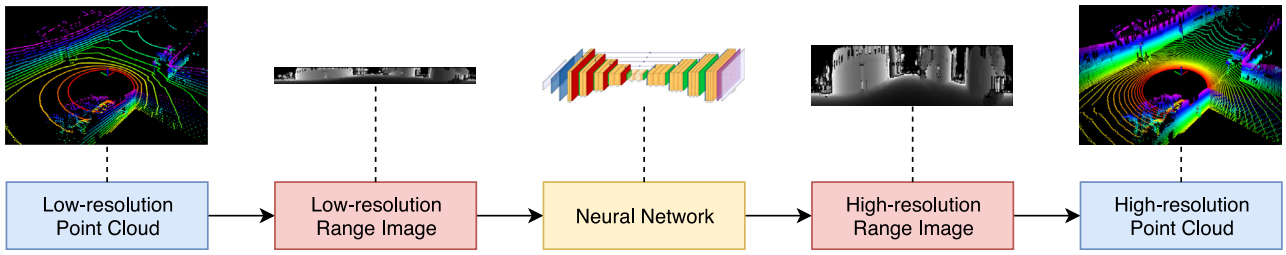


Fig. 1. Workflow for lidar super-resolution. Given a sparse point cloud from a 3D lidar, we first project it and obtain a low-res range image. This range image is then provided as input to a neural network, which is trained purely on simulated data, for upscaling. A high-res point cloud is received by transforming the inferred high-res range image pixels into 3D coordinates.

To the best of our knowledge, this is the first paper to present an approach for lidar super-resolution enabled by deep learning. It produces accurate high-res point clouds that predict the observations of a high-res lidar using low-res data. The contributions of this paper are as follows:

- A novel architecture for deep learning-enabled lidar super-resolution;
- A procedure for training the architecture in simulation, which is thoroughly evaluated using datasets recorded in both simulated and real-world environments;
- A study of the framework's suitability for enhancing relevant robot mapping tasks, with comparisons against both deep learning and simpler interpolation techniques.

2. Related work

Our work is most related to the *image super-resolution* problem, which aims to enhance the resolution of a low-res image. Many techniques have been proposed over the past few decades and have achieved remarkable results [4]. Traditional approaches such as linear or bicubic interpolation [5], or Lanczos resampling [6], can be very fast but oftentimes yield overly smooth results. Recently, with developments in the machine learning field, deep learning has shown superiority in solving many prediction tasks, including the image super-res problem. Methods based on deep learning aim to establish a complex mapping between low-res and high-res images. Such a mapping is usually learned from massive training data where high-res images are available. For example, a super-resolution convolutional neural network, SR-CNN, trains a three-layer deep CNN end-to-end to upscale an image [7]. Over time, deeper neural networks with more complex architectures have been proposed to further improve the accuracy [8–11]. Among them, SR-GAN [11] achieves state-of-the-art performance by utilizing a generative adversarial network [12]. The generator of SR-GAN, which is called SR-ResNet, is composed of two main parts, 16 residual blocks [13] and an image upscaling block. A low-res image is first processed by the 16 residual blocks that are connected via skip-connections and then upscaled to the desired high resolution. The discriminator network of SR-GAN is a deep convolutional network that performs classification. It discriminates real high-res images from generated high-res images. It outperforms many other image super-res methods, including nearest neighbor, bicubic, SR-CNN and those of [8–10], by a large margin.

Another problem that is related to our work is *depth completion*. The goal of this task is to reconstruct a dense depth map with limited information. Such information usually includes a sparse initial depth image from a lidar or from an RGB-D camera [14,15]. Typically, an RGB image input is also provided to support depth completion, since estimation solely from a single sparse depth image is oftentimes ambiguous and unreliable. For instance, a fast depth completion algorithm that runs on a CPU is proposed in [16]. A series of basic image processing operations, such as dilation and Gaussian blur, are implemented for acquiring a dense depth map from a sparse lidar scan. Though this method is fast and does not require training on vast data, its performance is inferior when compared with many other approaches. A self-supervised depth completion framework is proposed in [17]. In this work, a deep regression network is developed to predict dense depth from sparse depth. The proposed network resembles an encoder–decoder architecture and uses sparse depth images generated by a lidar, with RGB images as optional inputs. Another problem that is closely related to depth completion is *depth prediction*, which commonly utilizes images from a monocular or stereo camera [18–21]. Due to our focus here on a lidar-only super-resolution method, an in-depth discussion of this problem lies beyond the scope of this paper.

Instead of solving the super-resolution problem in image space, PU-Net [22] operates directly on point clouds for upsampling, and adopts a hierarchical feature learning mechanism from [23]. However, this approach performs super-resolution on point cloud models of individual small objects, which differs from our approach that attempts to increase sensor resolution. The upsampled high-res point clouds of our method retain the "ring" structure characterizing the output of a real lidar. This preserves our approach's compatibility with other lidar perception algorithms that operate directly on 3D lidar scans as input.

3. Technical approach

This section describes the proposed lidar super-resolution methodology in detail. Since the horizontal resolution of a modern 3D lidar is typically high enough, we only enhance vertical resolution throughout this paper. However, the proposed approach, without loss of generality, is also applicable for enhancing the horizontal resolution of a lidar with only a few modifications to the neural network. The workflow of the proposed approach is shown in Fig. 1. Given a sparse point cloud from a 3D lidar, we first project it and obtain a low-res range image. This range image is then provided as input to a neural network, which is trained purely on simulated data, for upscaling. A dense point cloud is received by transforming the inferred high-res range image pixels into 3D coordinates.

3.1. Data gathering

Similar to the method proposed in [24], we leverage a rich virtual world as a tool for generating high-res point clouds with simulated lidar. There are many open source software packages, e.g. CARLA, Gazebo, Unity, that are capable of simulating various kinds of lidar on ground vehicles. Specifically, we opt to use the CARLA simulator [25] in this paper due to its ease of use and thorough documentation.

The first step involves identifying the lidar we wish to simulate. Let us assume we have a VLP-16 and we wish to quadruple

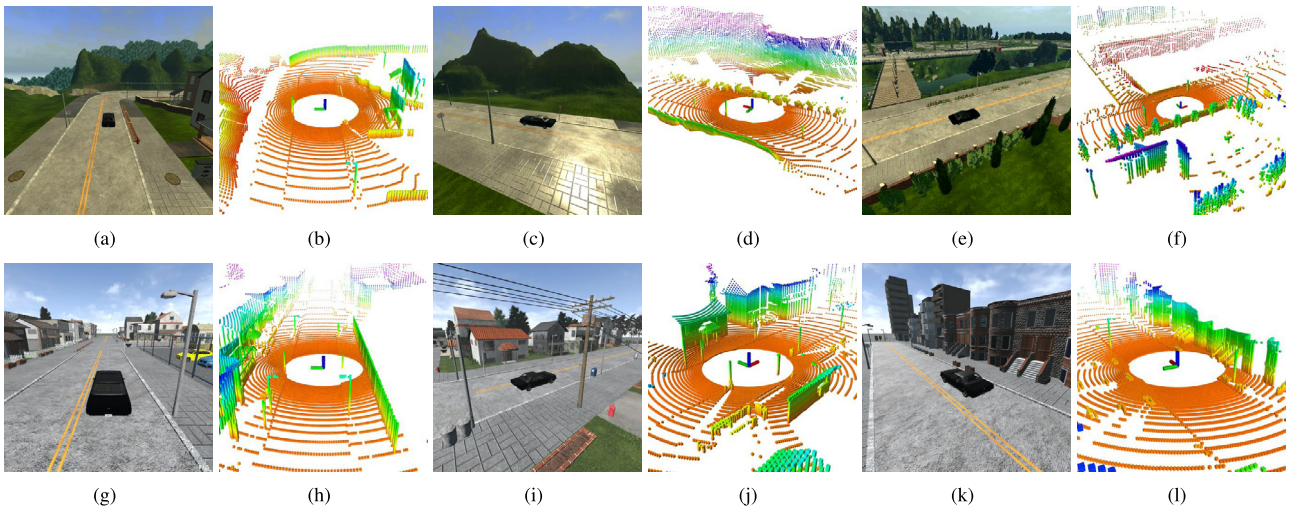


Fig. 2. Illustration of high-res point clouds (64-channel) captured in CARLA Town 01 (a–f) and Town 02 (g–l). CARLA Town 01 features a suburban environment with roads, trees, houses, and a variety of variable-height terrain. Town 02 features an urban environment.

