Robotics and Autonomous Systems 136 (2021) 103701

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

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

On deep learning techniques to boost monocular depth estimation for autonomous navigation

Raul de Queiroz Mendes [*], Eduardo Godinho Ribeiro, Nicolas dos Santos Rosa, Valdir Grassi Jr.

Department of Computer and Electrical Engineering, São Carlos School of Engineering, University of São Paulo, Brazil

ARTICLE INFO

Article history:
Received 13 October 2020
Accepted 23 November 2020
Available online 21 December 2020

Keywords:
SIDE
CNN
Deep learning

ABSTRACT

Inferring the depth of images is a fundamental inverse problem within the field of Computer Vision since depth information is obtained through 2D images, which can be generated from infinite possibilities of observed real scenes. Benefiting from the progress of Convolutional Neural Networks (CNNs) to explore structural features and spatial image information, Single Image Depth Estimation (SIDE) is often highlighted in scopes of scientific and technological innovation, as this concept provides advantages related to its low implementation cost and robustness to environmental conditions. In the context of autonomous vehicles, state-of-the-art CNNs optimize the SIDE task by producing high-quality depth maps, which are essential during the autonomous navigation process in different locations. However, such networks are usually supervised by sparse and noisy depth data, from Light Detection and Ranging (LiDAR) laser scans, and are carried out at high computational cost, requiring high-performance Graphic Processing Units (GPUs). Therefore, we propose a new lightweight and fast supervised CNN architecture combined with novel feature extraction models which are designed for real-world autonomous navigation. We also introduce an efficient surface normals module, jointly with a simple geometric 2.5D loss function, to solve SIDE problems. We also innovate by incorporating multiple Deep Learning techniques, such as the use of densification algorithms and additional semantic, surface normals and depth information to train our framework. The method introduced in this work focuses on robotic applications in indoor and outdoor environments and its results are evaluated on the competitive and publicly available NYU Depth V2 and KITTI Depth datasets.

1. Introduction

Autonomous navigation technologies are increasingly present in the academy, industries, agriculture and, to a lesser extent, on the streets and homes to aid in everyday tasks. However, many challenges still need to be solved to make autonomous vehicles a reality for the population. One of these challenges involves the improvement of perception mechanisms that integrate the robotic platforms of the autonomous system. These mechanisms are responsible for mapping the environment in which the vehicle is inserted, helping it to comprehend the surrounding 3D space [1].

In order to understand the 3D space, the perception system must be able to estimate the distance in which the objects are positioned in the scene. From this, the autonomous vehicle may avoid obstacles and perform safer traffic on navigable surfaces of different scenarios such as cities and roads. Therefore, it is essential that robotic platforms have the ability to explore depth cues of the environment, which can also benefit other Computer Vision tasks such as plane estimation [2], occluding contours estimation [3], object detection [4], Visual Odometry (VO) [5] and Simultaneous Localization and Mapping (SLAM) [6].

RGBD sensors, Light Detection and Ranging (LiDAR), stereo cameras, RADAR and SONAR are widely commercialized technologies for depth sensing. However, there are limitations regarding the use of these technologies for depth perception applications. LiDAR sensors are cost-prohibitive and perform sparse and noisy depth measurements over long distances, Kinect sensors are sensitive to light and have a lower distance measurement limit when compared to LiDAR, stereo cameras fail in reflective and low texture regions whereas RADAR and SONAR are generally used as auxiliaries.

Thus, the use of monocular cameras, in the context of robotic perception, becomes advantageous, once they demand a smaller installation space, are low cost, capture RGB images, are robust to environmental conditions, are efficient in terms of energy consumption and are a widespread technology [7].

[*] Corresponding author.
E-mail addresses: raulmendes@usp.br (R. de Queiroz Mendes), eduardogr@usp.br (E.G. Ribeiro), nicolas.rosa@usp.br (N. dos Santos Rosa), vgrassi@usp.br (V. Grassi Jr.).

https://doi.org/10.1016/j.robot.2020.103701
0921-8890/© 2020 Elsevier B.V. All rights reserved.

Recently, the advance of Convolutional Neural Networks (CNNs), in addition to the availability of databases built with structured-light and LiDAR sensors, motivated the emergence of new works in the Single Image Depth Estimation (SIDE) area. However, state-of-the-art SIDE approaches use sparse and noisy ground truth to train networks and do not fully benefit from how depth, semantics and surface normals can be added to improve the performance of the method regarding speed and accuracy. Moreover, deep models applied to the SIDE task still present significant estimation errors which indicate that the area is open to new solutions.

In this work, we aim to introduce a Fully Convolutional Network (FCN), with an encoder–decoder structure, to tackle single-view depth estimation problems in robotic applications. For the training of this network, we centered on developing a simple 2.5D loss function that focuses on geometric cues of the image. Such a loss is used in conjunction with a novel module that estimates surface normals based on the depth prediction step.

At the same time, this work also shows how some Deep Learning methods may be applied to improve the results of our framework with respect to the quality and accuracy of the predictions. Among such methods, we employ: depth completion [8,9], semantic segmentation [10], surface normals estimation [11], Atrous Spatial Pyramid Pooling (ASPP) [12], skip-connections [13], multi-scale models [14] and multi-layered deconvolutions [15].

The developed pipeline is tested and evaluated indoors using the NYU Depth V2 dataset [16] and outdoors using the KITTI Depth dataset [17]. Moreover, our work provides comprehensive surveys over the monocular depth estimation and depth completion areas and we compare the proposed method with other approaches in the SIDE literature with the aid of metrics that are widely disseminated.

The main contributions of this paper are:

- Extensive and detailed surveys on monocular depth estimation and depth completion;
- A simple CNN architecture, along with four feature extraction models, that can be applied for both SIDE and correlated tasks at a high frame rate;
- A lightweight module that is capable of estimating surface normals with state-of-the-art accuracy;
- A practical 2.5D loss function that is employed in the depth and surface normals experiments;
- Ablation studies considering variations in loss functions, network structures, fine-tuning techniques and additional training information;
- Application of different indoor and outdoor datasets and evaluation on the publicly available and competitive KITTI Depth [17] and NYU Depth V2 [18].

2. Depth estimation: A brief survey

There is a considerable amount of classic methods in the literature that tackle monocular depth inference problems [19–21]. However, with the development of more complex and efficient deep CNNs [22–25], several recent works started to exploit such networks, besides graphic models, to extract dense feature maps from RGB images in SIDE applications.

For the presented task, typical CNN models have a fully convolutional architecture with an encoder–decoder structure. The encoder extracts feature maps from the input and the decoder retrieves full resolution images. The training process is usually performed by minimizing a regression loss function such as the Mean Squared Error (MSE) [26] and the Mean Absolute Error (MAE) [27]. Regarding the type of learning, SIDE models can be trained in three different manners: supervised, semi-supervised and self-supervised.

2.1. Supervised learning

This type of SIDE approach uses sparse and noisy reference depth maps, constructed through point clouds from laser scans, to train supervised deep CNNs.

Eigen et al. [28] presented a fundamental work in which the developed architecture is composed of two stacks. The first stack estimates the scene depth from a global point of view and the second stack performs local refinements. The authors also introduced a scale-invariant loss function (SILog) that highlights the depth relationships of the image pixels, instead of focusing on the general scale.

With another fundamental work, Laina et al. [29] introduced an FCN composed of up-projection and residual learning mechanisms [30] to predict accurate depth maps in a faster way. The authors also presented the benefits of using the Huber reverse loss function (BerHu) [31,32] to train SIDE networks.

Exploring the benefits of graphic models, Liu et al. [33] combined a CNN with a continuous Conditional Random Field (CRF) and proposed a new superpixel pooling method that addresses the problem of information loss after successive downsampling operations. Cao et al. [34] used classification methods, as opposed to common regression techniques, through the discretization of the ground truth (depth bins). Their results show that SIDE by classification can surpass the conventional regression. However, the accuracy depends on the number of depth bins since the loss function is not able to differentiate depth values within a bin.

Inspired by this idea, Liao et al. [35] employed a regression loss combined with a classification function to improve accuracy. The authors also introduced a CNN that outputs depth maps from RGB images concatenated with reference depth data from 2D laser scans.

Using different approaches, Xu et al. [14] proposed a network structure that integrates a CNN with a fusion module composed of continuous CRFs. Fu et al. [26] developed an ordinal regression method based on a spacing-increasing discretization technique. Exploiting transfer learning strategies, Alhashim et al. [15] employed the DenseNet-169 [22] model pretrained on the ILSVRC [36] in a new FCN architecture.

Recently, Lee et al. [13] introduced a deep network architecture that applies the DenseNet-161 [22] and the ResNext-101 [24] as encoder. Coupled with the encoder structure, the authors used a dense multi-scale feature learning module (Dense ASPP), followed by a decoder with convolution and upconvolution blocks, downsampling and concatenation operations and Local Planar Guidance layers (LPG).

Exploring temporal features, Mancini et al. [1] introduced Fully Connected Long Short-Term Memory (FC-LSTM) [37] layers in a CNN to address the monocular depth prediction task. However, to use image sequences as input to an FC-LSTM, they need to be transformed into 1D vectors, which impairs the network's learning of spatial relationships. Conv-LSTM [38], on the other hand, is a type of neural layer that allows the extraction of spatial and temporal features. This layer is employed in the works of Kumar et al. [39] and Wang et al. [40].

Applying attention-based techniques, which are adaptive and allow only features that have meaningful information to be focused on during training, Xu et al. [41] proposed a multi-scale CNN composed of an attention-guided CRF module to perform single-view depth prediction. Following a resembling idea, Chen et al. [42] introduced an attention-based context aggregation CNN that aggregates contextual information at pixel and global levels from the feature maps.

2.1.1. SIDE and semantic segmentation

Semantic segmentation and monocular depth estimation are fundamental and correlated tasks in Computer Vision, since, from them, it is possible to infer scene geometry, artifacts scale and location, the scene's structure and the distances at which objects are located [43]. Moreover, the aforementioned tasks benefit mutually, as understanding the depth of the scene helps semantic segmentation during categorization, especially for the case when there are different artifacts with similar structures and at different depths [44]. On the other hand, semantic labels provide geometric and perspective information that assists the depth estimation of each object in the scene, improving mainly the predictions at high gradient regions (borders).

