Connecting the Dots for Real-Time LiDAR-based Object Detection with YOLO

Benny Dai*, Cedric Le Gentil* and Teresa Vidal-Calleja
Centre for Autonomous Systems at the Faculty of Engineering and IT, University of Technology Sydney
{benny.dai, cedric.legentil}@student.uts.edu.au, teresa.vidalcalleja@uts.edu.au

Abstract

In this paper we introduce a generic method for people and vehicle detection using LiDAR data only, leveraging a pre-trained Convolutional Neural Network (CNN) from the RGB domain. Typically with machine learning algorithms, there is an inherent trade-off between the amount of training data available and the need for engineered features. The current state-of-the-art object detection and classification heavily rely on deep CNNs trained on enormous RGB image datasets. To take advantage of this inbuilt knowledge, we propose to fine-tune You only look once (YOLO) network transferring its understanding about object shapes to upsampled LiDAR images. Our method creates a dense depth/intensity map, which highlights object contours, from the 3D-point cloud of a LiDAR scan. The proposed method is hardware agnostic, hence can be used with any LiDAR data, independently on the number of channels or beams. Overall, the proposed pipeline exploits the notable similarity between upsampled LiDAR images and RGB images preventing the need to train a deep CNN from scratch. This *transfer learning* makes our method data efficient while avoiding the creation of heavily engineered features. Evaluation results show that our proposed LiDAR-only detection model has equivalent performance to its RGB-only counterpart.

1 Introduction

One of the largest efforts in robotics is the race to solve the perception challenges that are involved in the application of autonomous driving. In order to develop a robust object detection system for these autonomous systems, there is an emphasis to utilise multiple modalities to overcome the challenges that are encountered with an individual sensor. Currently, RGB image object detection systems are considered the most accurate, but it is prone to suffer from the inability to detect objects in low-light conditions - whereas LiDAR-based systems do not [Vaquero *et al.*, 2018].

In the past few years, there has been a significant amount of development in object detection using RGB images since [Krizhevsky *et al.*, 2012] demonstrated the efficacy of deep CNNs - whilst the adoption of deep learning methods has only been recently explored in the LiDAR domain. One of the main benefits of harnessing deep learning is that the network is capable of learning abstract and powerful feature representations - superseding and alleviating the resources that were typical in feature engineering [Liu *et al.*, 2018].

Some of the more recently proposed methods utilise LiDAR information to detect objects in the environment [Zhou and Tuzel, 2017] as a sole source of information, while other methods either fuse both domains or exploit depth information to reduce the search space [Berrio *et al.*, 2017; Qi *et al.*, 2016; Ku *et al.*, 2017]. In this paper, we propose a method for object detection system using only LiDAR scans.

The largest downside of incorporating deep CNNs is that it requires a significant amount of data in order to solve their 'task'. Most of the LiDAR-based methods, including ours, have incorporated the use of the KITTI Dataset [Geiger *et al.*, 2012] as part of the training pipeline, which contains a relatively dense labelled point cloud - through the use of a Velodyne HDL-64E Sensor and image annotations in the form of bounding boxes.

While the use of large labelled datasets can substantially assist deep learning training tasks, it is insufficient to train deep CNNs from scratch. It also lacks the capabilities of being able to be repurposed easily to both indoor and outdoor environments, which is a requirement for any deployed robots that traverse in urban environments.

*These authors contributed equally to this work

This work addresses these particular issues by utilising transfer learning, which reduces the amount of data and the time required to learn the model. To be specific, we adopt the use of pre-trained convolutional weights from ImageNet [Jia Deng *et al.*, 2009] that have been provided by [Redmon and Farhadi, 2016] in the RGB domain, and train for people and vehicle detections in the LiDAR domain. In essence, the convolutional neural network's capability to reason on shapes is transferred from visual images into LiDAR 'images', providing a generalised model that can perform in both indoor and outdoor environments.

To recover LiDAR 'images', we develop a fast algorithm that upsamples point cloud data to yield a dense representation that resembles closely to a typical RGB/visual image. Depth and LiDAR intensity are exploited to recover dense shapes. We focus - in particular - on extremely sparse LiDAR data such as the VLP-16, although the algorithm is general to be applied to LiDARs with denser information such as the 32 and 64 beams/channels.

In this paper, we address other related works, explain our methodology/pipeline and discuss our results.

2 Related work

Deep learning models that have been proposed in recent years, specifically LiDAR-based, can be categorised by the different input strategies utilised, such as

- voxel encoding of 3D-point cloud,
- fusion of LiDAR and RGB data,
- search space reduction through LiDAR depth cues to propose regions of interest,
- usage of raw LiDAR point clouds, and
- construction of dense maps from LiDAR data.

Note that a fixed-size input vector is generally required for deep learning architectures but 3D data does not necessarily contain a fixed number of 3D-points. Hence, various methods to address this issue have been recently proposed. The approach introduced in PointNet [Qi *et al.*, 2016] subsamples the input point cloud to fit a fixed-size input. PointNet also addresses the problem of data ordering by proposing an algorithm invariant regarding 3D-point permutations. Despite achieving highly accurate results classifying and segmenting small objects, it is limited when it comes to generalising to more complex scenes. In [Qi *et al.*, 2017a] and [Qi *et al.*, 2017b] the authors propose to enhance PointNet with respectively a multimodal approach called Frustum-PointNet, and a hierarchical network that recursively applies PointNet.