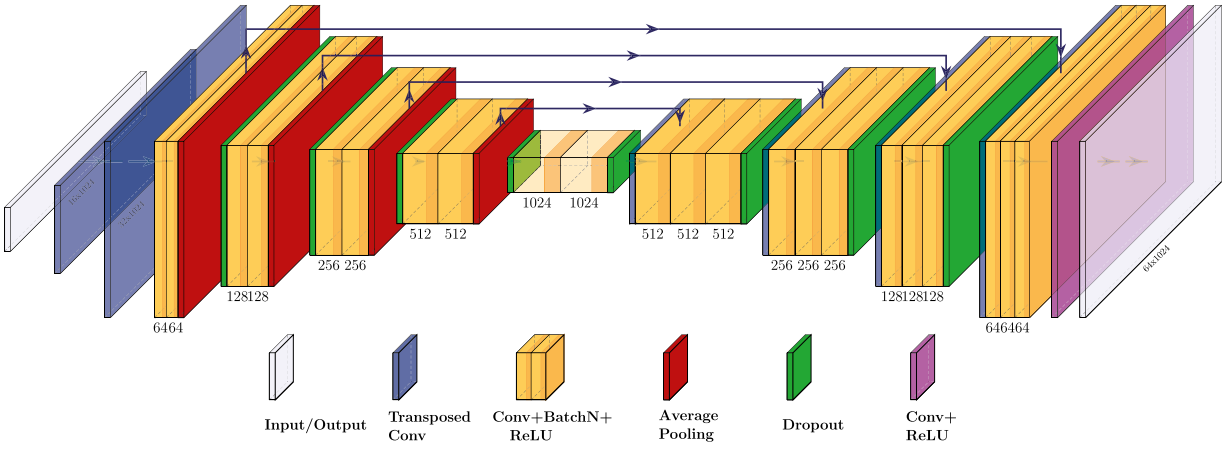


Fig. 3. Our proposed neural network architecture for range image super-resolution. The network follows an encoder–decoder architecture. Skip-connections are denoted by solid lines with arrows.

(4× upscaling (16 to 64)) its resolution. The VLP-16 has a vertical field of view (FOV) of 30° and a horizontal FOV of 360°. The 16-channel sensor provides a vertical angular resolution of 2°, which is very sparse for mapping. We want to simulate a 64-channel "VLP-64" in CARLA, which also has a vertical and horizontal FOV of 30° and 360° respectively. With the simulated lidar identified, we can either manually or autonomously drive a vehicle in the virtual environment and gather high-res point clouds captured by this simulated lidar. An example of the lidar data produced in CARLA is shown in Fig. 2.

We note that the simulated high-res lidar should have the same vertical and horizontal FOV as the low-res lidar. For example, we cannot train a neural network that predicts the perception of HDL-64E using the data from VLP-16 because their vertical FOVs are different.

3.2. Data preparation and augmentation

We then project the simulated high-res point cloud onto a range image, which can be processed by the neural network. A scan from the simulated "VLP-64" 3D lidar will yield a high-res range image with a resolution of 64-by-1024. This high-res range image will serve as the ground truth comprising our training data. Then, we evenly extract 16 rows from this high-res range image and form a low-res range image, which has a resolution of 16-by-1024. This low-res range image is equivalent to the point cloud data captured by a VLP-16 after projection, and comprises the input to the neural network during training. We note that the resolution of the original range image from a VLP-16 sensor varies from 16-by-900 to 16-by-3600 depending on the sensor rotation rate. For the purpose of convenience and demonstration, we choose the horizontal resolution of all range images to be 1024 to accommodate different sensors throughout the paper. We also "cut" every range scan at the location facing the rear of the vehicle, for the purpose of converting it to a flattened 2D image. This is typically the region of least importance for automated driving and robot perceptual tasks, and is in many cases obstructed by the body of the vehicle.

We then augment the data by performing top-down flipping, horizontal flipping and shifting, and range scaling to account for different environment structures and sensor mounting heights (such as driving on different sides of the road, and underneath structures). To increase prediction robustness, we also vary sensor mounting attitudes during data gathering before augmentation. Finally, the low-res and high-res range images are then normalized to $0 - 1$ and sent to train the neural network. For example, the maximum detection range of the VLP-16 is 100 m. Thus we divide the ranges of the range image by 100 to obtain the

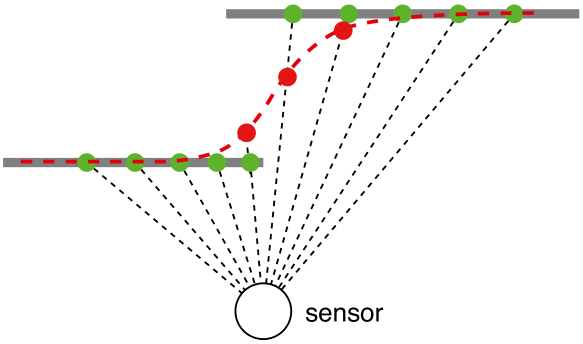


Fig. 4. Smoothing effects after applying convolutions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

normalized range image. For objects that are outside the sensor's range, their corresponding ranges in the image are set to be 0 as they yield no valid readings.

3.3. Neural network architecture

The lidar super-res problem can now be formulated as an image super-res problem. Adapted from the encoder–decoder architecture of [26], we configure a neural network for range image super-resolution, shown in Fig. 3. The input, low-res range image is first processed by two transposed convolutions for increasing the image resolution to the desired level. Then the encoder consists of a sequence of convolutional blocks and average pooling layers for downsampling the feature spatial resolutions while increasing filter banks. On the other hand, the decoder has a reversed structure with transposed convolutions for upsampling the feature spatial resolutions. All convolutions in the convolutional blocks are followed by batch normalization [27] and ReLU [28]. The output layer produces the final high-res range image using a single convolution filter without batch normalization.

3.4. Noise removal

We note that we have placed numerous dropout layers before and after the convolutional blocks in Fig. 3. This is because performing convolutional operations on a range image will unavoidably cause smoothing effects on sharp and discontinuous object boundaries [29]. An illustrative example of this smoothing effect is shown in Fig. 4. Ten range measurements from a lidar channel are shown in a top-down view. The gray lines represent two walls, and the green dots indicate the true range measurements. After convolution, the range measurements are smoothed (shown by the red curve) in places where environmental discontinuities occur. Incorporating smoothed range predictions, such as the three red dots shown, into a robot's point cloud will deteriorate the accuracy of the resulting map.