Classic approaches already employ methods addressing semantic segmentation to estimate depth [45,46] and RGBD images to obtain precise semantic classes [18,47]. Using CNNs, Eigen et al. [48] introduced a deep multi-scale network, which, with minor modifications, can predict depth, surface normals and semantic maps. Benefiting from graphic models, Mousavian et al. [49] developed a jointly CNN and CRF to estimate depth and semantics. Wang et al. [44] also tackled SIDE and semantics simultaneously using two CNNs and a Hierarchical CRF. Going further, Jiao et al. [43] achieved impressive accuracy approaching both tasks with a new CNN, whose training is based on minimizing an attention-guided loss function.

2.1.2. SIDE and surface normals estimation

The first studies on 3D understanding sought to obtain, among other information, illumination changes and color intensities [50] and volumetric shapes [51]. Due to their limitations, later works focused on other types of geometric cues such as segments and vanishing points [20] and super-pixels [52]. However, these methods depend on volumetric relationships and structural constraints. With the aid of datasets generated by RGBD sensors, later methods sought to estimate the 2.5D layout of scenes through depth and surface normals estimations, which are highly correlated. Some of these works were developed by Silberman et al. [18] and Fouhay et al. [53,54].

In the deep learning scenario, Wang et al. [2] introduced a framework that estimates depth and surface normals through a four-stream CNN, coupled with a dense CRF, which retrieves planar surfaces and edges, as well as partial depth and surface normals. Also in a multitask context, Qi et al. [55] created a network with two branches, called normal-to-depth and depth-to-normal, for depth and surface normals estimations. Based on the affinity rate between pixel pairs produced in correlated tasks, Zhang et al. [56] proposed a CNN that explores similarity patterns between corresponding pixels, through an affinity matrix, to generate depth, semantics and surface normals maps simultaneously.

Rather than using multi-scale decoding structures, Yin et al. [11] developed a CNN that performs SIDE and benefits from 3D geometric features of the scene through virtual normals. From another perspective, Ramamonjisoa et al. [3] introduced the SharpNet, which predicts depth, surface normals and occluding contours, as well as a loss function that forces consistency between depth and occluding contours and between depth and surface normals.

2.2. Self-supervised learning

With the premise that supervised methods demand a large amount of ground truth depth data, which require significant time and effort to be produced, self-supervised learning strategies began to be developed. The emergence of self-supervised methods showed that it is possible to train SIDE models using only synchronized image pairs from stereo rigs [27,57,58] or sequences of frames [59].

2.2.1. Training with stereo images

With a breakthrough work, Garg et al. [57] proposed a CNN that is fed with pairs of stereo images and retrieves depth maps. The model learns the nonlinear transformations necessary to recover the depth map using a loss function equivalent to the photometric difference between images. Godard et al. [27] also formulated a deep CNN, called Monodepth, which receives only the left image of the stereo pair during testing. During training, their network learns to infer the disparity maps of both camera images. Going further, Godard et al. [60] improved Monodepth with a loss function able to reduce the number of artifacts, to deal with occluded pixels and to disregard pixels that do not change in a sequence of images.

With a semi-supervised scope, the pipelines proposed by Kuznietsov et al. [61] and Amiri et al. [62] employ supervised learning from sparse data produced by LiDAR sensors, as well as self-supervised training reasoned by stereo pairs. On the other hand, the pipelines proposed in the works of Yang et al. [63] and Andraghetti et al. [64] use a self-supervised training strategy, based on stereo images, and also a supervised training procedure assisted by VO. Furthermore, Ramirez et al. [65] was the first to consider stereo self-supervised learning along with the supervision of semantic classes.

Following the stereo matching setup, Luo et al. [66] and Tosi et al. [67] presented SIDE frameworks that synthesize the right view of the stereo pair from the original left image. In a multi-task approach, Li et al. [68] introduced a deep network that estimates depth and 6D-camera pose with the assistance of a loss function that addresses the spatial and temporal aspects of the incoming image pairs. Also, Babu et al. [69] developed a framework that predicts depth maps and camera pose with the help of a loss based on the Charbonnier penalty [70]. This penalty expression is used in both spatial and temporal reconstruction errors that constitute the objective function.

Other stereo-based methods resort to trinocular training [71], synthetic training data [72], generative adversarial training [73–75], uncertainty handling [76,77], occlusion handling [78] and knowledge distillation [79]. Network architectures that are lighter and faster for real-time applications in embedded systems [80–82] are also applied in the stereo matching self-supervised approach of SIDE.

2.2.2. Training with monocular video

With the first efforts to perform self-supervised depth estimation from monocular video, Zhou et al. [59] proposed a self-supervised model that performs SIDE and camera pose estimation. On the other hand, Yang et al. [83] developed a CNN-based method that estimates depth and surface normals through edge recognition. In a later work, Yang et al. [84] introduced an architecture that retrieves depth maps, surface normals and edge representations using a regularization method named 3D as-smooth-as-possible.

Also with a multitask approach, Yin et al. [85] proposed a self-supervised framework that predicts depth, optical flow and camera pose through an adaptive geometric consistency loss function, which is robust to occlusions and outliers. In parallel, Mahjourian et al. [86] introduced a method that estimates depth and ego-motion from temporally consistent adjacent images using a 3D geometric loss function. Wang et al. [87] leveraged depth normalization operations and a differentiable version of a direct VO algorithm. Showing promising results on SIDE and VO, the method from Casser et al. [88] can model dynamic objects present in the scene with the support of instance segmentation masks.

Other SIDE strategies, self-supervised by monocular videos, employ semantics, flow and camera motion information [10,89–92], as well as other types of geometric constraints [92–94] and lightweight CNNs [95].

The monocular video approach is challenging since the deep network needs to estimate the camera transformations through the sequence of input images, in addition to the depth of the scene. CNNs, trained with stereo images, do not need to estimate the camera pose, as they explore the correspondence of both images of the stereo pair. However, as a disadvantage, faults are prone to occur in areas close to occlusions [60].

2.3. Depth completion

Depth completion is distinct from the SIDE task, as the former is focused on the densification of sparse and noisy point cloud data, obtained from light-structured sensors, LIDAR sensors or SLAM/Structure from Motion (SfM) algorithms [96]. In addition to filling data into depth channel spaces without information (depth inpainting) [97,98], the depth completion task also aims at depth denoising [99,100] and depth super-resolution [101,102].

The first depth completion strategies typically resorted to handcrafted features to infer dense depth or disparity images [103]. In this context, Ku et al. [104] presented an unguided depth completion algorithm, based on inversion, expansion and morphological closing. On the other hand, the method of Schneider et al. [105] exploits the guidance of semantic labels and edge maps to estimate dense images. However, such classic approaches cannot generalize well for different types of scenes, demanding fine adjustments of specific parameters for each circumstance. With that in mind, these classic algorithms were outperformed by novel learning-based models, which can be divided regarding the presence or absence of a guidance RGB image.

2.3.1. Depth completion from sparse samples

In this category, depth completion techniques retrieve dense depth maps from sparse depth samples. Uhrig et al. [17] introduced a sparse convolutional network composed of sparse convolutional layers that handle sparsity through the use of observation masks. Such masks hold the valid pixels information from the input feature maps, which are propagated by pooling operations.

Enhancing the normalized convolution approach, Eldesokey et al. [106] proposed an unguided CNN that receives sparse depth and confidence maps as input. This network employs new normalized convolutions that infer confidence maps and transmit them to the adjacent layer. Chodosh et al. [107] used established concepts of sparse representation learning (dictionary learning) in a neural network pipeline, based on the alternating direction neural network, to address the task of depth completion.

Huang et al. [108] developed a CNN that benefits from observation masks and sparsity-invariant operations such as upsampling, average and convolution. Recently, sparsity-invariant convolutions [17] are being revisited and complemented by novel mask aware operations to handle the RGB guided depth completion task [109]. Other few works are focused on closing the gap between unguided and guided depth completion methods [110].

2.3.2. Guided depth completion

The approaches that deal with guided depth completion leverage RGB image cues to densify sparse and noisy depth maps [111]. Ma et al. [7] proposed a CNN that is fed with color images and sparse depth data sampled with the Bernoulli probability strategy. This deep model can predict dense depth maps from different numbers of target samples. In a later work, Ma et al. [9] introduced a self-supervised pipeline that handles sparse data and also receives sequences of RGB images as input.

Addressing both SIDE and depth completion, Cheng et al. [112] designed a framework that employs a Convolutional Spatial Propagation Network (CSPN). This CNN propagates through the application of recurrent convolutions in a fixed neighborhood and, due to this, it can learn the affinity matrix related to the pixels of the output depth map from the depth estimation stage. Posteriorly, Cheng et al. [96] enhanced the CSPN's depth completion performance through an adaptive technique that infers some network's hyper-parameters.

Inspired by the CSPN, the network architecture proposed by Park et al. [113] learns the affinities of the estimated depth map pixels, and its confidences, in a non-local neighborhood. In another variation, Xu et al. [114] designed a deformable spatial propagation network that propagates with adaptive receptive fields and affinity matrices.

On the other hand, Chen et al. [115] developed a depth completion CNN that learns 2D–3D features in a multi-level way, while Li et al. [116] used blocks of hourglass networks that are fed with sparse samples and guided by the encoder features. Recently, Tang et al. [111] designed a guided method that employs spatially variable convolution kernels. Other works also benefit from morphological operators [117], additional surface normals information [103,118], phase masks [119] and cross-guidance techniques [120].

2.4. Final considerations

From Table 1, we can see that state-of-the-art supervised CNNs reach a prediction speed that does not exceed 20 fps, using powerful Graphic Processing Units (GPUs) and feature extraction networks with no less than 25M trainable parameters. However, comparing the results obtained in [60,77,88,95] with [11,13,26], we have that supervised methods still obtain more accurate depth predictions. Furthermore, self-supervised methods, in some cases, require VO techniques [64,87,90], semi-supervised pipelines [61,67,121], both images from the stereo pair during training [57,78,79], as well as optical flow, semantic and surface normals cues [10,84,91,122] to obtain better results.

Therefore, this work aims to explore the benefits of supervised networks, along with other deep learning techniques, to solve problems that involve SIDE. The next section shows how the current proposal is correlated to state-of-the-art FCN architectures, which employ supervised learning to perform monocular depth estimation. Other deep learning methods, which include the tasks of semantic segmentation, surface normals estimation and depth completion, are also covered.

3. Method

In this section, we exploit the concepts behind our novel supervised monocular depth estimation method. We aim to introduce the details from the proposed baseline CNN architecture, which is divided into the encoder and decoder well-defined phases. Along with such architecture, 4 different encoder models are presented. The developed models are used in the ablation studies step so that we may verify the one that best fits our framework. We also cover a significant variety of loss functions that are employed in the monocular depth estimation studies.