Another common approach to tackle the variable input size of 3D point clouds is the use of voxels. Methods like [Maturana and Scherer, 2015; Zhou and Tuzel, 2017; Berrio *et al.*, 2017] extend the convolutional neural network principles, originally designed for 2D images, to 3D data by discretising the space. These techniques possess the substantial advantage of not requiring any engineering features.

On the other hand, the live data provided by spinning lasers have a structure that can easily be exploited. Projecting LiDAR data into 2D-images is done in [Gonzalez *et al.*, 2015], where the authors propose a multi-view classifier based on random forests to detect pedestrians in urban scenarios. A similar goal is achieved in [Premebida *et al.*, 2014] using a support vector machine based on the same multimodal input. While these methods estimate a depth map from sparse LiDAR data, they still leverage colour images and do not use CNNs. Our method bears some similarities with [Li, 2017] through the use of deep CNNs. Although the concepts presented in [Li, 2017] are agnostic to any LiDAR sensor, using a different numbered channel device would require retraining of the deep CNN model. In our proposed method the type of upsampled LiDAR maps produced by the proposed approach can be adopted for sparse or denser LiDARs. Thus, a network trained with a given LiDAR, *e.g.* HDL-64, by using our upsampled method can be used to infer detections on other types of LiDAR data, *e.g.* VLP-16.

The work presented in [Asvadi *et al.*, 2018a] introduces a detection technique for vehicles using only lidar data after segmenting the incoming point cloud. This pre-processing is built assuming the sensor is operated in an autonomous driving context. Our framework does not rely on a heavily engineered process to prune LiDAR data but exploits the structure of the live 3D data in our upsampling process. We leverage the similarity between RGB images and LiDAR dense depth/intensity maps, to address the object detection task. In other words, our proposed pipeline aims at transferring the trained knowledge from a pre-trained deep CNNs using RGB images to another modality (laser range) in a data efficient manner.

The upsampling problem has been addressed in the literature under the name of depth completion. While some techniques, [Schneider *et al.*, 2016; Uhrig *et al.*, 2017], rely only on 3D data, other methods, [Ma *et al.*, 2018; Jaritz *et al.*, 2018], exploit both RGB and depth information. Regardless of the cues used, many techniques are built around deep CNNs. The authors in [Schneider *et al.*, 2016] leverage multi-cue information through an optimisation-based prediction using semantic labels as part of the input. While still relying on a deep neural network for the semantic cues in the RGB modality, this last method can reuse an already trained network.

A completely different approach is presented in [Ku *et al.*, 2018], where classical image processing is used to

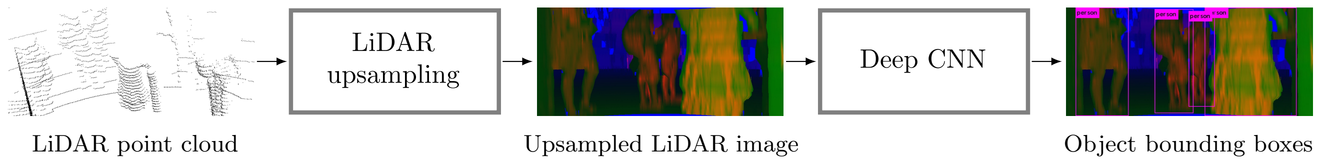


Figure 1: The pipeline of the proposed method. The input is a 360° laser scan (depth and intensity). The first step upsamples the lidar data into a dense depth/intensity image. This image is fed into a deep CNN based on YOLO9000. We transferred pre-trained weights from an RGB object detector by fine-tuning with LiDAR images.

predict dense depth maps from lidar data only. Similarly, the upsampling methods presented in [Asvadi *et al.*, 2018a], [Premebida *et al.*, 2014], and [Asvadi *et al.*, 2018b] do not rely on neural network architectures but on more traditional image processing. Even though these techniques provide dense depth maps, they suffer from blurriness at the objects' boundaries as the depth information is underexploited in the interpolation process. For example, in [Asvadi *et al.*, 2018a] the Delaunay triangulation is performed over lidar points projected in the camera coordinate system ignoring the difference of depth between projected points. Our method triangulates the 3D data directly in the 3D-space relying on concepts similar to the one exposed in [Moosmann *et al.*, 2009]. Our method differs from the neighbourhood graph in [Moosmann *et al.*, 2009] as we mesh the 3D-points with triangles after detecting potential object boundaries: in [Moosmann *et al.*, 2009] the links between points belonging to two consecutive lidar channels are almost 'vertical' whereas the triangles generated in our method can follow oblique object boundaries.

3 Overview

The proposed overall framework aims to produce image-like representations from LiDAR data as an input for a deep CNN. Initially, the sparse LiDAR data is converted into dense maps based on depth and laser intensity information. From there, the deep CNN takes the dense map input and propose bounding boxes on persons and vehicles if present in the scene. Fig. 1 illustrates the pipeline of this framework.