To address this problem, we novelly apply Monte-Carlo dropout (MC-dropout) for estimating the uncertainty of our range predictions [2]. MC-dropout regularization approximates a BNN by performing multiple feed-forward passes with active dropout at inference time to produce a distribution over outputs [2]. Given observations $\mathcal{D} = \{(\mathbf{x_i}, y_i)_{i=1:N}\}$, we seek to infer a probability distribution of a target value parameterized on the latent space Θ:

$$p(y^*|\mathbf{x}^*, \mathcal{D}) \propto \int p(y^*|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*|\mathbf{x}^*, \mathcal{D})d\boldsymbol{\theta}^*, \tag{1}$$

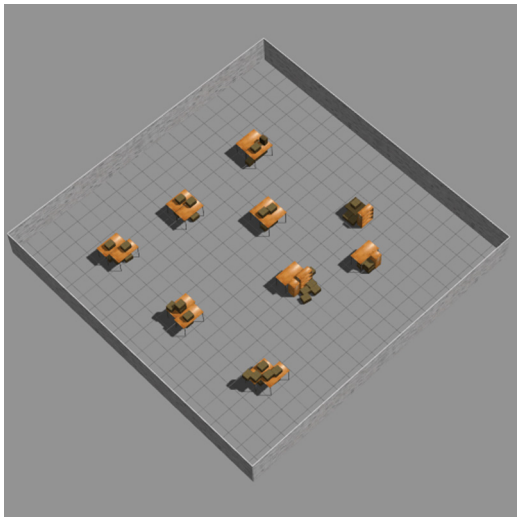


Fig. 5. A total number of 25 scans are obtained in this simulated office-like environment for generating Octomaps using different methods.

where $\boldsymbol{\theta}^*$ are the latent parameters associated with the target input. More specifically, given a test image $\mathbf{x}^*$, the network performs T inferences with the same dropout rate used during training. We then obtain:

$$p(y^*|\mathbf{x}^*) = \frac{1}{T}\sum_{t=1}^{T} p(y^*|\mathbf{x}^*, \boldsymbol{\theta}_t^*), \tag{2}$$

where $\boldsymbol{\theta}_t^*$ are the weights of the network for the tth inference, and y^* are the averaged predictions. We can evaluate the uncertainty of our range predictions by inspecting the variance of this probability distribution. The final prediction is as follows:

$$y_f^* = \begin{cases} y^*, & \text{if } \sigma < \lambda y^* \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

in which y^* is the predicted mean from Eq. (2), and σ is its standard deviation:

$$\sigma = \sqrt{\frac{1}{T}\sum_{i=1}^{T}(y_i^* - y^*)}, \tag{4}$$

where y_i^* is the value of the ith prediction.

The parameter λ causes the noise removal threshold to scale linearly with the predicted sensor range, capturing the fact that the noise level worsens with distance from the sensor. Throughout this paper we choose a value of 0.03 for λ, as it is found to give the most accurate mapping results, and we choose an inference quantity T of 50 for all experiments. A larger T yields improved results, as the true probability distribution $p(y^*|\mathbf{x}^*)$ can be better approximated with more predictions.

Since the predictions of this step are between 0 and 1, to obtain the high-res point cloud, we first multiply the range image by the normalization value used in Section 3.2. Then we project the high-res range image back to a high-res point cloud. Note that we do not generate points from the zero-valued pixels in the range image, as they are yielded either from noisy predictions deleted from the range image, occluded objects, or objects outside the sensor's range.

4. Experiments

We now describe a series of experiments to quantitatively and qualitatively analyze the performance of our lidar super-resolution architecture. We perform 4× upscaling (16 to 64) for

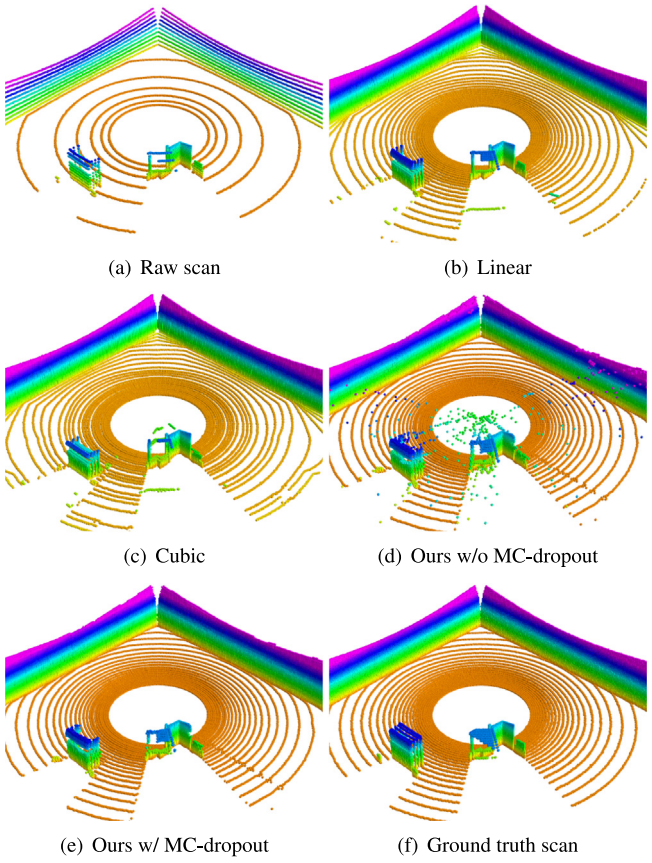


Fig. 6. An example lidar input (a), predictions by linear and cubic interpolation (b and c), and our methods without and with MC-dropout (d and e) and ground truth (f). As is shown in (d), the inferred point cloud is noisy due to points that have high uncertainty, motivating our use of MC-dropout. Color variation indicates elevation change. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

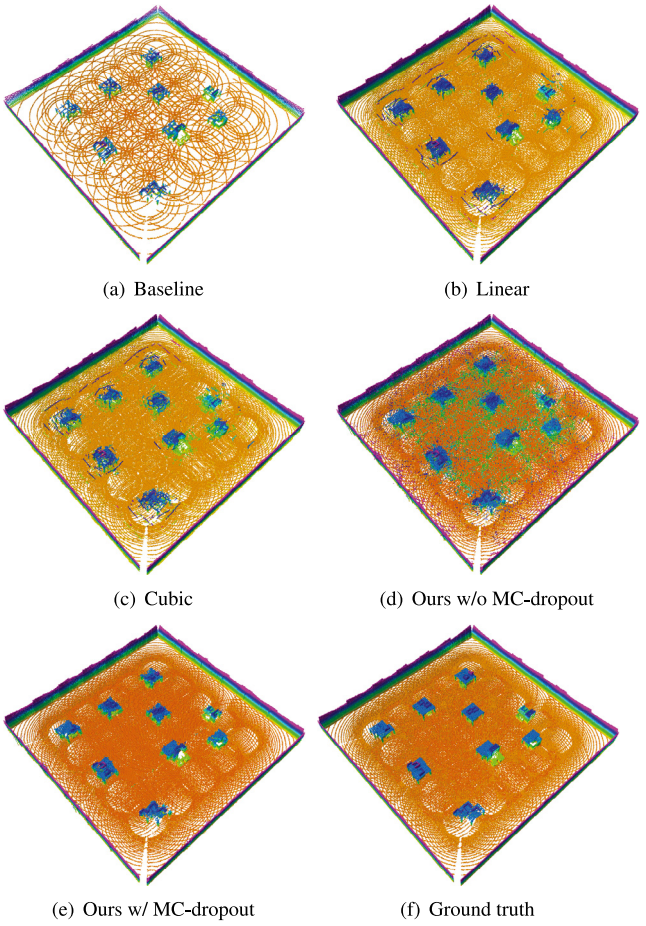


Fig. 7. Full occupancy mapping results generated using the simulated indoor dataset. Color variation indicates elevation change. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

all experiments in this section. For more experimental results, please refer to the supplementary Appendix.