Later, the pipelines that account for semantic segmentation, surface normals estimation and depth completion are described. Through small modifications in the proposed CNN, the former approach uses some pre-trained weights for semantic segmentation to enhance the depth inference accuracy. In the second strategy, a new surface normals estimation module is developed, which leverages the retrieved depth maps, the feature maps produced by the entire baseline and edge constraints. Furthermore, a simple geometric loss function is introduced to handle both depth and surface normals predictions efficiently. We add a Gaussian blur technique to the proposed module to verify its robustness to noisy inputs.

Table 1
Specifications of SIDE state-of-the-art works. The symbol † indicates the total number of trainable parameters of the encoder, while ‡ denotes the total size of the framework.

Method	Dataset	GPU	Network encoder	Parameters	Prediction speed
BTS [13]	KITTI Depth / NYU Depth V2	NVIDIA GTX 1080 Ti	DenseNet-161	47 M‡	16 fps
DORN [26]	KITTI Depth / Make3D / NYU Depth V2	NVIDIA Titan X Pascal	ResNet-101	110 M‡	2 fps
VNL [11]	KITTI Depth / NYU Depth V2	Huawei GPU-accelerated Cloud Server	ResNext-101 ($32 \times 4d$)	44 M†	2 fps
DenseDepth [15]	KITTI Depth / NYU Depth V2	NVIDIA TITAN Xp	DenseNet-169	42 M‡	–
ACAN [42]	KITTI Depth / NYU Depth V2	NVIDIA GTX 1080 Ti	ResNet-101	44 M†	–
PAP-Depth [56]	KITTI Depth / SUNRGBD / NYU Depth V2	NVIDIA P40	ResNet-50	25 M†	5 fps
DABC [123]	ScanNet / KITTI Depth	NVIDIA GTX 1080	ResNext-101 ($64 \times 4d$)	–	1 fps
APMoE [124]	NYU Depth V2 / BSDS500 / Stanford-2D-3D / Cityscapes	NVIDIA Titan X	ResNest-50	25 M†	5 fps
CSWS [125]	NYU Depth V2 / KITTI Depth	NVIDIA Tesla Titan X	ResNet-152	60 M†	5 fps
Cao et al. [34]	NYU Depth V2 / KITTI Depth / SUNRGBD	–	ResNet-152	82 M‡	–
Kuznietsov et al. [61]	KITTI Depth	NVIDIA GTX 980 Ti	ResNet-50	80 M‡	–
LSIM [126]	KITTI Depth	NVIDIA Titan X	ResNet-50	25 M†	12 fps
DHGRL [127]	KITTI Depth / Make3D / NYU Depth V2	NVIDIA K80	ResNet-50	25 M†	5 fps
SDNet [128]	KITTI Depth / Cityscapes	NVIDIA Titan Xp	ResNet-50	25 M†	5 fps
monoResMatch [67]	KITTI Depth / Cityscapes	NVIDIA 2080 Ti	monoResMatch	42 M‡	29 fps
monoResMatch [67]	KITTI Depth / Cityscapes	Jetson TX2	monoResMatch	42 M‡	1 fps
3Net [71]	KITTI Depth / Cityscapes	NVIDIA 2080 Ti	ResNet-50	25 M†	49 fps
3Net [71]	KITTI Depth / Cityscapes	Jetson TX2	ResNet-50	25 M†	2 fps
VOMonodepth [64]	KITTI Depth	NVIDIA 2080 Ti	ResNet-50	25 M†	41 fps
VOMonodepth [64]	KITTI Depth	Jetson TX2	ResNet-50	25 M†	1 fps
Monodepth [27]	KITTI Depth / Make3D	NVIDIA Titan X	ResNet-50	31 M‡	28 fps
Monodepth2 [60]	KITTI Depth	NVIDIA Titan X	ResNet-18	14 M‡	–
SA-Attention [129]	KITTI Depth / Make3D	–	ResNet-50	34 M‡	–
Struct2depth [88]	KITTI Depth / Cityscapes / Fetch Indoor Navigation	GTX 1080Ti	Struct2depth	14 M‡	30 fps

In the last pipeline, we employ the sampling strategy based on the categorical distribution to densify sparse and noisy depth maps for indoor and outdoor environments. All the variations of the baseline network can be executed at a frame rate above 20.

Posteriorly, we introduce the experimental phases that cover the implementation details, the datasets used and the evaluation criteria applied to our results. Finally, we present the ablation studies, with the quantitative and qualitative analysis, and the conclusion.

3.1. Network architecture

The proposed supervised FCN, named DenseSIDENet (DSN), is inspired by the framework of Lee et al. [13] since their model is state-of-the-art for the SIDE task. Compared to the most accurate method of Lee et al. [13], ours can reach 56× fewer trainable parameters. Fig. 1 illustrates the baseline of the DenseSIDENet highlighting the encoder–decoder structure.

We implemented as encoder the following lightweight feature extraction CNNs: MobileNet [130], DenseNet-121 [22] and new variations of the robotic grasp detection network proposed by Ribeiro et al. [131]. The aforementioned networks have $0.6M$ (up to $4M$), $6M$ and $0.3M$ (up to $9M$) trainable parameters, respectively, which makes it possible to evaluate the framework at a high frame rate without losing accuracy.

The CNN depicted in Fig. 2 refers to one of the proposed variations of Ribeiro's model [131]. Due to its efficiency and size, it is designed for the DSN encoder stage. Particularly, the number of filters ($\{32, 164, 96, 128\}$) and the kernel size (3×3) of each convolution are maintained from the original network. However, the flatten operation, the FC layers and the output layer are removed.

We also developed a dense configuration of the network of Fig. 2. The standard structure of the DenseNet presents a sequence of dense convolution blocks interspersed with transition layers which are composed of convolution and average pooling operations. Therefore, layers 1, 2, 3 and 4 of the architecture in Fig. 2 are redesigned to resemble dense blocks alternated with transition layers. However, the filter ratio ($\{32, 164, 96, 128\}$), the kernel size (3×3) and the max-pooling operations are preserved, according to what is proposed by Ribeiro et al. [131].

Table 2 details 3 lightweight versions of the proposed dense architecture. For each model, k refers to the growth rate of the filters, which is added at the end of the dense blocks, θ corresponds to the filter's compression and t refers to the total number of trainable parameters. Beyond the differences presented in Table 2, the stages of the proposed models are initialized with the following filter distribution: 32 filters for the initial convolution, 164 filters for the dense blocks 1, 2 and 3, 96 filters for the transition layers 1, 2 and 3, 128 filters for the dense block 4. The first convolutions of the dense blocks are set up with 2× the current number of filters.

Attached to the DSN encoder, a pair of upconvolution, concatenation and convolution blocks replace the last pooling layers of the feature extraction network to prevent the feature maps resolution from decreasing. Afterwards, a Dense ASPP [132] is implemented according to Lee et al. [13], who also utilize the dilated rates 3, 6, 12, 18, 24.

Since this module consists of densely connected dilated convolution layers, it does not suffer from the same degradation problems as the original ASPP. Thus, the densely connected ASPP is applied to extract multi-scale features and, at the same time, to avoid the excessive reduction of the receptive field.

Due to this, the DSN decoder stage processes larger feature maps, with multiple-sized receptive fields, and performs less upsampling operations to recover the original image resolution (D) at the output. Fig. 1 shows that the decoder stage does not add highly complex neural layers that could increase the number of trainable parameters and decrease the DSN's prediction speed.

As also can be noticed in Fig. 1, the upconvolution block is formed by a convolution layer with 3×3 kernel, which is preceded by a nearest neighbors upsampling operation that retrieves feature maps with 2× increased resolution. In the framework, such blocks are followed by batch normalization operations and, subsequently, by skip-connections/concatenations. The first upconvolution block is initialized with only 256 filters which are decreased by a fraction of 2 until the last layer.

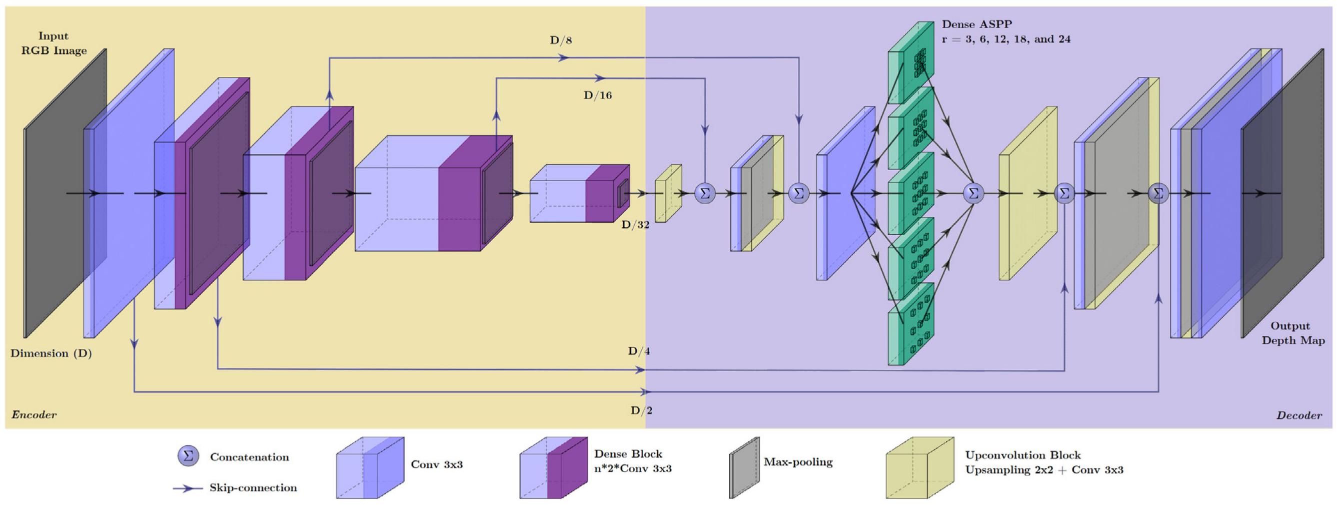


Fig. 1. Proposed baseline of the DenseSIDENet.

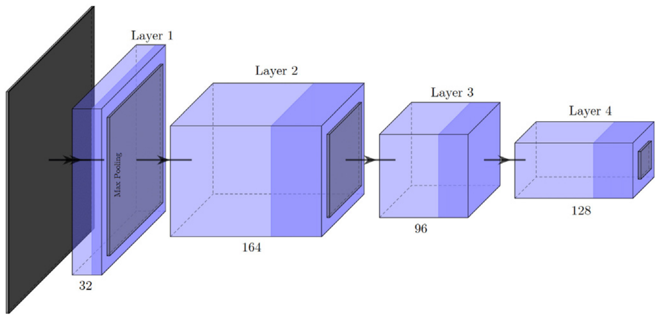


Fig. 2. Modified structure of the CNN proposed by Ribeiro et al. [131] for the task of robotic grasping detection.