The choice of RGB deep CNN detection model that we considered is You only look once (YOLO) [Redmon and Farhadi, 2016]. As a state-of-the-art one stage detection model, it provides a mean of having fast and accurate detections, with extensive documentation to apply transfer learning easily. In addition, the DarkNet framework [Redmon, 2013 2016] is also written in C and CUDA - which can port directly as a real-time implementation on embedded systems.

The process and model architecture used here is identical to [Redmon and Farhadi, 2016] implementation of YOLO trained on the PASCAL dataset [Everingham *et al.*, 2015]. The only difference in architecture is we adjust the last convolutional layer's filters to conform to detecting persons and vehicles only. This is indicated in Table 1 which outlines the configuration that we have adopted [Redmon and Farhadi, 2016] to do fine-tuning for a two class detection problem. We incorporate the use of pre-trained weights from ImageNet [Jia Deng *et al.*, 2009] to classify RGB images, and train YOLO for detecting objects on LiDAR images.

At runtime, YOLO is used to predict detections on a 13 x 13 feature map and proposes an output of the type of class, confidence, and locations of bounding boxes.

4 Lidar upsampling

This section details the proposed upsampling procedure that is summarised in Algorithm 1. This algorithm generates a dense depth/intensity image $\mathfrak{I}$ from raw LiDAR data. Unlike many other algorithms, our algorithm does not aim at maximising the depth information coverage. Instead, we aim to highlight relevant shapes. The generated upsampled LiDAR images are made of three channels to fit the input of the RGB deep CNN used later in the proposed framework. Each of these channels can be independently associated with depth, inverse depth, or intensity information, not necessarily in that particular order.

4.1 Pre-processing

Let us consider an N-channel spinning LiDAR and a virtual camera. The camera is only used as a support to project LiDAR 3D-points into the image $\mathfrak{I}$. Each 3D-point collected by the LiDAR is associated with an intensity value that represents the return strength of the laser pulse that generated the point. The rotation matrix in between the two sensors is $\mathbf{R}_c$, and the camera matrix is $\mathbf{K}$. A 3D-point $\mathbf{x}_i^l$ expressed in the LiDAR frame is projected in the camera image with:

$$\begin{bmatrix} x_i^c \\ y_i^c \end{bmatrix} = \begin{bmatrix} \frac{x_i'}{z_i'} \\ \frac{y_i}{z_i'} \end{bmatrix} \text{ where } \begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \mathbf{K}\mathbf{R}_c\mathbf{x}_i^l. \quad (1)$$

An unordered 3D-point cloud $\mathcal{P}$ collected by the LiDAR is first sorted according to the points' azimuth and

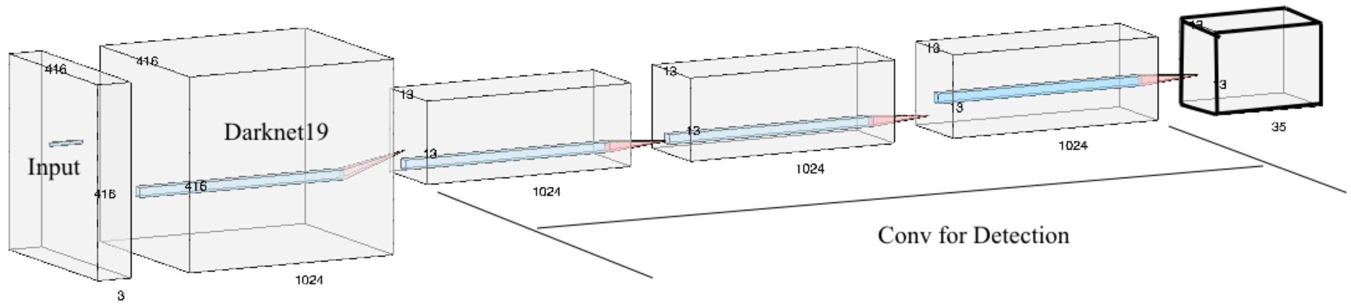


Figure 2: Overall Scheme of the YOLO model in our proposed LiDAR only detection. The last box in bold indicates a modified convolutional layer of 13 x 13 x **35** to solve for a two class problem. The DarkNet19 block represents all the layers that have been pre-trained on ImageNet.

```
Data: Lidar 3D-point cloud $\mathcal{P}$.
Result: Lidar depth and intensity dense image $\mathfrak{I}$.
$\mathfrak{I} \leftarrow$ Initialise output image according to pruned
  Voronoi partitioning;
$\mathcal{S} \leftarrow$ Sort $\mathcal{P}$ by increasing azimuth;
$\mathcal{C} \leftarrow$ Crop out points in $\mathcal{S}$ projected out of $\mathfrak{I}$ ;
$\mathcal{L}_k \leftarrow$ Split $\mathcal{C}$ into "lines";
$\mathcal{T} \leftarrow$ Initialise empty set of 3D-triangles;
foreach $\mathcal{L}_k$ do
    $\mathcal{B}_k \leftarrow$Extract border-point candidates in $\mathcal{L}_k$
end
foreach $\{\mathcal{L}_k, \mathcal{L}_{k+1}\}$ do
    $\mathcal{E} \leftarrow$ Associate points from $\mathcal{B}_k$ and $\mathcal{B}_{k+1}$ (knn);
    $\mathcal{E} \leftarrow$ Remove border-edges crossing each other when
      projected in $\mathfrak{I}$;
    $\mathcal{T}_{temp} \leftarrow$ Build triangles between two successive
      border-edges in $\mathcal{E}$;
    $\mathcal{T} \leftarrow \mathcal{T} + \mathcal{T}_{temp}$;
end
foreach triangle in $\mathcal{T}$ do
    $\mathfrak{I} \leftarrow$ Project triangle in image;
    $\mathfrak{I} \leftarrow$ Interpolate depth and intensity for pixels inside
      triangle projection (barycentric coordinates
      interpolation);
end Optional
Independent histogram equalisation of $\mathfrak{I}$'s channels;
```
Algorithm 1: LiDAR upsampling