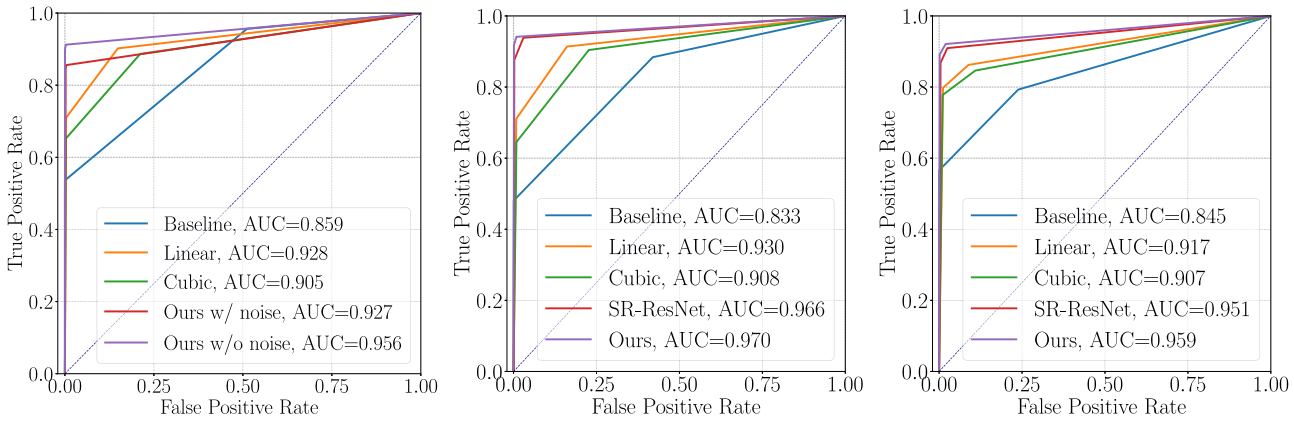
For network training, Adam optimizer [30] is used with a learning rate of 10^{-4} and decay factor of 10^{-5} after each epoch. $\mathcal{L}1$ loss, the sum of absolute differences between the true values and the predicted values associated with range image pixels, is utilized for penalizing the differences between the network output and ground truth, as it achieves high accuracy, fast convergence and improved stability during training. A computer equipped with a Titan RTX GPU was used for training and testing. The training framework was implemented in Keras [31] using Tensorflow [32] as a backend in Ubuntu Linux. The software package of the proposed method is publicly available.[1]

Besides benchmarking various methods in 2D image space using $\mathcal{L}1$ loss, we also show that our method is able to produce dense Octomaps [33] with high accuracy in 3D Euclidean space. 3D occupancy maps can support a variety of robotics applications, e.g., planning [34] and exploration [35,36]. However, sparsity in the point cloud of a 3D lidar can leave gaps and inconsistencies in traditional occupancy grid maps, which can be misleading when applied in planning and exploration scenarios. Intuitively, 3D occupancy mapping can benefit greatly from a higher resolution lidar. We use receiver operating characteristic (ROC) curves to benchmark the predictive accuracy (with respect to the binary classification of occupancy) of each method. The ROC curves plot the true positive rate against the false positive rate. We compare all methods to the ground-truth occupancy (0 — free, 0.5 — unknown, 1 — occupied) for all cells in the map. The area under the curve (AUC) is provided for each method for comparison of prediction accuracy. We treat the underlying 64-channel range scan as ground truth, rather than a complete map with all cells filled, because our specific goal is to truthfully compare the range prediction accuracy of each method.

For the simulated experiments described in Sections 4.1 and 4.2, we use the exact same neural network to demonstrate that the proposed method is capable of performing accurate prediction for sensors with different mounting positions in different environments. The training data for the neural network is gathered from CARLA Town 02, which features an urban environment, by simulating a 64-channel lidar "VLP-64" that has a vertical FOV of 30°. A low-res 16-channel lidar scan is obtained by evenly extracting 16-channel data from the high-res data. The low-res data here is equivalent to the scan obtained from the VLP-16. The training dataset contains 20,000 scans after data augmentation.

Since the real-world Ouster lidar used in Section 4.3 has a different FOV (33.2°), we gather a new training dataset for network training (see Section 3.1). Similarly, we simulate a 64-channel lidar, the OS-1-64, in CARLA Town 02 and gather high-res data. The 16-channel data is extracted in the same way as described

[1] https://github.com/RobustFieldAutonomyLab/lidar_super_resolution.

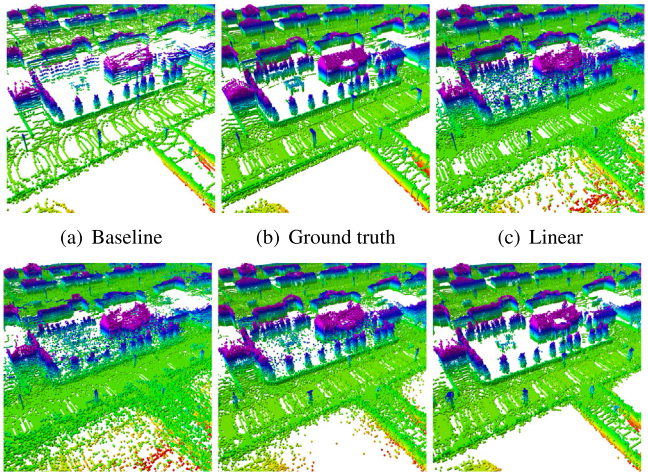


(a) Simulated indoor dataset (b) Simulated outdoor dataset (c) Ouster dataset

Fig. 8. Receiver operating characteristic (ROC) curves and area under the curve (AUC) for all competing methods. The results are obtained by comparing the Octomaps of each method with the ground truth Octomap. Though the neural network is trained using the data from a completely different map (CARLA Town 02), our proposed method produces dense Octomaps with the highest AUC among all methods evaluated in all experiments. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(a) (b)

Fig. 9. Two representative scenes from CARLA Town 01. The resulting Octomaps are shown in Figs. 10 and 11.

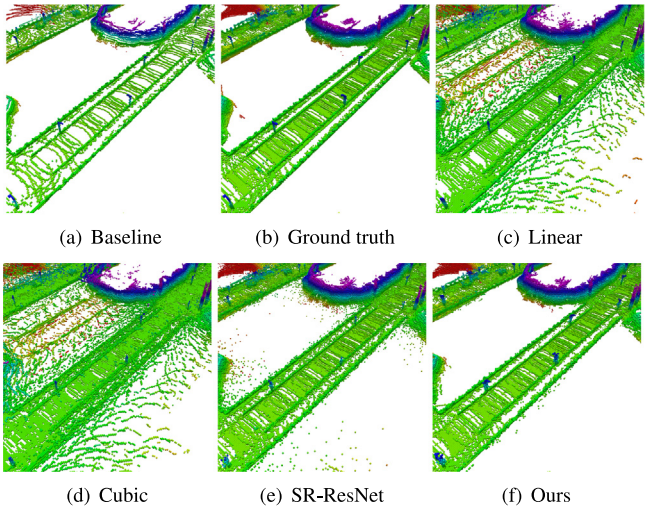


(a) Baseline (b) Ground truth (c) Linear

(d) Cubic (e) SR-ResNet (f) Ours

Fig. 10. Occupancy mapping results for scene shown in Fig. 9(a). Color variation indicates elevation change. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

before. The low-res data here is equivalent to the scan obtained from an OS-1-16 sensor. The training dataset also contains 20,000 scans after data augmentation.



(a) Baseline (b) Ground truth (c) Linear

(d) Cubic (e) SR-ResNet (f) Ours

Fig. 11. Occupancy mapping results for scene shown in Fig. 9(b). Color variation indicates elevation change. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Simulated indoor dataset

We first demonstrate the benefits of applying MC-dropout. We simulate a 64-channel lidar "VLP-64" and gather 25 high-res scans in an office-like environment in Gazebo. The lidar is assumed to be installed on top of a small unmanned ground vehicle (UGV). The sensor is 0.5 m above the ground. As is shown in Fig. 5, the environment is populated with desks and boxes. The low-res 16-channel testing scans are obtained by evenly extracting 16-channel data from the high-res data. Note that none of these scans are used for network training, nor is the height at which the sensor is mounted.