Table 2
Composition of the proposed models to be applied in the feature extraction stage of the DenseSIDENet baseline. For all models, k indicates the filters growth rate, θ is the filters compression and t is the total number of parameters.

Layers	Model 1 ($k = 6,\ \theta = 1,\ t = 1M$)	Model 2 ($k = 16,\ \theta = 1,\ t = 3M$)	Model 3 ($k = 24,\ \theta = 1,\ t = 9M$)
Initial convolution	3 × 3 conv, stride 2	3 × 3 conv, stride 2	3 × 3 conv, stride 2
Initial pooling	3 × 3 max pooling, stride 2	3 × 3 max pooling, stride 2	3 × 3 max pooling, stride 2
Dense block 1	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 6$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 6$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 6$
Transition layer 1	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2
Dense block 2	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 12$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 12$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 12$
Transition layer 2	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2
Dense block 3	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 18$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 18$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 24$
Transition layer 3	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2	3 × 3 conv, stride 1 2 × 2 max pooling, stride 2
Dense block 4	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 16$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 14$	$\begin{bmatrix} 3\times3 \text{ conv, stride } 1 \\ 3\times3 \text{ conv, stride } 1 \end{bmatrix} \times 18$

Throughout the network structure, ReLU [133] and eLU [134] activation functions are employed, with the exception of the output layer that uses the sigmoid function. The output depth map of the DSN is multiplied by the maximum depth value comprising the dataset used. The network is fed with a single RGB image with 352 × 704 and 416 × 544 pixels and predicts a depth map with 352 × 1216 and 480 × 640 pixels for the outdoor [17,135] and indoor [18,136–138] datasets respectively.

3.2. SIDE loss functions

The losses Scale-invariant Error ($\mathcal{L}_{SILog}$) [28], modified Scale-invariant Error ($\mathcal{L}^{+}_{SILog}$) [13], Huber ($\mathcal{L}_{Huber}$) [139], BerHu ($\mathcal{L}_{BerHu}$) [31], Mean Absolute Error ($\mathcal{L}_1$) [27], Mean Squared Error ($\mathcal{L}_2$) [26], Charbonnier ($\mathcal{L}_{charbo}$) [140] and Log-cosh ($\mathcal{L}_{cosh}$) [141] are used in the training phase of the network depicted in Fig. 1. The pixels without depth information are disregarded by the loss functions, as they do not influence the adjustment of the network weights.

The $\mathcal{L}_{SILog}$ expression (Eq. (1)) was introduced in [28] and has often been used in monocular depth estimation CNNs [13,39,139] since it is an efficient function to determine the relationships between the depths of a scene without the influence of its scale [28]. Such a loss function, modified in [13], can be expressed by Eq. (2), in which y_i^* represents the ground truth depth values, y_i refers to the prediction of the network for the ith pixel and N is equal to the total number of samples.

$$\mathcal{L}_{SILog}(h) = \frac{1}{N}\sum_i h_i^2 - \frac{\lambda}{N^2}\left(\sum_i h_i\right)^2, \tag{1}$$
$$h_i = \log(y_i) - \log(y_i^*), \quad \lambda = 0.85$$

$$\mathcal{L}_{SILog}^{+} = \theta\sqrt{\mathcal{S}(h)}, \tag{2}$$
$$\mathcal{S}(h) = \frac{1}{N}\sum_{i=1}^{N} h_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} h_i\right)^2 + (1-\lambda)\left(\frac{1}{N}\sum_{i=1}^{N} h_i\right)^2,$$
$$h_i = \log(y_i) - \log(y_i^*), \quad \lambda = 0.85, \quad \theta = 10.0$$

BerHu, the inverse function of $\mathcal{L}_{Huber}$ (Eq. (3)), is also applied in this work due to its advantages related to the combination of the losses $\mathcal{L}_1$ and $\mathcal{L}_2$. Among these benefits, residuals ($|y_i - y_i^*|$) with high values are penalized by the function $\mathcal{L}_2$ which is sensitive to outliers. On the other hand, residuals with low values are penalized by the $\mathcal{L}_1$ function, which returns greater depth values than those returned by the $\mathcal{L}_2$. The BerHu penalty can be expressed by Eq. (4) with the default $\kappa = 5$.

$$\mathcal{L}_{Huber}(h) = \begin{cases} |h| & |h| \geq g, \\ \frac{h^2+g^2}{2g} & |h| < g, \end{cases} \tag{3}$$
$$h = \log(y) - \log(y^*), \quad g = \frac{1}{\kappa}\max_i\left(|\log(y_i) - \log(y_i^*)|\right)$$

$$\mathcal{L}_{BerHu}(h) = \begin{cases} |h| & |h| \leq g, \\ \frac{h^2+g^2}{2g} & |h| > g \end{cases} \tag{4}$$
$$h = \log(y) - \log(y^*), \quad g = \frac{1}{\kappa}\max_i\left(|\log(y_i) - \log(y_i^*)|\right)$$

In addition to the $\mathcal{L}_{SILog}$, $\mathcal{L}_{SILog}^{+}$, $\mathcal{L}_{BerHu}$ and $\mathcal{L}_{Huber}$ expressions, the $\mathcal{L}_1$ and $\mathcal{L}_2$ loss functions, represented by Eqs. (5) and (6), respectively, are employed during training, as they are widely addressed by single-view depth estimation networks [7,26,27]. The Charbonnier penalty, described by Eq. (7), has been applied to problems involving regression, mainly in flow estimation approaches [140] and self-supervised SIDE frameworks [69]. The parameters a and α of $\mathcal{L}_{charbo}(h)$ are adjusted according to what is established by the work of Babu et al. [69].

$$\mathcal{L}_1(h) = \frac{1}{N}\sum_{i=1}^{N}|h_i|, \quad h_i = \log(y_i) - \log(y_i^*) \tag{5}$$

$$\mathcal{L}_2(h) = \frac{1}{N}\sum_{i=1}^{N}(h_i)^2, \quad h_i = \log(y_i) - \log(y_i^*) \tag{6}$$

$$\mathcal{L}_{charbo}(h) = \frac{1}{N}\sum_{i=1}^{N}\left(h_i^2 + \alpha^2\right)^a, \tag{7}$$
$$h_i = \log(y_i) - \log(y_i^*), \quad a = 0.45, \quad \alpha = 10^{-3}$$

The Log-cosh loss (Eq. (8)) presents an inverse behavior to that of BerHu, operating as $\mathcal{L}_1$ for high residuals and as $\mathcal{L}_2$ for low residuals. This makes its behavior similar to the Huber function, yet the $\mathcal{L}_{cosh}(h)$ has the advantage of being totally differentiable.

$$\mathcal{L}_{cosh}(h) = \sum_{i=1}^{N}\log\left(\cosh\left(h_i\right)\right), \quad h_i = y_i - y_i^* \tag{8}$$

Based on the presented losses, we aim to use the one that produces the best results in the tests of the proposed network, replacing the term $\mathscr{L}(y, y^*)$ of the attention-based function at distant depths, Eq. (9), which is introduced in the work of Jiao et al. [43]. According to the authors, this function allows the network to focus on regions where the availability of pixels with depth information is reduced, once they are unevenly distributed.

The constant β, the attention term γ in logarithmic scale and the function $\mathcal{L}_a$ in summation setup are the modifications proposed over the original expression [43]. In our approach, β is set to 10.0 and 1.0 for the indoor and outdoor datasets, respectively, due to scale variations.

$$\mathcal{L}_a = \sum_{i=1}^{N}(\gamma + \epsilon)\,\mathscr{L}(y, y^*), \tag{9}$$
$$\gamma = \frac{\log(y_i^*)}{\max_i(\log(y_i^*))}, \quad \epsilon = 1.0 - \frac{\min(\log(\beta y_i), \log(\beta y_i^*))}{\max(\log(\beta y_i), \log(\beta y_i^*))}$$

Finally, the loss functions $\mathcal{L}_1^{+}$ and $\mathcal{L}_2^{+}$ are similar to the $\mathcal{L}_1$ and $\mathcal{L}_2$ expressions respectively. However, the former ones are computed only by the total sum of the equation results, instead of the average calculation. Based on experimental analysis, these variations of $\mathcal{L}_1$ and $\mathcal{L}_2$ are applied only when training the network with the indoor datasets.

3.3. SIDE with semantic cues pipeline

The influence of additional semantic information on the proposed FCN is analyzed through fine-tuning. According to such a strategy, the DSN is firstly pre-trained for semantic segmentation and part of the computed weights are transferred to the monocular depth estimation framework.

The loss function that is used for the semantic segmentation task can be represented by Eq. (10), where K is equal to the total number of categories, y_i^* is the ground truth in the one-hot encoding format, y_i is the network prediction, after the application of the softmax activation function, and t represents the elements of the input feature map.

$$\mathcal{L}_{cce}(y, y^*) = -\sum_{i=1}^{K} y_i^* \log(y_i), \quad y_i = \frac{e^{t_i}}{\sum_j^K e^{t_j}} \tag{10}$$

Only part of the weights obtained with the semantic segmentation framework is fine-tuned in the depth estimation phase due to the structural differences between the two CNNs, especially in the last layers.

3.4. SIDE with surface normals cues pipeline

Differently from previous works [2,55,56] and closer to the approach of Yin et al. [11], we leverage the correlation between depth and surface normals estimation in only one framework. Thus, we do not employ any extra branches or other supplementary models to exploit 3D geometric constraints from surface normals. Contrariwise, they are inferred from the output depth map and the same neural layers used for SIDE.