stored in the set $\mathcal{S}$. The set $\mathcal{C}$ refers to the set of 3D-points from $\mathcal{S}$ that have their projection, as per (1), in the camera image boundaries. Points of $\mathcal{C}$ are grouped into 'lines' that correspond to the N LiDAR-channels. These lines are denoted $\mathcal{L}_k$ with $k = 1, \cdots, N$. The indices k are given according to the increasing elevation of the LiDAR-channels.

The output image $\mathfrak{I}$ is initialised with constant patches determined by a pruned Voronoi partitioning of the image containing the projected 3D-points. The pruning of the classic Voronoi cells is done by removing the patches that span over azimuth and elevation thresholds.

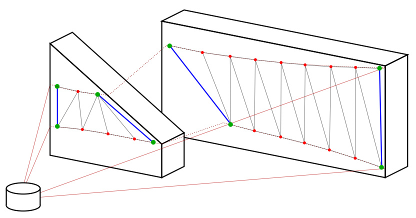


Figure 3: Triangulation of the 3D data between object borders between two consecutive LiDAR channels. The mesh generated is shown in grey, the border-points in green, the border-edges in blue and the LiDAR points in red (the dashed red line represent the places swept by the lasers).

4.2 Border detection

Potential object border-points are detected in each $\mathcal{L}_k$ using distance and azimuth thresholds; i.e. if two consecutive 3D-points are spaced over a certain Euclidean distance, or over a certain difference of azimuth, they are considered as border-points candidates and appended to a set $\mathcal{B}_k$.

Border-point candidates from two consecutive LiDAR lines, $\mathcal{B}_k$ and $\mathcal{B}_{k+1}$, are associated into pairs based on a nearest-neighbour search. We call these pairs border-edges. Points in $\mathcal{B}_k$ and $\mathcal{B}_{k+1}$ need to mutually be the nearest neighbour of one another to be considered as a potential edge. A threshold on the potential border-edge length is applied. The border-edge candidates are temporarily grouped in $\mathcal{E}$. If two edges in $\mathcal{E}$ cross each other when projected in $\mathfrak{I}$, the edge the furthest away from the LiDAR is removed from $\mathcal{E}$.

4.3 Triangulation

Once the edges are found, we perform a simple triangulation over the point clouds constituted of the 3D-points, from $\mathcal{L}_k$ and $\mathcal{L}_{k+1}$, contained between successive pairs

	Type	Filters	Size/Stride	Output
0	Convolutional	32	3 x 3	416 x 416
1	Maxpool		2 x 2/2	208 x 208
2	Convolutional	64	3 x 3	208 x 208
3	MaxPool		2 x 2/2	104 x 104
4	Convolutional	128	3 x 3	104 x 104
5	Convolutional	64	1 x 1	104 x 104
6	Convolutional	128	3 x 3	104 x 104
7	MaxPool		2 x 2/2	52 x 52
8	Convolutional	256	3 x 3	52 x 52
9	Convolutional	128	1 x 1	52 x 52
10	Convolutional	256	3 x 3	52 x 52
11	MaxPool		2 x 2/2	26 x 26
12	Convolutional	512	3 x 3	26 x 26
13	Convolutional	256	1 x 1	26 x 26
14	Convolutional	512	3 x 3	26 x 26
15	Convolutional	256	1 x 1	26 x 26
16	Convolutional	512	3 x 3	26 x 26
17	MaxPool		2 x 2/2	13 x 13
18	Convolutional	1024	3 x 3	13 x 13
19	Convolutional	512	1 x 1	13 x 13
20	Convolutional	1024	3 x 3	13 x 13
21	Convolutional	512	1 x 1	13 x 13
22	Convolutional	1024	3 x 3	13 x 13
23	Convolutional	1024	3 x 3	13 x 13
24	Convolutional	1024	3 x 3	13 x 13
25	Route from Layer 16			
26	Reorg		/ 2	13 x 13
27	Route from Layer 24 and 26			
28	Convolutional	1024	3 x 3	13 x 13
29	**Convolutional**	**35**	1 x 1	13 x 13

Table 1: Modified version of YOLO's network architecture, with the last convoluttional layer's filters reduced to fit to our two class problem.

of edges. Fig. 3 illustrates this meshing. Some basic thresholds are applied to set a maximum triangle edge length. This process of edge detection and triangulation is repeated for each pair of consecutive LiDAR lines.