A representative low-res scan is shown in Fig. 6(a). Using this scan as input, the predicted high-res scans using the naive linear and cubic interpolation methods are shown in Fig. 6(b) and (c). The predictions using our method with and without the application of MC-dropout are shown in Fig. 6(d) and (e). Without applying MC-dropout, the range prediction is noticeably noisy due to the smoothing effect caused by convolution, hence the scan shown in Fig. 6(d). After noise removal by applying MC-dropout, the predicted scan shows significantly less noise and

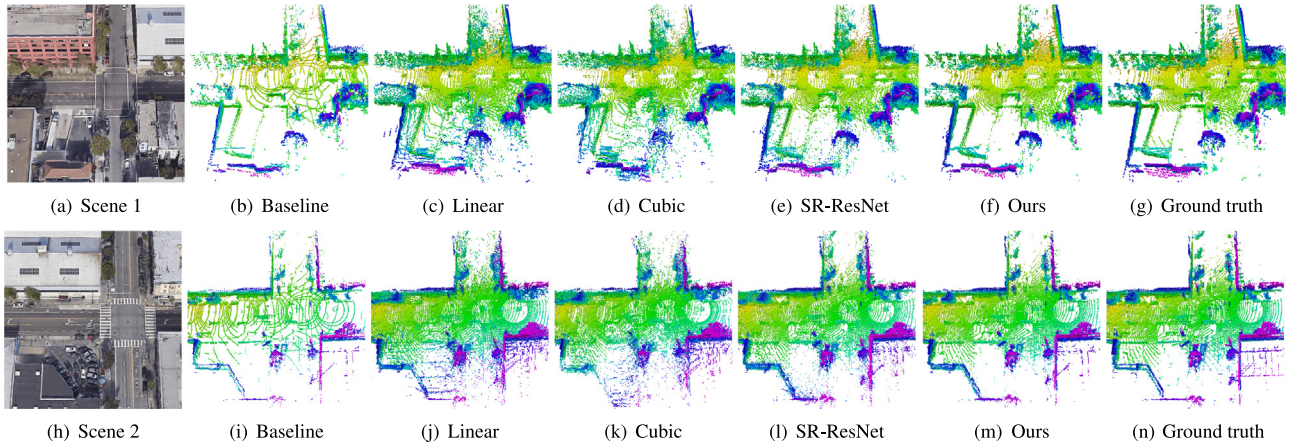


Fig. 12. Occupancy mapping results using the Ouster dataset. Color variation indicates elevation change. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Quantitative results for the experiments discussed in Sections 4.2 and 4.3.

Dataset	Method	$\mathcal{L}1$ loss	Removed points (%)
CARLA Town 01	Linear	0.0184	N/A
	Cubic	0.0303	N/A
	SR-ResNet	0.0089	12.37
	Ours	0.0087	4.13
Ouster	Linear	0.0324	N/A
	Cubic	0.0467	N/A
	SR-ResNet	0.0211	17.70
	Ours	0.0214	8.37

resembles the scan of ground truth. The resulting maps are shown in Fig. 7. All the Octomaps have a resolution of 0.05 m. We refer to the approach of using low-res lidar scans to produce an Octomap as the *baseline* approach. The ground truth Octomap is obtained by using the high-res scans. The map of baseline approach is sparse, as no inference is performed. As is shown in Fig. 7(e), the proposed method is able to produce a dense Octomap that resembles the ground truth Octomap. The receiver operating characteristic (ROC) curves and area under the curve (AUC) for each method are shown in Fig. 8(a). The AUC is improved when applying MC-dropout.

We also note that though the network is trained using data from an outdoor environment — CARLA Town 02, our method is capable of producing meaningful and accurate predictions for indoor usage, with a different sensor mounting scheme. This demonstrates that the network is able to learn the complex mapping between low-res input and high-res output while properly maintaining the structure of surrounding objects. More comparisons of this experiment can be found in the Appendix.

4.2. Simulated outdoor dataset

In this experiment, we compare our method with various approaches, which include the standard linear and cubic interpolation techniques and also the state-of-the-art super-resolution approach — SR-ResNet [11], using a simulated large scale outdoor dataset that is gathered in CARLA Town 01. CARLA Town 01 features a suburban environment with roads, trees, houses, and a variety of terrain. The same sensor that is used in 4.1 is used here. The "VLP-64" sensor, which has a height of 1.8 m from the ground, is mounted on top of a full-sized passenger vehicle. We drive the vehicle along a trajectory of 3300 m and gather a lidar scan every 10 m. Thus this dataset contains 330 scans.

The $\mathcal{L}1$ loss of each method is shown in Table 1. The deep learning approaches outperform the traditional interpolation approaches by a large margin. For fair comparison, we also apply MC-dropout on SR-ResNet by adding a dropout layer to the end of each residual block for noise removal. The losses of SR-ResNet and our method are very close. However, the amount of noise removed per scan from SR-ResNet is much larger than our method. Though we can adjust λ in Eq. (3) to retain more points, the mapping accuracy deteriorates greatly as more noisy points are introduced. We can also decrease the value of λ for SR-ResNet to filter out more noise. The mapping accuracy then also deteriorates, as more areas in the map become unknown.

The Octomaps of the competing methods using a low-res scan as input are shown in Figs. 10 and 11. The baseline approach naturally yields the most sparse map. Though offering better coverage, the Octomaps of the linear and cubic methods are very noisy due to range interpolation between different objects. The deep learning-enabled approaches, SR-ResNet and our proposed method, outperform the interpolation-based approaches by offering true representation of the environment. Though SR-ResNet outperforms linear and cubic interpolation methods in 2D image space by yielding smaller $\mathcal{L}1$ loss, its predictions, when shown in 3D Euclidean space, still contain a great deal of noise at object boundaries. Our proposed approach introduces much less noise into the map. As a result, our method produces a map that is easier to interpret visually, and which also achieves the highest AUC among all methods. The AUC and ROC curves for each method using 330 scans are shown in Fig. 8(b).

4.3. Real-world outdoor dataset

In this experiment, we evaluate the proposed method over one publicly available driving dataset, which we refer to as *Ouster*.[2] The Ouster dataset, which consists of 8825 scans over a span of 15 min of driving, is captured in San Francisco, CA using an Ouster OS-1-64 3D lidar. This 64-channel sensor naturally gives us the ground truth for validation, as we only need to extract a few channels of data for generating low-res range image inputs. Again, the networks evaluated here are purely trained using a simulated dataset gathered from CARLA Town 02. As is shown in

[2] https://git.io/fhbBt.

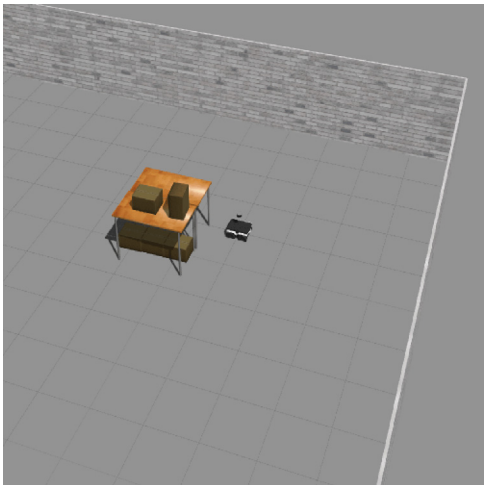


Fig. A.13. A local region of the environment shown in Fig. 5.

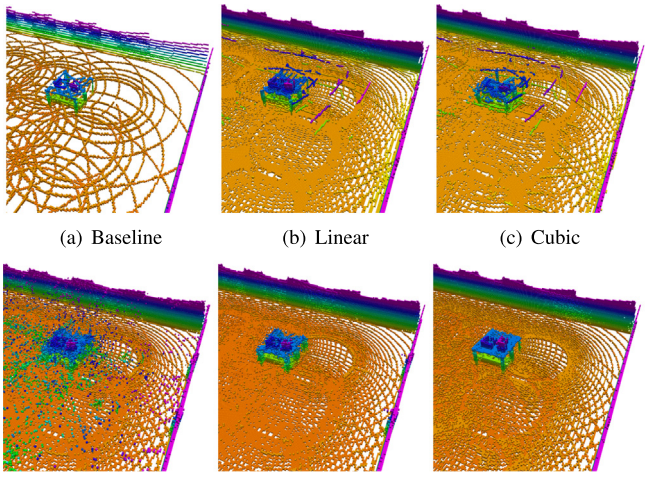


Fig. A.14. Mapping results of the area shown in Fig. A.13 using several competing methods. Color variation indicates elevation change. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