To train our CNN for surface normals estimation, the ground truth is obtained as in the works of Fouhey et al. [53], Qi et al. [55] and Zhang et al. [56]. Therefore, firstly, the point cloud in 3D space is recovered from the depth maps, assuming the pinhole camera model according to Eq. (11). In such an equation, (υ_i, ν_i) refers to the pixel positions in the image, (x_i, y_i, z_i) represents the 3D coordinates of the point cloud, where the values of z_i are equal to the depths of the scene ρ_i, (o_x, o_y) is the location of the optical axis and f_x and f_y are the focal lengths of the axes x and y respectively.

$$x_i = \frac{z_i\,(\upsilon_i - o_x)}{f_x}, \quad y_i = \frac{z_i\,(\nu_i - o_y)}{f_y}, \quad z_i = \rho_i \tag{11}$$

As proposed by Qi et al. [55], after reconstructing the point cloud, we determine the tangent planes to the cloud depth data by defining a neighborhood of points of size m, which belongs to the same plan. Once computed the local planes, the surface normal vectors $s = (s_x, s_y, s_z)$ can be obtained through the relationship $Ds = c$, where $D_{m\times3}$ is the coordinate matrix of the point cloud and $c_{m\times1}$ is a vector with constant values. In order to minimize $\|Ds - c\|_2^2$, the least-squares solution $\hat{s}_{3\times m}$ is applied to the normal equations system of Eq. (12).

$$D^T D\hat{s} = D^T c \tag{12}$$

From Eq. (12), we can get to Eq. (13) which is used to directly calculate the normalized surface normal vectors. Following what was established by Qi et al. [55], the vector $c_{m\times1}$ is defined with unit values.

$$\hat{s} = \frac{(D^T D)^{-1} D^T c}{\|(D^T D)^{-1} D^T c\|_2} \tag{13}$$

The developed module shares the same feature maps produced in the SIDE step, yet they are concatenated with the predicted depth map and with the output from the Sobel edge detection algorithm [142]. This algorithm is computationally efficient, it is robust to the noise present in the input grayscale image and it retrieves the high gradient features of the input which helps the proposed CNN to better reason about depth in the boundaries of the objects. Eqs. (15) and (16) describe the H_x and H_y masks used by the Sobel filter to calculate the gradients approximation of the image I (Eq. (14)). With the achieved gradient vector, its magnitude H and direction Ψ are computed to determine the pixels in border regions, which follow the relationship $H(i,j) > \tau$ wherein τ is equal to the threshold value.

$$I_x = \frac{\partial I}{\partial x}, \quad I_y = \frac{\partial I}{\partial y} \tag{14}$$

$$I_x(i,j) = (I * H_x)(i,j), \quad I_y(i,j) = (I * H_y)(i,j) \tag{15}$$

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_y = -H_x^T,$$
$$H = \sqrt{I_x^2 + I_y^2}, \quad \Psi = arctan\left(\frac{I_y}{I_x}\right) \tag{16}$$

To validate the noise robustness of the surface normals module, a convolutional layer with Gaussian kernel [143] is implemented before the Sobel edge detector. This convolutional layer is responsible for filtering the noise present in the input grayscale image I, making the output of the edge detection step less affected by them. The proposed linear Gaussian filter is free from second peaks in the frequency domain and can be expressed by Eq. (17) wherein I_g is the filtered image and G refers to the $m \times n$ kernel that obeys a Gaussian distribution with $\sigma = 1.76$.

$$I_g(i,j) = (I * G)(i,j) = \sum_m \sum_n G(m,n) I(i-m, j-n), \tag{17}$$
$$G(m,n) = \frac{1}{2\pi\sigma^2} e^{-\frac{m^2+n^2}{2\sigma^2}}$$

For the joint approach of depth and surface normals estimation, we designed the geometric loss function $\mathcal{L}_{2.5D}$, whose expression comprises the functions $\mathscr{L}(y, y^*)$ and $\Phi(n, n^*)$. The former loss is replaced by the one that generates the most accurate depth predictions. The latter loss computes the cosine similarity between the valid pixels from the surface normals predictions n and the ground truth n^*. Also, $\Phi(n, n^*)$ is weighted by a proportionality constant ψ.

$$\mathcal{L}_{2.5D} = \mathscr{L}(y, y^*) + \psi(\Phi(n, n^*)), \tag{18}$$
$$\Phi(n, n^*) = 1 - \left(\frac{\langle n, n^*\rangle}{\|n\|_2 \|n^*\|_2}\right)$$

3.5. Depth completion pipeline

This work also aims to analyze the behavior of the results produced by the network when it is trained and tested with additional depth data at the input, which can be obtained through low-cost 2D LiDARs and the outputs of visual odometry algorithms [7].

We simulate visual odometry algorithms or depth sensors by a sampling technique which incorporates the multinomial distribution with the number of independent experiments $n = 1$. Thus, only a random sampling attempt of a sparse depth data χ is performed for the ground truth pixels paired with the network input images. This special case of the multinomial distribution is called categorical distribution, in which the number of categories is fixed at $k = 2$. The Eq. (19) regards the mass probability function associated with the random variable χ.

$$F(\chi \mid \varphi) = \prod_{i=0}^{k-1} \varphi_i^{\chi_i}, \quad \varphi_i > 0, \quad \sum_{i=0}^{k-1} \varphi_i = 1 \tag{19}$$

The probabilities φ_1 and φ_0 refer, respectively, to the success and failure of sampling a sparse depth data. Therefore, as proposed by Ma et al. [7], $\varphi_1 = \frac{d}{d^*}$ and $\varphi_0 = 1 - \frac{d}{d^*}$ wherein d and d^* represent the desired amount of samples and the amount of valid data available per depth map respectively.

4. Experiments

4.1. Implementation details

To implement and train the proposed network, a server with 4.20 GHz CPU, 2TB HD, 200GB SSD, Ubuntu 16.04 operating system, Tensorflow 1.13.1 development framework and two NVIDIA Titan X GPUs (both are used in the training phase and only one in the test phase). The Adam optimization algorithm [144] is standard for all experiments, with $\alpha = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$, as proposed by Kingma et al. [144].

Furthermore, the learning rate is set to 10^{-4} with a polynomial decay at a rate of 0.9. In each experiment, the CNN is trained for 50 epochs with batch size 16. However, the batch size is set to 32 in the training steps that involve the MobileNetV2-50, the network of Fig. 2 and the proposed Model 1.

Although the feature maps generated by the first CNN convolution layers present primitive visual information, the parameters of the batch normalizations and the weights of the first two convolutions of the feature extraction network are not fixed, as occurs in the methods proposed by Fu et al. [26] and Lee et al. [13].

To avoid overfitting and artificially increase the number of training samples, three types of online data augmentation are employed. Among them, the random horizontal flip operations, with a probability of 50%, and rotation in the intervals of [−2.5°, 2.5°] and [−1.0°, 1.0°] for the indoor and outdoor datasets respectively. Brightness [0.75, 1.25], color [0.9, 1.1] and contrast [0.9, 1.1] distortions are also considered with a probability of 50%.

4.2. Datasets

4.2.1. Outdoor

In previous works of the SIDE literature, the KITTI Raw [145], whose images are obtained directly from the sparse point clouds generated by a LiDAR sensor, is used as ground truth. However, recently, the KITTI Vision Benchmark Suite has released the official KITTI Depth dataset, which is composed of denser depth maps due to the combination of 11 laser scans [17]. The KITTI Depth consists of stereo images in gray and color scales, which are captured by two pairs of stereo cameras that compose the autonomous navigation platform capable of traversing external environments to collect data.

Moreover, the dataset provides depth measurements generated by the scans of the Velodyne HDL-64E LiDAR in the form of point clouds. Another relevant aspect of the KITTI Depth is the denser ground truth, compared to the point clouds, which presents a grayscale image format, whose pixels provide the depth information of the left and right scenes from the stereo cameras.

Both the stereo images and the densified ground truth have a typical resolution of 375 × 1242 pixels and are divided into 61 scenes for training and testing, which comprise the categories "city", "residential", "road" and "campus". At first, the KITTI Raw [145] was subdivided into 32 different training scenes and 29 test scenes by Eigen et al. [28]. The training subdivision contains 23 488 images and the testing set comprises 697 images. Bringing the Eigen Split to the KITTI Depth [17], we aim to use 23 158 images randomly selected from the total 23 488 training samples.

For the semantic segmentation task, the KITTI Vision Benchmark provides 200 manually annotated maps for training and 200 test samples with a typical resolution of 375 × 1242 pixels. Due to this, additional training data from the Cityscapes dataset [135] are included. The Cityscapes consists of 5K semantic maps (with a resolution of 1024 × 2048 pixels), which are formed by fine annotations that sum up 30 different classes. From the 5K images, 2975, 1525 and 500 are reserved for training, testing and validation respectively. Therefore, in this work, the KITTI's 200 semantic training maps are complemented by 3475 Cityscapes samples.

Corresponding to the semantic maps, the Cityscapes dataset also provides the same number of disparity maps, which can be used to reconstruct their respective depth maps. Due to the availability of disparity maps for training, testing and validation, 5K depth maps reconstructed from the Cityscapes are used to pre-train the proposed framework.

We also apply the KITTI Odometry dataset [146] in the VO studies with the proposed CNN. Such a dataset is composed of 22 sequences with stereo images of outdoor scenarios. However, only the first 11 sequences have their respective reference trajectories, while the remaining ones are used for benchmark evaluation. As proposed by Loo et al. [5], we focus on the stereo sequences with available ground truth.

4.2.2. Indoor

The NYU Depth V2 [18] is composed of pairs of RGB images and depth maps, with a standard resolution of 480 × 640 pixels, captured by the camera and depth sensor of a Microsoft Kinect. Altogether, the official dataset is formed by 240K pairs of RGBD training samples that integrate 464 indoor scenes. As other SIDE state-of-the-art works [11,15,42], we apply the official subdivision of the NYU Depth V2 data which comprises 249 training scenes and 215 test scenes.

From the 249 training scenes, totaling 120K samples, 24 931 images are selected to train the proposed CNN. On the other hand, from the 215 test scenes, 654 images are pre-defined to test our framework. Additionally, 8587 samples, from the SUNRGBD [136], Berkeley B3DO [137] and SUN3D [138] datasets are employed in pre-training phases.

To address the semantic segmentation task, the NYU Depth V2 provides 1449 images, with 35 064 different objects, that present semantic annotations with 894 different categories. We also add 10 336 samples, with their corresponding semantic labels, from the SUNRGBD dataset [136] to the 1449 available training images.

The surface normals ground truth, for both indoor and outdoor datasets, is produced according to what is proposed in the work of Fouhey et al. [53].

4.3. Evaluation metrics

The proposed FCN is tested outdoors according to the Eigen Split. However, as this subdivision is originally proposed for the KITTI Raw dataset [145], only 652 of the 697 test images present ground truth in the KITTI Depth [17]. Furthermore, the network predictions are evaluated considering the central crop established by Garg et al. [57] and distance caps of 0–50 m and 0–80 m. For the evaluation of the framework in indoor environments, the central crop proposed in [28] is used in the predictions that correspond to the 654 test images from the NYU Depth V2.