Once the 3D-data has been meshed, each of the triangles is projected in $\mathfrak{I}$. The interpolated values for pixels contained into a triangle consists into a weighted average of the three vertices' depth or intensity value. Barycentric coordinates are used to define the weights. Each of the three channels of the output image can be independently set with depth, inverse depth or intensity information. Our implementation also comprises an optional histogram equalisation for each of the image channels.

Fig. 4 shows examples of upsampled images. On the left of Fig. 4 d), we see that the top of the chair has not meshed with the background, there is a blank space in the mesh. This behaviour results from our strategy to conserve sharp object edges. We believe that sharp edges are more suitable for the rest of the detection pipeline as per the small size of the CNN filters. The pruned Voronoi partitioning present in the pre-processing step can provide data in this area depending on the thresholds used. The Voronoi partitioning provides sharper edges than simply interpolating between LiDAR lines. Note that the rotation matrix $\mathbf{R}_c$ can be modified to produce several images that will cover the full LiDAR field of view.

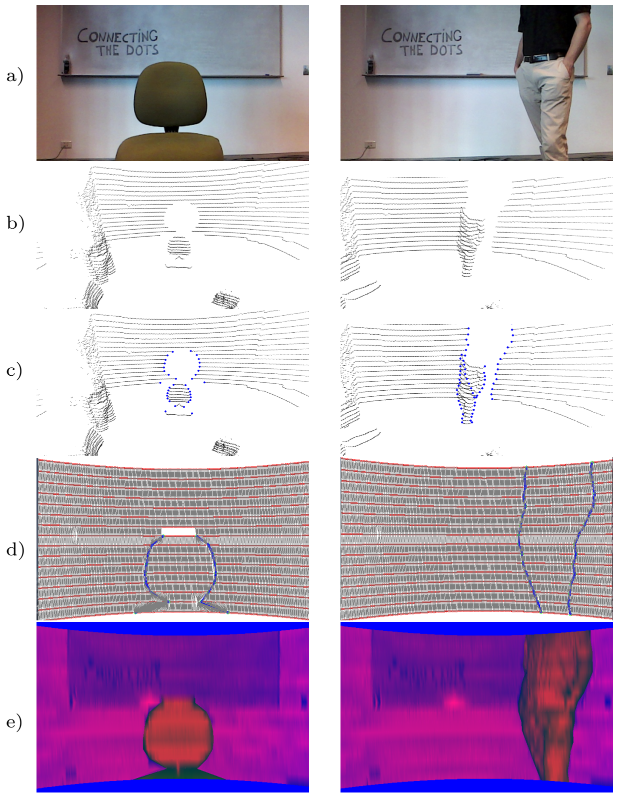


Figure 4: Upsampled LiDAR images based on real data from Velodyne VLP-16. The row a) is an actual colour snapshot of the scene (that is not used in the upsampling process). The row b) represents the 3D-point clouds used at the input of the upsampling process. The row c) displays in blue the border-points extracted from the input point cloud. The row d) shows in grey the projection of the mesh generated, in blue the border-edges and in red the LiDAR points. Finally e) display the upsampled LiDAR image. In this example, channel 0 is associated with intensity, channel 1 with inverse depth, and channel 2 with depth (for visualisation: channel 0 = red, channel 1 = green, channel 2 = blue).

5 Training details

As mentioned earlier, the proposed method leverages similarities between upsampled LiDAR images and RGB images. Based on the weights pre-trained on Imagenet [Jia Deng *et al.*, 2009] for image object classification - provided by [Redmon and Farhadi, 2016]) - we fine-tuned YOLO with two LiDAR datasets. The first one is the KITTI 2D Object Training dataset [Geiger *et al.*, 2012]. The second one has been collected with our own Velodyne VLP-16 in a combined indoors and outdoors environment at UTS.

The proposed upsampling method allows for different combinations of parameters. Among these parameters is the choice of the information allocated to each channel of the upsampled LiDAR image. After a qualitative analysis of several parameter sets, we opted for the following configuration that converts LiDAR information into dense images without histogram equalisation:

- channel 0 (R): LiDAR intensity values
- channel 1 (G): inverse depth values
- channel 2 (B): depth values

The LiDAR images of the KITTI dataset have been generated, as per the given calibration parameters, to overlap the actual camera images that contain the object labels. The Velodyne HDL-64 field of view being oriented 'toward the floor', the top part of the upsample LiDAR images does not contain any information. Therefore these upper parts have been cropped out. To preserve a similar resolution across all images whilst training - without making alterations on the network - we vertically split the cropped LiDAR images into three chunks. By adopting this, we increase the KITTI dataset [Geiger *et al.*, 2012] by three-fold with true negatives in it.

The given object labels from the RGB domain have been transferred to the corresponding upsampled LiDAR images according to the different cropping/splitting operations. This eventuated a customised KITTI dataset [Geiger *et al.*, 2012] that contains 22443 images containing 5211 persons and 23251 vehicles. We considered the person class from the KITTI labels' fields 'Cyclist', and 'Pedestrian'; while vehicles class consists of fields 'Car', 'Van', and 'Truck'. In the case of our VLP-16 dataset, we manually annotated 3002 images counting 4550 persons and 895 vehicles.