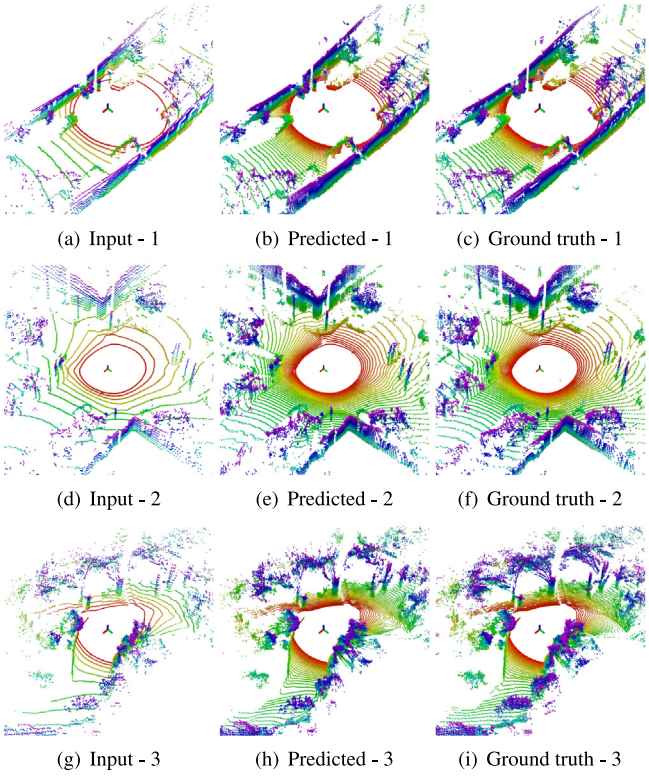


Fig. B.15. Visualization of several representative point clouds from the Ouster dataset. The low density point clouds before inference are shown in (a), (d) and (g). The inferred high-res point clouds (4× upscaling) are shown in (b), (e) and (h). The ground truth point cloud captured by the lidar is shown in (c), (f) and (i). Color variation indicates lidar "ring" index. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

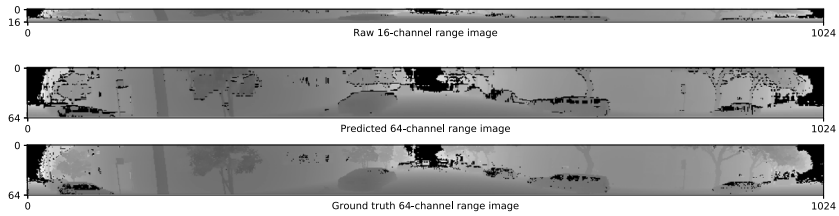
Table 1, both SR-ResNet and our method achieve similar $\mathcal{L}1$ loss, which is evaluated over 8825 scans. We note that the $\mathcal{L}1$ loss of SR-ResNet is slightly smaller than that of our proposed method. This is because we compute the $\mathcal{L}1$ loss using all the predicted ranges without applying Eq. (3), to ensure a fair comparison. However, the percentage of removed points of our approach is much less when compared with the results of SR-ResNet. This is because the predictions of SR-ResNet are more influenced by the smoothing effects discussed in Section 3.4. In other words, the predictions of our approach are of lower variance.

We use 15 scans from this dataset to obtain real-world low-res and high-res lidar scans, which are then used to obtain Octomaps, in the same way that is described in our previous experiments. The scans are registered using LeGO-LOAM [37]. The mapping results at two intersections are shown in Fig. 12. All the Octomaps have a resolution of 0.3 m. The AUC and ROC curves for each method using these 15 scans are shown in Fig. 8(c). Again, our proposed approach outperforms all methods by achieving the highest AUC. For the mapping visualization using all 15 scans, please refer to the Appendix. A visualization of the inference performed throughout the dataset can be found in our video attachment.[3]

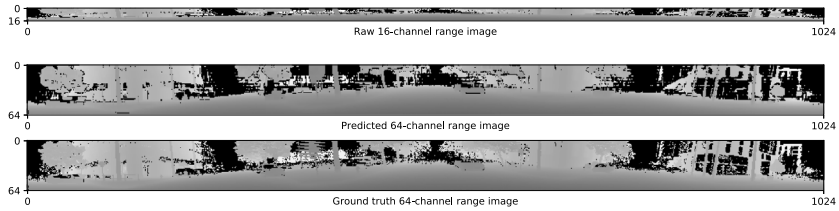
5. Conclusions and discussion

We have proposed a lidar super-resolution method that produces high resolution point clouds with high accuracy. Our method transforms the problem from 3D Euclidean space to an image super-resolution problem in 2D image space, and deep learning is utilized to enhance the resolution of a range image, which is projected back into a point cloud. We train our neural network using computer-generated data, which affords the flexibility to consider a wide range of operational scenarios. We further improve the inference accuracy by applying MC-dropout. The proposed method is evaluated on a series of datasets, and the results show that our method can produce realistic high resolution maps with high accuracy. In particular, we evaluate the super-resolution framework through a study of its utility for *occupancy mapping*, since this is a widely useful perceptual end-product that robots may use to support planning, exploration, inspection, and other activities. In addition to the appealing generalizability of up-scaling at the front end, by predicting the measurements of

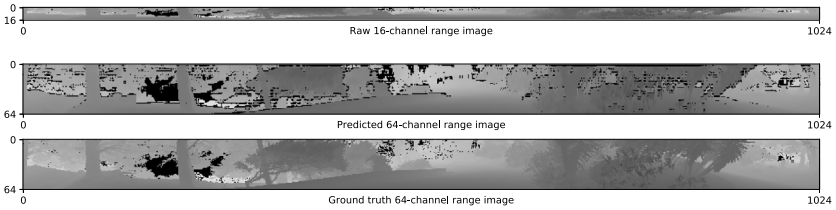
[3] https://youtu.be/rNVTpkz2ggY.



(a) Range images for scene 1 shown in Fig. B.15



(b) Range images for scene 2 shown in Fig. B.15



(c) Range images for scene 3 shown in Fig. B.15

Fig. B.16. Range images of the projected point clouds shown in Fig. B.15.

a higher-resolution sensor, our approach also achieves superior accuracy in the end-stage maps produced, as compared with both deep learning methods and simpler interpolation methods.

Future work may involve using a generative adversarial network to further improve the inference quality. We conducted tests using SR-GAN, which is proposed in [11]. However, we did not obtain additional benefits from using this network for our tests. The discriminator is not able to distinguish generated high-res range images from real high-res range images with an accuracy of more than 40%. We suspect the low accuracy is caused by the lack of texture details in range images, as SR-GAN is proposed for photo-realistic images. We also encounter noisy predictions on irregular objects, such as trees and bushes. One cause of this problem is that the simulated environment is relatively simply structured. Though vegetation appear in simulation, they are only represented by simple geometries. Thus the training data gathered in the simulation possesses significantly less noise when compared with data from real-world environments. Another potential direction for future work may involve training with a combination of real and synthetic data.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported in part by the National Science Foundation, USA, grant number IIS-1723996.

Appendix A. Supplements for Section 4.1

We show detailed Octomaps produced by several methods introduced in Section 4.1. The scene shown in Fig. A.13 is located at the top corner of Fig. 5. A lidar mounted on top of a small unmanned ground vehicle (UGV), located in the center of the image, is used for capturing the data. The Octomaps for the entire environment are shown in Fig. 7. The resulting Octomaps of this region are shown in Fig. A.14. Note that, when using linear or cubic interpolation, the "ring" structure that covers the ground at the top-right corner differs greatly from the structure of (f). This is because these methods are interpolating over Euclidean space rather than the sensor's field of view. Naive interpolation methods are not able to retain the real range measurements characterizing the output of a real lidar. As is visible in (b) and (c), these methods also introduce erroneous range measurements by interpolating among the returns from distinctly different objects (such as the floor and above-ground objects). Our proposed deep learning method does not encounter this problem.