Therefore, in order to present quantitative results, the Scale-invariant Error (Eq. (20)), Threshold (Eq. (21)), Absolute Relative Difference (Eq. (22)), Squared Relative Difference (Eq. (23)), Log10 (Eq. (24)), Mean Absolute Error (Eq. (25)), Linear RMSE (Eq. (26)) and Log RMSE (Eq. (27)), are applied in the evaluation phase wherein T is equivalent to the total amount of pixels with depth information:

$$\frac{1}{|T|}\sum_{y\in T}\left(y-y^*\right)^2-\frac{1}{2T^2}\left(\sum_{y\in T}\left(y-y^*\right)\right)^2 \tag{20}$$

$$\%\ of\ y_i\ s.t.\ \max\left(\frac{y_i}{y_i^*},\frac{y_i^*}{y_i}\right)=\delta<threshold \tag{21}$$

$$\frac{1}{|T|}\sum_{y\in T}\frac{|y-y^*|}{y^*} \tag{22}$$

$$\frac{1}{|T|}\sum_{y\in T}\frac{\|y-y^*\|^2}{y^*} \tag{23}$$

$$\frac{1}{|T|}\sum_{y\in T}|\log_{10}y-\log_{10}y^*| \tag{24}$$

$$\frac{1}{|T|}\sum_{y\in T}|y-y^*| \tag{25}$$

$$\sqrt{\frac{1}{|T|}\sum_{y\in T}\|y-y^*\|^2} \tag{26}$$

$$\sqrt{\frac{1}{|T|}\sum_{y\in T}\|\log y-\log y^*\|^2} \tag{27}$$

The surface normals maps retrieved by our pipeline are evaluated according to the metrics introduced by Fouhey et al. [53], which are commonly used by recent works [2,55,56,147]. Such metrics consider the angular difference for each valid pixel between the estimated surface normals and the ground truth by calculating the mean, median, root mean squared error (RMSE) and the precision of the estimated pixels. This accuracy is measured by the percentage of vectors whose angular difference does not exceed a threshold $\widetilde{n}=\{11.25^\circ,22.5^\circ,30^\circ\}$.

Table 3
Comparison of state-of-the-art depth completion methods according to the KITTI Benchmark [17] metrics ($iMAE = \frac{1}{MAE}$ and $iRMSE = \frac{1}{RMSE}$).

Method	MAE↓	RMSE↓	iMAE↓	iRMSE↓
SGDU [105]	605.470	2312.570	2.050	7.380
NN+CNN [17]	416.140	1419.750	1.290	3.250
ADNN [107]	439.480	1325.370	3.190	59.390
IP-Basic [104]	302.600	1288.460	1.290	3.780
DFuseNet [149]	429.930	1206.660	1.790	3.620
VOICED [150]	299.410	1169.970	1.200	3.560
Morph-Net [117]	310.490	1045.450	1.570	3.840
CSPN [112]	279.460	1019.640	1.150	2.930
DCrgb_80b_3coef [151]	215.750	965.870	0.980	2.430
Conf-Net [152]	257.540	962.280	1.090	3.100
DFineNet [153]	304.170	943.890	1.390	3.210
Spade-RGBsD [154]	234.810	917.640	0.950	2.170
IR_L2 [110]	292.360	901.430	1.350	4.920
DDP [155]	203.960	832.940	0.850	2.100
NConv [106]	233.260	829.980	1.030	2.600
Sparse-to-Dense [9]	249.950	814.730	1.210	2.800
CrossGuidance [120]	253.980	807.420	1.330	2.730
Revisiting NN+CNN [109]	225.810	792.800	0.990	2.420
PwP [118]	235.170	777.050	1.130	2.420
RGB_guide&certainty [8]	215.020	772.870	0.930	2.190
DSPN [114]	220.360	766.740	1.030	2.470
MSG-CHN [116]	220.410	762.190	0.980	2.300
DeepLiDAR [103]	226.500	758.380	1.150	2.560
UberATG-FuseNet [115]	221.190	752.880	1.140	2.340
CSPN++ [96]	209.280	743.690	0.900	2.070
NLSPN [113]	199.590	741.680	0.840	1.990
GuideNet [111]	218.830	736.240	0.990	2.250
Closing ($k = 3$, $it = 5$)	148.963	519.606	0.597	1.082
Closing ($k = 3$, $it = 4$)	129.353	457.346	0.533	0.972
Closing ($k = 3$, $it = 3$)	106.925	391.138	0.458	0.867
Closing ($k = 3$, $it = 2$)	82.137	326.323	0.363	0.747
Closing ($k = 3$, $it = 1$)	**46.724**	**226.214**	**0.211**	**0.541**

4.4. Outdoor ablation studies

4.4.1. Preliminary analysis

Firstly, regarding depth completion approaches to enhance the depth estimation results, we used the closing morphological operation to partially densify the reference maps of the KITTI Depth [17], generating the KITTI Morphological dataset. Along with the closing technique, we applied the RGB_guide&certainty method, introduced by Gansbeke et al. [8], and the depth completion strategy of Hilbert Maps [148] to produce the KITTI Completed and the KITTI Continuous respectively.

Through the KITTI Benchmark metrics, the depth completion results from the application of 5 iterations of the closing morphological technique are compared to other methods in the literature in Table 3. In such a table, it is possible to note that the closing operator, with kernel 3 × 3, is the method that best preserves the ground truth depth information. However, the learning-based techniques retrieve totally dense depth maps whereas the closing ones can be tuned to output images with different densification levels.

In Table 4, we may infer that the bigger the number of iterations (it) the denser the KITTI Morphological dataset becomes. Moreover, the proposed KITTI Completed is the densest dataset employed once it is built with the learning-based RGB_guide&certainty approach. The results exposed in Table 5 show that our pipeline achieves slightly better performance when jointly trained with the partially dense dataset and the BerHu loss function.

However, the accuracy of the method decreases when it is trained with the KITTI Completed and the KITTI Continuous datasets due to the presence of greater distortions in the reference depth maps. Furthermore, although the KITTI Continuous is less dense than the KITTI Completed, the former dataset includes a higher amount of artifacts that hinder the network learning. Therefore, the KITTI Morphological presents the best balance between densification and modification of the original ground truth values, so that our CNN can better understand the structure of the scene.

4.4.2. Structural studies

The following studies are conducted with the KITTI Morphological, with setup $k = 3$ and $it = 2$ since it is beneficial to the training of our CNN. To explore the response of the proposed pipeline to other encoder networks, we carried out new experiments whose results are presented in Table 6. From the metrics of Table 6, we can observe that the DSN with Model 3 as encoder achieves the best trade-off regarding the number of trainable parameters, prediction speed and accuracy. Moreover, analyzing the methods with comparable sizes, it is possible to state that all the proposed lightweight feature extraction networks (Fig. 2, Model 1, Model 2 and Model 3) surpass the state-of-the-art MobileNetV2 and DenseNet-121.

To improve the performance of the DSN with the Model 3 as its feature extraction network (DSN 3), we impose modifications in the structure of the CNN decoder and in its training procedure. Through Table 7, we may verify that the use of the attention loss, combined with the $\mathcal{L}_{BerHu}$ expression, decreases the framework accuracy whereas the application of the $\mathcal{L}^{+}_{BerHu}$ function enhances the results. This indicates that the BerHu loss function is already capable of balancing the penalties for residuals generated in regions closer and more distant to the scene, without the need for an attention term. Furthermore, the network learning is boosted when the term $\mathcal{L}_1$ of the BerHu penalty acts on a wider range of residuals, considering the KITTI Morphological dataset.

The pyramid modules are simple blocks that concatenate multi-scale feature maps of the DSN 3. In the decoder, the Pyramid^1 module concatenates the upsampled feature maps from the last 3 convolution blocks with the output from the final upconvolution, while the Pyramid^2 adds the feature maps from the first convolution to this concatenation. Regarding the metrics presented in Table 7, both the pyramid modules are beneficial to the proposed framework, yet they slow down the network's processing speed.

Also, the results obtained when the DSN 3 is pre-trained with semantic and disparity maps show that both pre-trainings improve the metrics values. Although our method acquires important cues, like the object edges, from the semantic maps with 19 different categories, the monocular features learned from the same depth estimation task show better improvements. All the structural changes in the decoder of the proposed pipeline reduce its inference speed, yet the number of trainable parameters is not significantly increased.

4.4.3. Surface normals analysis

The surface normals module is tested in the DSN 3 framework, along with the Pyramid^1 module, the edge detection phase, the 2.5D loss function and the weights pre-trained on the reference depth maps of the Cityscapes [135] which are produced from their respective disparity maps. Based on the Abs Rel metric in Table 8, we may see that the network configuration with $\psi_1 = 10^6$, $\psi_2 = 10^6$ and without the presence of the Gaussian filter is the one that generates the smallest errors, implying that the edge detector of the surface normals module is robust to noise in the input image.

Moreover, considering the errors and the accuracy of the methods exposed in Table 8, we have that the geometric cues, provided by the surface normals ground truth and computed by the geometric loss function, support the DSN to retrieve 2.5D maps more consistent with the scene. The surface normals

Table 4
Analysis of the densification degrees of the datasets used according to the average number and average percentage of valid pixels. 375×1242 is the typical resolution for the KITTI Depth [17] and 352×704 is the image size that is fed into our framework.

Dataset	Average number of valid pixels		Average percentage of valid pixels	
	375×1242	352×704	375×1242	352×704
KITTI Depth	75 407	43 649	16.190%	17.614%
KITTI Morphological ($k = 3$, $it = 1$)	142 309	82 377	30.555%	33.242%
KITTI Morphological ($k = 3$, $it = 2$)	166 765	96 525	35.806%	38.951%
KITTI Morphological ($k = 3$, $it = 3$)	182 949	105 884	39.280%	42.728%
KITTI Morphological ($k = 3$, $it = 4$)	195 587	113 172	41.994%	45.669%
KITTI Morphological ($k = 3$, $it = 5$)	205 478	118 875	44.118%	47.971%
KITTI Continuous	291 489	158 255	62.585%	63.862%
KITTI Completed	428 031	247 807	91.901%	100%

Table 5
Evaluation of the proposed method considering variations in the loss function and the dataset used during training. The DenseNet-121 is fixed as the feature extraction network of the DSN.