We incorporated multi-scale training like [Redmon and Farhadi, 2016], to ensure robust detections of any image/object size. From there, we combined both datasets, shuffled it, and formed a 70/20/10 split which corresponds to training, testing, and validation datasets respectively. The model has been trained for detections over 150 epochs with a batch size of 128. The starting learning rate was 10^{-3}. We adopted a strategy similar to [Redmon and Farhadi, 2016] by using a weight decay of 0.0005, a momentum of 0.9, and divide the learning rate after the 1st and 100th epoch. We choose the best model based on the cross-fold validation score.

Furthermore, we employ a subdivision size of 16 in Darknet [Redmon, 2013 2016] during training, which utilises between 4-6 GB of GPU at a given time depending on the scaling of the image. Since we have similar resolutions across our training set, the fluctuations in GPU usage are much more predictable. We employ the use of the UTS Cluster during training which has a Quadro P4000 that holds 8GB VRAM.

6 Experimental Results

Our fine-tuned model has been evaluated based on the Intersect over Union (IoU) and mean Average Precision (mAP) metrics. IoU accounts for the correctness of the predicted bounding boxes' locations and mAP, reflects the precision of the classification. To benchmark our method, we also fine-tuned the same CNN architecture for RGB images only. This last model has been trained from the PASCAL dataset [Everingham *et al.*, 2015] and the KITTI 2D Object Training dataset [Geiger *et al.*, 2012]. Images from KITTI have been vertically split similarly to the LiDAR images at the CNN input. Table 2 shows quantitative performances for both RGB-only and LiDAR-only models. Fig. 5 presents the precision-recall curves per class.

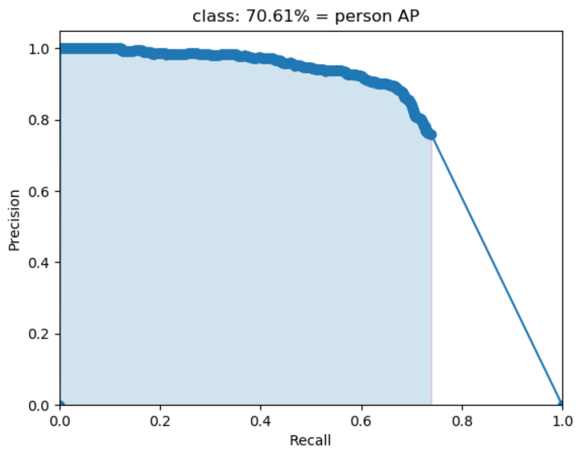
Metric	RGB	**LiDAR**
mAP	80.67 %	**81.27** %
Average IoU	63.15 %	**65.11** %
F1-Score	0.82	**0.82**

Table 2: Quantitative comparison between a RGB-only object detector and the proposed LiDAR-only detector. Numbers are similar for both models proving the ability for pre-trained neural network architectures to adapt to new modalities in a data efficient manner.

The numerical results of both models are quite similar. It demonstrates the possibility to apply transfer learning in a data efficient manner while conserving the original network performances. While displaying solid classification abilities with high mAP scores, one could argue that the average IoU scores could be improved. It is partly explained by our splitting technique that we applied to conserve resolution and increase number of training images in the KITTI dataset.

Fig. 6 demonstrates a wrongly fitted bounding box after a vertical split. The truck label on the bottom right image should be resized to provide more accurate training data. This phenomenon impacts both the RGB and LiDAR models.

Qualitative results are shown in Fig. 7 and the attached video. As per the low vertical resolution of the Velodyne VLP-16, people and vehicles far away from the LiDAR cannot be detected as there is not enough information present in the collected 3D-point clouds. It explains why the person across the street in the first row of Fig. 7 is not labelled as human.



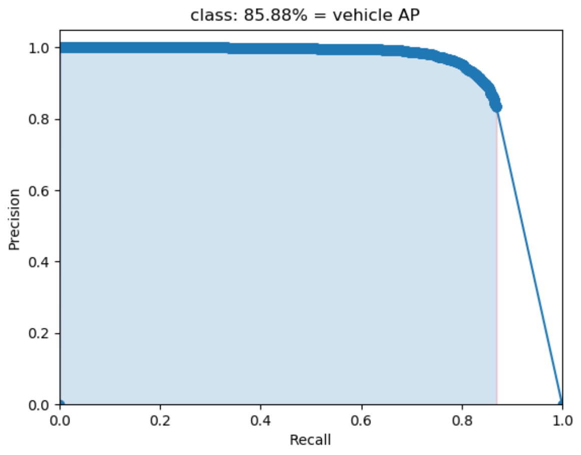


Figure 5: Precision-Recall curves per class computed with our proposed LiDAR-only detection system on our validation set. Area under the curve demonstrates the classification performances of our model.

For the real-time implementation, we used a Jetson TX2 and encapsulated our working code into ROS nodes. Employing this hardware, we achieve consistently 6 frames per second during runtime. To encapsulate DarkNet [Redmon, 2013 2016], we used [Bjelonic, 2016 2018] repository which makes the CNN pipeline as a ROS node.