Appendix B. Supplements for Section 4.3

We give further detailed benchmarking results by comparing four metrics using different upscaling factors: (1) $\mathcal{L}1$ loss, which is the inference loss tested on the Ouster dataset using the network trained with simulated dataset from CARLA Town 02. (2) Removed pixels, which indicates the mean percentage of pixels deleted from the range images of each dataset. (3) Training time, which is the processing time for each range image during training on the simulated dataset from CARLA Town 02. (4) Testing time, which is the processing time for each range image during testing on the Ouster dataset.

Table B.2
Quantitative results for testing using Ouster dataset with different upscaling factors.

Upscaling factor	Method	$\mathcal{L}1$ loss	Removed pixels (%)	Training time per image (ms)	Testing time per image (ms)
8x (8 to 64)	Linear	0.0455	N/A	N/A	N/A
	Cubic	0.0595	N/A	N/A	N/A
	SR-ResNet	0.0320	30.70	21	7
	Ours	0.0318	15.71	18	6
4x (16 to 64)	Linear	0.0324	N/A	N/A	N/A
	Cubic	0.0467	N/A	N/A	N/A
	SR-ResNet	0.0211	17.70	25	7
	Ours	0.0214	8.37	18	6
2x (32 to 64)	Linear	0.0213	N/A	N/A	N/A
	Cubic	0.0346	N/A	N/A	N/A
	SR-ResNet	0.0118	8.92	29	9
	Ours	0.0117	2.38	17	6

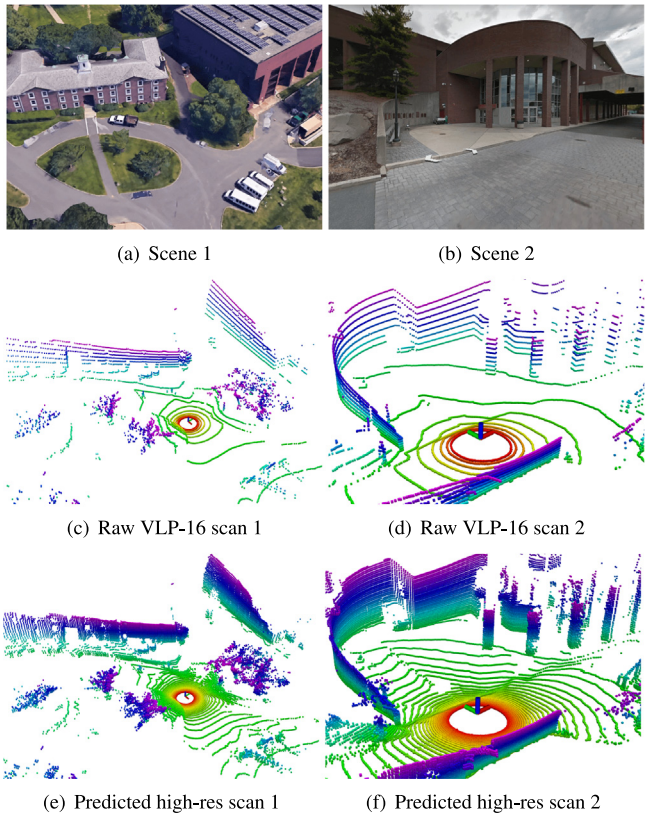


Fig. B.17. Lidar super-resolution using Velodyne VLP-16 data. The low-res point clouds shown in (c) and (d) are captured using a Velodyne VLP-16 lidar. The high-res point clouds shown in (e) and (f) are predicted by our approach (4× upscaling). Color variation indicates lidar "ring" index. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

As is shown in Table B.2, We have observed that the prediction accuracy degrades when less resolution is provided. Both SR-ResNet and our method achieve similar loss over different upscaling factors. However, the percentage of removed pixels of our method is much less when compared with the results of SR-ResNet. In other words, the predictions of our approach are of lower variance. Additionally, the proposed approach requires up to 45% less time to train the neural network. Note that all the results in the table are evaluated and averaged over 8825 scans.

We also note that the testing time of our proposed framework per image, which does not exceed 10 ms for any of the upscaling factors considered, is compatible with real-time performance over the lidars considered in this paper. Scanning at 10 Hz, these lidars would require the testing time not to exceed 100 ms, but our framework is well within this real-time performance envelope using the arrangement of hardware described in Section 4.

We also show qualitative results of performing 4× upscaling inference using our method on the Ouster dataset in Fig. B.15. The three representative point clouds are captured from a narrow street, an open intersection, and a slope surrounded by vegetation, respectively. We can observe that objects such as buildings, roads and pillars are inferred well.

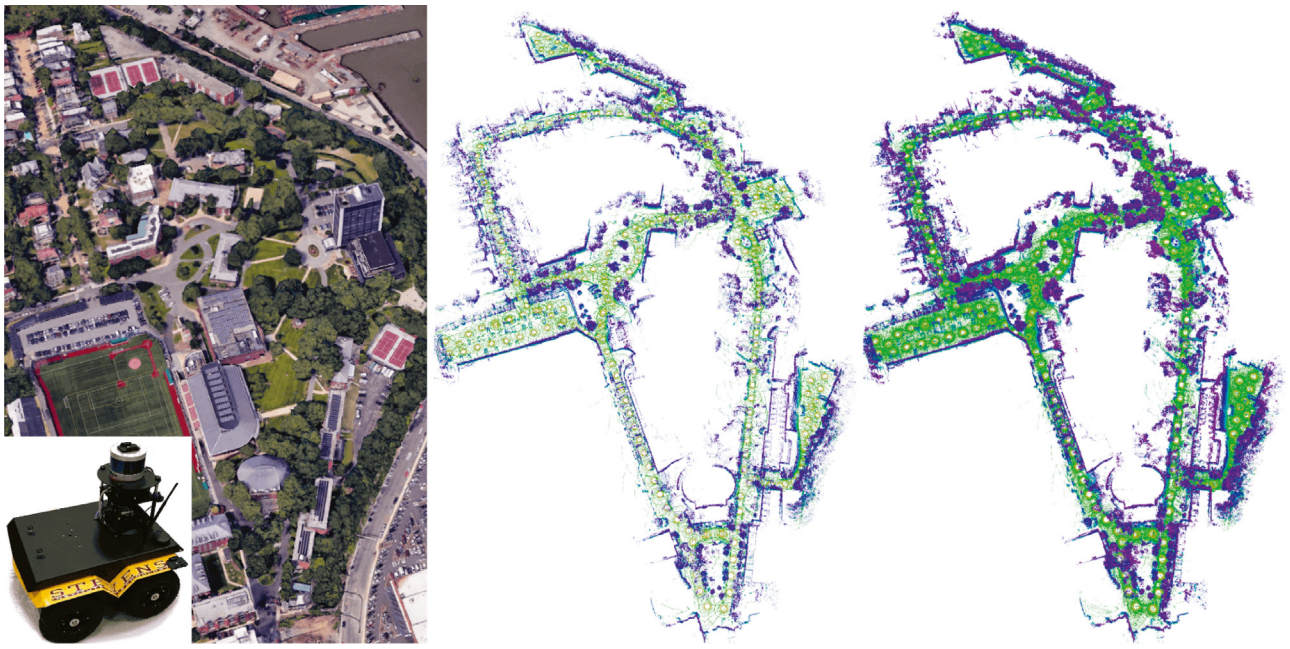
The corresponding range images of the projected point clouds are shown in Fig. B.16. Black color indicates a range value of zero, for which no points are added to the point cloud. Since performing convolutional operations on a range image will unavoidably cause smoothing effects on sharp and discontinuous object boundaries, we apply MC-dropout to identify these erroneous predictions. Accordingly, the range predictions with high variance are removed.