Method	Loss	Dataset	Cap	Abs Rel↓	Sqr Rel↓	RMSE↓	RMSE (log)↓	SILog↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
DenseNet-121	$\mathcal{L}_2$	KITTI Depth	0–80 m	0.114	0.663	4.367	0.173	15.890	0.050	0.853	0.963	0.989
DenseNet-121	$\mathcal{L}_1$	KITTI Depth	0–80 m	0.108	0.624	4.200	0.166	15.294	0.047	0.868	0.965	0.990
DenseNet-121	$\mathcal{L}_{charbo}$	KITTI Depth	0–80 m	0.108	0.639	4.255	0.167	15.323	0.047	0.866	0.966	0.989
DenseNet-121	$\mathcal{L}_{Huber}$	KITTI Depth	0–80 m	0.099	0.538	4.076	0.154	14.308	0.044	0.883	0.972	0.992
DenseNet-121	$\mathcal{L}_{SILog}$	KITTI Depth	0–80 m	0.097	0.513	3.939	0.150	13.569	0.043	0.890	0.975	0.993
DenseNet-121	$\mathcal{L}_{cosh}$	KITTI Depth	0–80 m	0.096	0.518	3.855	0.150	13.800	0.042	0.892	0.974	0.993
DenseNet-121	$\mathcal{L}^{+}_{SILog}$	KITTI Depth	0–80 m	0.095	0.494	3.856	0.146	13.214	0.042	0.894	0.976	**0.994**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Depth	0–80 m	0.089	0.478	3.788	0.142	13.004	**0.040**	0.903	0.977	**0.994**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 1$)	0–80 m	**0.088**	0.471	3.792	0.141	12.750	**0.040**	0.903	**0.979**	**0.994**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 2$)	0–80 m	**0.088**	0.461	**3.780**	**0.140**	**12.747**	**0.040**	**0.905**	0.978	**0.994**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 3$)	0–80 m	**0.088**	**0.458**	3.800	0.142	12.860	**0.040**	0.903	0.977	**0.994**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 4$)	0–80 m	**0.088**	0.479	3.855	0.143	13.024	**0.040**	0.904	0.976	0.993
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 5$)	0–80 m	**0.088**	0.474	3.881	0.143	13.053	**0.040**	0.903	0.977	**0.994**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Continuous	0–80 m	1.640	36.498	24.577	0.978	21.627	0.415	0.013	0.036	0.072
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Completed	0–80 m	0.220	1.741	6.269	0.249	20.577	0.084	0.698	0.924	0.980
DenseNet-121	$\mathcal{L}_2$	KITTI Depth	0–50 m	0.110	0.509	3.203	0.161	14.807	0.047	0.867	0.970	0.992
DenseNet-121	$\mathcal{L}_1$	KITTI Depth	0–50 m	0.104	0.480	3.102	0.155	14.293	0.045	0.881	0.970	0.992
DenseNet-121	$\mathcal{L}_{charbo}$	KITTI Depth	0–50 m	0.103	0.487	3.122	0.156	14.277	0.045	0.879	0.972	0.991
DenseNet-121	$\mathcal{L}_{Huber}$	KITTI Depth	0–50 m	0.094	0.393	2.892	0.142	13.165	0.041	0.897	0.978	0.994
DenseNet-121	$\mathcal{L}_{SILog}$	KITTI Depth	0–50 m	0.093	0.377	2.833	0.139	12.542	0.041	0.904	0.980	0.995
DenseNet-121	$\mathcal{L}_{cosh}$	KITTI Depth	0–50 m	0.092	0.390	2.817	0.140	12.839	0.040	0.904	0.979	0.994
DenseNet-121	$\mathcal{L}^{+}_{SILog}$	KITTI Depth	0–50 m	0.090	0.365	2.771	0.136	12.229	0.040	0.907	0.981	**0.996**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Depth	0–50 m	0.085	0.352	2.721	0.131	11.981	**0.037**	0.916	0.982	0.995
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 1$)	0–50 m	0.084	0.342	2.732	0.130	11.723	**0.037**	0.916	**0.983**	**0.996**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 2$)	0–50 m	**0.083**	0.335	2.701	**0.129**	**11.722**	**0.037**	**0.919**	**0.983**	**0.996**
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 3$)	0–50 m	**0.083**	**0.331**	**2.688**	0.130	11.784	**0.037**	0.916	0.982	0.995
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 4$)	0–50 m	**0.083**	0.345	2.729	0.131	11.906	**0.037**	0.917	0.982	0.995
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Morphological ($k = 3$, $it = 5$)	0–50 m	**0.083**	0.338	2.744	0.131	11.912	**0.037**	0.916	**0.983**	0.995
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Continuous	0–50 m	1.545	28.857	19.418	0.939	23.091	0.395	0.029	0.069	0.128
DenseNet-121	$\mathcal{L}_{BerHu}$	KITTI Completed	0–50 m	0.215	1.454	5.163	0.240	19.124	0.082	0.709	0.933	0.983

Table 6
Comparison of the results obtained by the proposed framework with different feature extraction networks in the KITTI Depth dataset [17]. Model 1, Model 2 and Model 3 refer to the DenseSIDENet with the encoder models of Table 2. The MobileNetV2-50 and MobileNetV2-140 are versions of the MobileNetV2 with depth multipliers 1.4 and 0.5 respectively. The BerHu loss function is used in all experiments and the Features column indicates the number of input feature maps to the decoder stage of the DSN.

Method	Features	Size	Speed	Cap	Abs Rel↓	Sqr Rel↓	RMSE↓	RMSE (log)↓	SILog↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Fig. 2 CNN	256	3 M	**88 fps**	0–80 m	0.102	0.594	4.094	0.160	14.797	0.045	0.876	0.968	0.991
Model 1	256	3 M	59 fps	0–80 m	0.093	0.517	3.941	0.149	13.776	0.041	0.895	0.974	0.992
Model 2	256	6 M	57 fps	0–80 m	0.091	0.501	3.851	0.146	13.243	0.041	0.901	0.977	0.993
Model 3	256	12 M	50 fps	0–80 m	**0.083**	**0.434**	**3.646**	**0.135**	**12.326**	**0.037**	**0.914**	**0.980**	**0.994**
MobileNetV2-50	128	**2 M**	87 fps	0–80 m	0.122	0.760	4.612	0.184	17.074	0.052	0.842	0.955	0.986
MobileNetV2-50	256	5 M	81 fps	0–80 m	0.116	0.729	4.508	0.178	16.326	0.051	0.853	0.959	0.987
MobileNetV2-140	256	10 M	62 fps	0–80 m	0.104	0.596	4.084	0.161	14.799	0.045	0.877	0.969	0.991
MobileNetV2-140	512	20 M	39 fps	0–80 m	0.100	0.575	4.065	0.158	14.206	0.044	0.883	0.970	0.991
DenseNet-121	256	12 M	41 fps	0–80 m	0.088	0.461	3.780	0.140	12.747	0.040	0.905	0.978	**0.994**
Fig. 2 CNN	256	3 M	**88 fps**	0–50 m	0.098	0.456	3.030	0.150	13.796	0.043	0.888	0.974	0.993
Model 1	256	3 M	59 fps	0–50 m	0.089	0.380	2.843	0.138	12.712	0.039	0.907	0.980	0.994
Model 2	256	6 M	57 fps	0–50 m	0.086	0.370	2.789	0.134	12.186	0.038	0.914	0.981	0.994
Model 3	256	12 M	50 fps	0–50 m	**0.079**	**0.314**	**2.592**	**0.124**	**11.315**	**0.035**	**0.926**	**0.984**	**0.996**
MobileNetV2-50	128	**2 M**	87 fps	0–50 m	0.117	0.592	3.431	0.172	15.892	0.049	0.856	0.961	0.989
MobileNetV2-50	256	5 M	81 fps	0–50 m	0.111	0.560	3.339	0.166	15.139	0.048	0.866	0.965	0.989
MobileNetV2-140	256	10 M	62 fps	0–50 m	0.100	0.463	3.044	0.151	13.810	0.043	0.889	0.974	0.993
MobileNetV2-140	512	20 M	39 fps	0–50 m	0.095	0.434	2.973	0.146	13.172	0.042	0.896	0.975	0.993
DenseNet-121	256	12 M	41 fps	0–50 m	0.083	0.335	2.701	0.129	11.722	0.037	0.919	0.983	**0.996**

module has little influence on the total size of the framework and can be used in other methods that tackle SIDE problems. Only the prediction speed is negatively affected by the use of the aforementioned strategy.

An overview of the works that address the SIDE task and that evaluate its approaches in the KITTI Depth [17] can be visualized in Table 9. The results obtained by our pipeline are comparable to those generated by state-of-the-art methods with different types of training procedures. Furthermore, based on Table 1, the proposed CNN has fewer trainable parameters and, due to its prediction speed, it may be applied to real self-driving scenarios.

The depth maps retrieved by the DSN configuration of Table 9 are depicted in Fig. 3(f). Compared to the methods of Figs. 3(c)–3(e), ours is able to better identify the shape of the objects that

Table 7
Behavior of the DSN when subjected to changes in its structure and training. The DSN 3 is the baseline CNN with the Model 3 as encoder. The Pyramid[1] and Pyramid[2] are pyramid-shaped modules implemented in the decoder. CS and Sem refer to pre-training with the disparities of the Cityscapes dataset [135] and the semantic maps of the KITTI [156] and Cityscapes [135] respectively. The $\mathcal{L}^{+}_{BerHu}$ is equal to the $\mathcal{L}_{BerHu}$ expression, yet with $\kappa = 4$.