7 Conclusion

This paper presents a pipeline for real-time people and vehicle detection from LiDAR data only. This method relies on a LiDAR upsampling technique that generates dense depth/intensity images out of raw and sparse laser data independently of any training dataset. We leveraged the similarity between these upsampled LiDAR images and RGB images by performing transfer learning from a state-of-the-art deep CNN trained in the RGB domain [Redmon and Farhadi, 2016]. Consequently, we demonstrated the possibility to fine-tune a pre-trained network to another modality in a data-efficient manner. Intuitively, it shows the understanding of shapes that is learnt in deep CNNs.

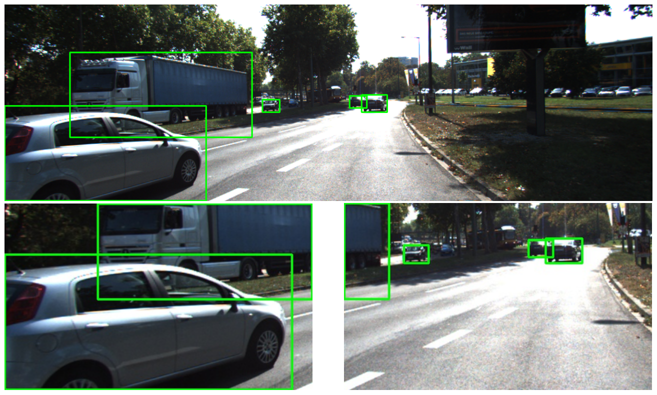


Figure 6: Example of image splitting in the KITTI dataset. Top row shows the original image with its corresponding labels. Bottom row displays two images and their new labels after splitting the original image. We can see that the bounding box at the back of the truck in the right image is located with poor precision.

Our upsampling method has been built upon purely geometric and traditional image processing principles exploiting the data pattern of spinning lasers. Therefore, this method can be used with LiDARs that possess a different number of channels without the need for any re-training step. The current implementation runs on a single CPU and processes VLP-16 scans real-time. The code can easily be parallelised: the border-edge detection can be run into $N-1$ independent threads and the interpolation in each of the triangles can be spread across thousands of GPU cores. Combined with the real-time performances of [Redmon and Farhadi, 2016], the proposed pipeline could be simultaneously run with different virtual camera orientations to cover the full LiDAR field-of-view.

In our experiments, the proposed upsampling method used an arbitrarily chosen set of parameters. We are interested in conducting a deeper analysis of these parameters. Maximising some similarity metrics between upsampled LiDAR images and the corresponding RGB snapshots could lead to properly justified parameter choices. Such work would make the transfer learning from RGB-trained networks even more efficient.

References

[Asvadi *et al.*, 2018a] Alireza Asvadi, Luis Garrote, Cristiano Premebida, Paulo Peixoto, and Urbano J. Nunes. DepthCN: Vehicle detection using 3D-LIDAR