Appendix C. Stevens dataset

We show qualitative results from our own dataset, which we refer to as the Stevens dataset, which was collected using a Velodyne VLP-16 mounted on a Clearpath Jackal UGV on the Stevens Institute of Technology campus. This dataset features numerous buildings, trees, roads and sidewalks. Two representative scans from the dataset are shown in Fig. B.17. A satellite photo of the Stevens campus is shown in Fig. C.18(a). A total number of 278 scans, which are registered with LeGO-LOAM [37], are used for generating a 3D map. The neural network that is used for testing here is the same as the network that is used in Sections 4.1 and 4.2. Fig. C.18(b) and (c) show the 3D map created by the raw lidar scans and the inferred lidar scans (4× upscaling) respectively. The map produced by our inferred scans includes better structural and terrain coverage without using a real high-res lidar.

Appendix D. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2020.103647.



(a) Satellite image (b) Map using baseline approach (c) Map using our approach

Fig. C.18. Lidar super-resolution using Velodyne VLP-16 data. The low-res point clouds shown in (c) and (d) are captured using a Velodyne VLP-16 lidar. The high-res point clouds shown in (e) and (f) are predicted by our approach (4× upscaling). Color variation indicates lidar "ring" index. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

References

[1] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.

[2] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 1050–1059.

[3] R.M. Neal, Bayesian Learning for Neural Networks, Springer Science Business Media, 2012.

[4] W. Yang, X. Zhang, Y. Tian, W. Wang, J.-H. Xue, Q. Liao, Deep learning for single image super-resolution: A brief review, IEEE Trans. Multimedia 21 (12) (2019) 3106–3121.

[5] R. Keys, Cubic convolution interpolation for digital image processing, IEEE Trans. Acoust. Speech Signal Process. 29 (6) (1981) 1153–1160.

[6] C.E. Duchon, Lanczos filtering in one and two dimensions, J. Appl. Meteorol. 18 (8) (1979) 1016–1022.

[7] C. Dong, C.C. Loy, K. He, X. Tang, Learning a deep convolutional network for image super-resolution, in: European Conference on Computer Vision, 2014, pp. 184–199.

[8] J.B. Huang, A. Singh, N. Ahuja, Single image super-resolution from transformed self-exemplars, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5197–5206.

[9] J. Kim, J.K. Lee, K.M. Lee, Deeply-recursive convolutional network for image super-resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1637–1645.

[10] W. Shi, J. Caballero, F. Huszar, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874–1883.

[11] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, W. Shi, Photo-realistic single image super-resolution using a generative adversarial network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 105–114.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.

[13] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[14] Y. Zhang, T. Funkhouser, Deep depth completion of a single RGB-D image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 175–185.

[15] F. Ma, L. Carlone, U. Ayaz, S. Karaman, Sparse sensing for resource-constrained depth reconstruction, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016, pp. 96–103.

[16] J. Ku, A. Harakeh, S.L. Waslander, In defense of classical image processing: Fast depth completion on the CPU, in: Conference on Computer and Robot Vision, 2018, pp. 16–22.

[17] F. Ma, G.V. Cavalheiro, S. Karaman, Self-supervised sparse-to-dense: Self-supervised depth completion from LiDAR and monocular camera, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2019, pp. 3288–3295.

[18] V. Casser, S. Pirk, R. Mahjourian, A. Angelova, Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos, in: AAAI Conference on Artificial Intelligence, 2019.

[19] C. Godard, O.M. Aodha, G.J. Brostow, Unsupervised monocular depth estimation with left–right consistency, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6602–6611.

[20] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, N. Navab, Deeper depth prediction with fully convolutional residual networks, in: Proceedings of the IEEE International Conference on 3D Vision, 2016, pp. 239–248.

[21] N. Smolyanskiy, A. Kamenev, S. Birchfield, On the importance of stereo for accurate depth estimation: An Efficient semi-supervised deep neural network approach, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 1120–1128.

[22] L. Yu, X. Li, C. Fu, D. Cohen-Or, P. Heng, Pu-net: Point cloud upsampling network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2790–2799.

[23] C.R. Qi, L. Yi, H. Su, L.J. Guibas, Pointnet++: Deep hierarchical feature learning on point sets in a metric space, in: Advances in Neural Information Processing Systems, 2017, pp. 5099–5108.

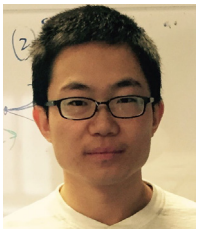
[24] M. Johnson-Roberson, C. Barto, R. Mehta, S.N. Sridhar, K. Rosaen, R. Vasudevan, Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks? Proceedings of the IEEE International Conference on Robotics and Automation, 2017, pp. 746–753.

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

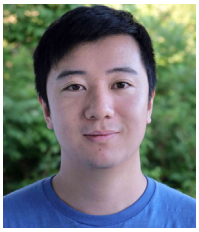
[26] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical Image Computing and Computer-Assisted Intervention, 2015, pp. 234–241.

[27] S. Ioffe, Christian Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the International Conference on Machine Learning Research, 2015, pp. 448-456.
[28] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the International Conference on Machine Learning, 2010, pp. 807–814.
[29] Y. Wang, W.L. Chao, D. Garg, B. Hariharan, M. Campbell, K. Weinberger, Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 8445-8453.
[30] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Proceedings of the 3rd International Conference on Learning Representations, 2015.
[31] F. Chollet, Keras: The python deep learning library, Astrophys. Source Code Libr. (2018).
[32] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, Tensorflow: a system for large-scale machine learning, in: USENIX Symposium on Operating Systems Design and Implementation, vol. 16 2016, pp. 265–283.
[33] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: An efficient probabilistic 3D mapping framework based on octrees, Auton. Robots 34 (3) (2013) 189–206.
[34] T. Shan, B. Englot, Belief roadmap search: Advances in optimal and efficient planning under uncertainty, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2017, pp. 5318–5325.
[35] F. Chen, S. Bai, T. Shan, B. Englot, Self-learning exploration and mapping for mobile robots via deep reinforcement learning, in: AIAA Scitech Forum, 2019, Paper AIAA-2019-0396.
[36] F. Chen, J. Wang, T. Shan, B. Englot, Autonomous exploration under uncertainty via graph convolutional networks, in: Proceedings of the 19th International Symposium on Robotics Research, 2019.
[37] T. Shan, B. Englot, LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2018, pp. 4758–4765.

Tixiao Shan received a Bachelor of Science in Mechanical Engineering and Automation from Qingdao University, Qingdao, China, in 2011, a Master's Degree in Mechatronic Engineering from Shanghai University, Shanghai, China, in 2014, and a Ph.D. in Mechanical Engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2019. He is currently a Postdoctoral Associate at the Massachusetts Institute of Technology, Cambridge, MA, USA, where he works jointly with the Computer Science and Artificial Intelligence Laboratory (CSAIL) and the Senseable City Lab, Department of Urban Studies and Planning.



Jinkun Wang received a Bachelor of Science in Mechanical Engineering from the University of Science and Technology of China, Hefei, China, in 2014, and M.Eng. and Ph.D. degrees in Mechanical Engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2016 and 2020, respectively. He is currently an Applied Scientist at Amazon Robotics in Boulder, CO, USA.



Fanfei Chen received a Bachelor of Engineering in Mechanical Design, Manufacturing and Automation from Tongji Zhejiang College, Jiaxing, China, in 2014, and a Master of Engineering in Mechanical Engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2016. He is currently a Ph.D. student in the Department of Mechanical Engineering, Stevens Institute of Technology.

Paul Szenher received a Bachelor of Engineering in Computer Engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2020. He is currently a Ph.D. student in the Department of Mechanical Engineering, Stevens Institute of Technology.

Brendan Englot received S.B., S.M., and Ph.D. degrees in Mechanical Engineering from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2007, 2009, and 2012, respectively. He was a research scientist with United Technologies Research Center, East Hartford, CT, USA, from 2012 to 2014. He is currently an Associate Professor with the Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, NJ, USA. His research interests include motion planning, localization, and mapping for mobile robots, learning-aided autonomous navigation, and marine robotics. He is the recipient of a 2017 National Science Foundation CAREER award and a 2020 Office of Naval Research Young Investigator award.