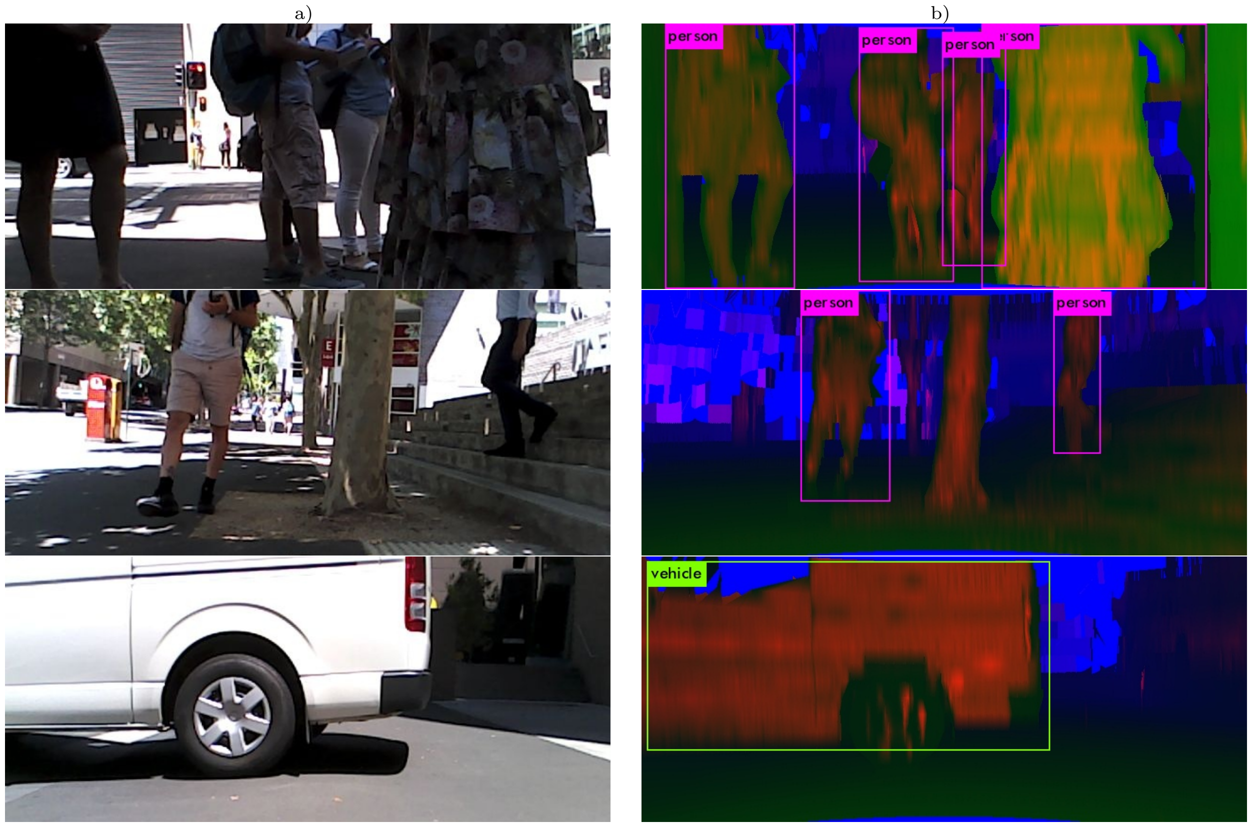


Figure 7: Examples of detection with Velodyne VLP-16. The column a) is a colour snapshot of the scene. These images are not used in the proposed method. The column b) shows the prediction of our framework. The bounding boxes are inferred over upsampled LiDAR images.

and ConvNet. *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2018-March:1–6, 2018.

[Asvadi *et al.*, 2018b] Alireza Asvadi, Luis Garrote, Cristiano Premebida, Paulo Peixoto, and Urbano Nunes J. Real-Time Deep ConvNet-Based Vehicle Detection Using 3D-LIDAR Reflection Intensity Data. In *ROBOT 2017: Third Iberian Robotics Conference*, pages 475—-486. Springer International Publishing, 2018.

[Berrio *et al.*, 2017] Julie Stephany Berrio, James Ward, Stewart Worrall, Wei Zhou, and Eduardo Nebot. Fusing Lidar and Semantic Image Information in Octree Maps. In *Australasian Conference on Robotics and Automation 2017*, 2017.

[Bjelonic, 2016 2018] Marko Bjelonic. YOLO ROS: Real-time object detection for ROS. `https://github.com/leggedrobotics/darknet_ros`, 2016–2018.

[Everingham *et al.*, 2015] Mark Everingham, S M Ali Eslami, Luc Van Gool, Christopher K I Williams, John Winn, Andrew Zisserman, M Everingham, S M A Eslami, J Winn, L KU Van Gool Leuven, Belgium L Van Gool ETH, Switzerland C K I Williams, and A Zisserman. The PASCAL Visual Object Classes Challenge: A Retrospective. *Int J Comput Vis*, 111:98–136, 2015.

[Geiger *et al.*, 2012] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[Gonzalez *et al.*, 2015] Alejandro Gonzalez, Antonio M López, and Alejandro Gonz. Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection Multiview Random Forest of Local Experts Combining RGB and LIDAR data for Pedestrian Detection. (October 2016), 2015.

[Jaritz *et al.*, 2018] Maximilian Jaritz, Raoul de Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and Dense Data with CNNs: Depth Completion and Semantic Segmentation. 2018.

[Jia Deng *et al.*, 2009] Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Hinton Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pages 1–9, 2012.

[Ku *et al.*, 2017] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. dec 2017.

[Ku *et al.*, 2018] Jason Ku, Ali Harakeh, and Steven L. Waslander. In Defense of Classical Image Processing: Fast Depth Completion on the CPU. 2018.

[Li, 2017] Bo Li. 3D fully convolutional network for vehicle detection in point cloud. In *IROS*, pages 1513–1518, 2017.

[Liu *et al.*, 2018] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep Learning for Generic Object Detection: A Survey. 2018.

[Ma *et al.*, 2018] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised Sparse-to-Dense: Self-supervised Depth Completion from LiDAR and Monocular Camera. 2018.

[Maturana and Scherer, 2015] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, sep 2015.

[Moosmann *et al.*, 2009] Frank Moosmann, Oliver Pink, and Christoph Stiller. Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. *IEEE Intelligent Vehicles Symposium, Proceedings*, pages 215–220, 2009.

[Premebida *et al.*, 2014] Cristiano Premebida, João Carreira, Jorge Batista, and Urbano Nunes. Pedestrian detection combining RGB and dense LIDAR data. In *IEEE International Conference on Intelligent Robots and Systems*, 2014.

[Qi *et al.*, 2016] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 601–610, dec 2016.

[Qi *et al.*, 2017a] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. 2017.

[Qi *et al.*, 2017b] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. 2017.

[Redmon and Farhadi, 2016] Joseph Redmon and Ali Farhadi. YOLO9000: Better, Faster, Stronger. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, dec 2016.

[Redmon, 2013 2016] Joseph Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[Schneider *et al.*, 2016] Nick Schneider, Lukas Schneider, Peter Pinggera, Uwe Franke, Marc Pollefeys, and Christoph Stiller. Semantically guided depth upsampling. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9796 LNCS:37–48, 2016.

[Uhrig *et al.*, 2017] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity Invariant CNNs. *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017*, pages 11–20, 2017.

[Vaquero *et al.*, 2018] Victor Vaquero, Alberto Sanfeliu, and Francesc Moreno-Noguer. Deep Lidar CNN to Understand the Dynamics of Moving Vehicles. *ICRA*, 2018.

[Zhou and Tuzel, 2017] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *arXiv:1711.06396 [cs.CV]*, 2017.