Method	Loss	Size	Speed	Cap	Abs Rel↓	Sqr Rel↓	RMSE↓	RMSE (log)↓	SILog↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
DSN 3	$\mathcal{L}_{BerHu}$	**12 M**	**50 fps**	0–80 m	0.083	0.434	3.646	0.135	12.326	0.037	0.914	0.980	0.994
DSN 3	$\mathcal{L}_{a} + \mathcal{L}_{BerHu}$	**12 M**	**50 fps**	0–80 m	0.084	0.437	3.621	0.133	12.259	0.037	0.914	0.981	0.994
DSN 3	$\mathcal{L}^{+}_{BerHu}$	**12 M**	**50 fps**	0–80 m	0.083	0.431	3.628	0.133	12.146	0.037	0.915	0.982	0.995
DSN 3 + Pyramid[1]	$\mathcal{L}^{+}_{BerHu}$	**12 M**	39 fps	0–80 m	0.082	0.418	3.593	0.132	12.122	0.037	0.919	0.981	0.995
DSN 3 + Pyramid[1] + Sem	$\mathcal{L}^{+}_{BerHu}$	**12 M**	39 fps	0–80 m	0.081	0.414	3.557	0.133	12.287	0.036	0.916	0.980	0.995
DSN 3 + Pyramid[1] + CS	$\mathcal{L}^{+}_{BerHu}$	**12 M**	39 fps	0–80 m	**0.078**	0.405	3.399	0.125	11.490	**0.034**	**0.926**	**0.985**	0.995
DSN 3 + Pyramid[2] + CS	$\mathcal{L}^{+}_{BerHu}$	**12 M**	32 fps	0–80 m	**0.078**	**0.376**	**3.344**	**0.124**	**11.323**	**0.034**	**0.926**	**0.985**	**0.996**
DSN 3	$\mathcal{L}_{BerHu}$	**12 M**	**50 fps**	0–50 m	0.079	0.314	2.592	0.124	11.315	0.035	0.926	0.984	0.996
DSN 3	$\mathcal{L}_{BerHu} + \mathcal{L}_{a}$	**12 M**	**50 fps**	0–50 m	0.080	0.319	2.594	0.123	11.267	0.035	0.927	0.985	0.996
DSN 3	$\mathcal{L}^{+}_{BerHu}$	**12 M**	**50 fps**	0–50 m	0.078	0.312	2.596	0.122	11.143	0.035	0.927	0.986	0.996
DSN 3 + Pyramid[1]	$\mathcal{L}^{+}_{BerHu}$	**12 M**	39 fps	0–50 m	0.078	0.301	2.553	0.121	11.093	0.034	0.931	0.985	0.996
DSN 3 + Pyramid[1] + Sem	$\mathcal{L}^{+}_{BerHu}$	**12 M**	39 fps	0–50 m	0.077	0.301	3.539	0.123	11.309	0.034	0.927	0.984	0.996
DSN 3 + Pyramid[1] + CS	$\mathcal{L}^{+}_{BerHu}$	**12 M**	39 fps	0–50 m	**0.074**	0.302	2.437	0.116	10.579	**0.032**	**0.937**	**0.988**	0.996
DSN 3 + Pyramid[2] + CS	$\mathcal{L}^{+}_{BerHu}$	**12 M**	32 fps	0–50 m	**0.074**	**0.274**	**2.393**	**0.114**	**10.413**	**0.032**	**0.937**	**0.988**	**0.997**

Table 8
Evaluation of the DSN with the surface normals module in the KITTI Depth dataset [17]. The DSN 3 + Pyramid[1] + CS + SN setup, as well as the Sobel edge detector and the $\mathcal{L}^{+}_{BerHu}$ loss, is standard for all experiments. SN means that the training is carried out with additional surface normals ground truth. The Blur column indicates the presence or absence of a Convolution with Gaussian kernel. ψ_1 and ψ_2 are the constants of the 2.5D loss function employed in the pre-training and training steps respectively.

Method	Size	Speed	Blur	ψ_1	ψ_2	Cap	Abs Rel↓	Sqr Rel↓	RMSE↓	RMSE (log)↓	SILog↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
DSN 3 + Pyramid[1] + CS + SN	**12 M**	**33 fps**	✗	10^6	10^6	0–80 m	**0.075**	0.363	**3.253**	**0.119**	**10.886**	**0.033**	**0.934**	**0.986**	**0.996**
DSN 3 + Pyramid[1] + CS + SN	**12 M**	**33 fps**	✗	10^6	10^7	0–80 m	0.076	**0.360**	3.259	0.120	10.950	0.034	0.932	**0.986**	**0.996**
DSN 3 + Pyramid[1] + CS + SN	**12 M**	32 fps	✓	10^6	10^6	0–80 m	**0.075**	0.365	3.264	0.120	10.947	**0.033**	**0.934**	**0.986**	**0.996**
DSN 3 + Pyramid[1] + CS + SN	**12 M**	32 fps	✓	10^6	10^7	0–80 m	0.081	0.391	3.347	0.126	11.591	0.036	0.924	0.985	**0.996**
DSN 3 + Pyramid[1] + CS + SN	**12 M**	**33 fps**	✗	10^6	10^6	0–50 m	**0.071**	0.267	**2.351**	**0.111**	**10.053**	**0.031**	**0.944**	**0.989**	**0.997**
DSN 3 + Pyramid[1] + CS + SN	**12 M**	**33 fps**	✗	10^6	10^7	0–50 m	0.072	**0.263**	2.365	**0.111**	10.129	0.032	0.942	**0.989**	**0.997**
DSN 3 + Pyramid[1] + CS + SN	**12 M**	32 fps	✓	10^6	10^6	0–50 m	0.072	0.268	2.355	**0.111**	10.113	**0.031**	**0.944**	**0.989**	**0.997**
DSN 3 + Pyramid[1] + CS + SN	**12 M**	32 fps	✓	10^6	10^7	0–50 m	0.077	0.296	2.473	0.118	10.827	0.034	0.934	0.987	0.996

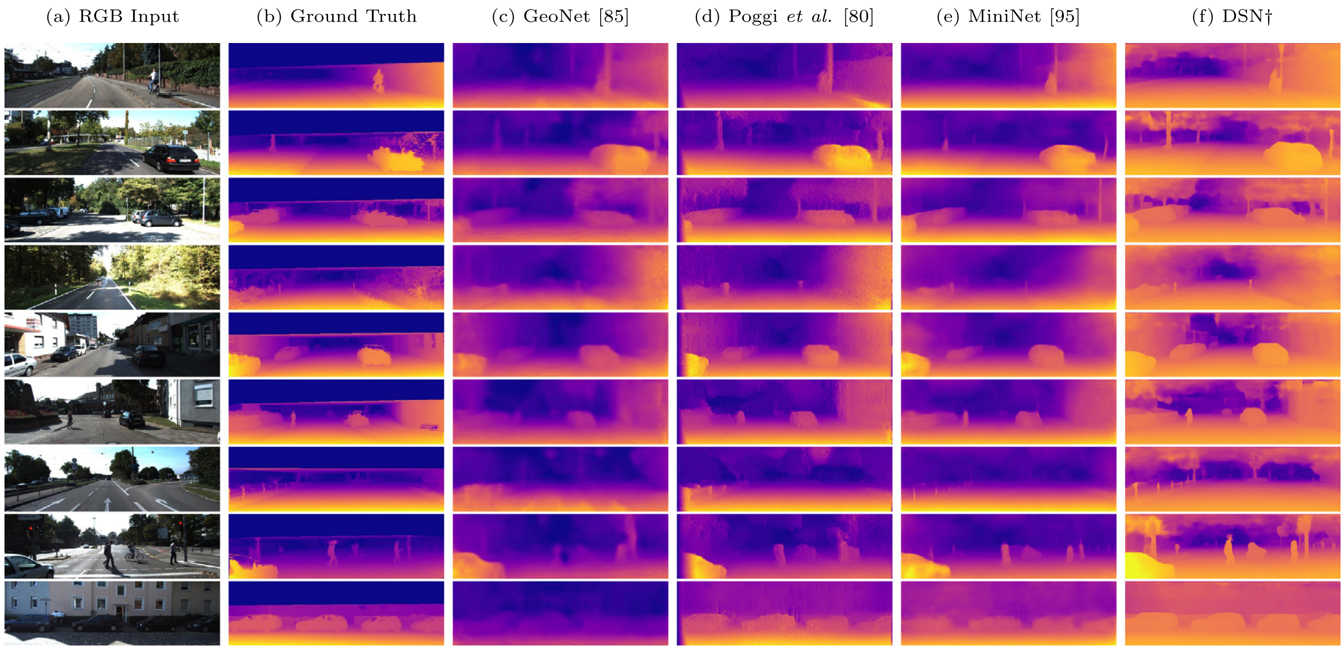


Fig. 3. Qualitative comparison between the predictions from our best approach (DSN†) and from recent self-supervised methods that address the SIDE task. The input images and the ground truth belong to the KITTI Depth dataset [17]. The reference depth maps are interpolated for visualization purposes.

compose the scene, outputting more smoothly and well-outlined contours, even in regions close to occlusions. Also, in the depth maps of Fig. 3(f), we may notice a higher amount of inferred artifacts, mainly in the more distant regions of the scene, which are more difficult to be comprehended by the other methods.

On the other hand, Fig. 4 compares the predictions from our approach with the ones produced by recent supervised frameworks. As can be seen in such a figure, the DSN outputs high-definition depth maps with fewer blurred regions under the sky limit. Through Fig. 5, it is possible to verify that the 3D point clouds reconstructed from the predictions of the DSN have a

Table 9
Performances of our framework and other works in the SIDE literature according to the KITTI Depth [17] metrics. Legend: DSN† — DSN setup that yields the most accurate overall results; MV — monocular video; SP — stereo pair; CM — camera motion; SI — single image; GT — ground truth; SL — semantic labels; SN — surface normals; DSO — direct sparse odometry; SGM — semi-global matching; DM — disparity map; Cal. Cam. — calibrated camera.

Method	Training Input/Auxiliar Info	Cap	Abs Rel↓	Sqr Rel↓	RMSE↓	RMSE *log*↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Saxena et al. [157]	SI/GT Depth	0–80 m	0.280	3.012	8.734	0.361	0.601	0.820	0.926
Eigen et al. [28]	SI/GT Depth	0–80 m	0.203	1.548	6.307	0.282	0.702	0.898	0.967
Liu et al. [158]	SI/GT Depth	0–80 m	0.201	1.584	6.471	0.273	0.680	0.898	0.967
Zhou et al. [59]	MV/-	0–80 m	0.198	1.836	6.565	0.275	0.718	0.901	0.960
UnDeepVO [68]	SP/Cal. Cam.	0–80 m	0.183	1.730	6.570	0.268	–	–	–
Yang et al. [159]	MV/-	0–80 m	0.182	1.481	6.501	0.267	0.725	0.906	0.963
AdaDepth [160]	SI and Synt. SI/Synt. GT Depth (self)	0–80 m	0.167	1.257	5.578	0.237	0.771	0.922	0.971
Klodt et al. [121]	MV/-	0–80 m	0.166	1.490	5.988	–	0.778	0.919	0.966
Mahjourian et al. [86]	MV/-	0–80 m	0.163	1.240	6.220	0.250	0.762	0.916	0.968
LEGO [84]	MV/-	0–80 m	0.162	1.352	6.276	0.252	–	–	–
Poggi et al. [80]	SP/-	0–80 m	0.153	1.363	6.030	0.252	0.789	0.918	0.963
GeoNet [85]	MV/-	0–80 m	0.153	1.328	5.737	0.232	0.802	0.934	0.972
Pilzer et al. [79]	SP/Cal. Cam.	0–80 m	0.152	1.388	6.016	0.247	0.789	0.918	0.965
DDVO [87]	MV/-	0–80 m	0.151	1.257	5.583	0.228	0.810	0.936	0.974
DF-Net [122]	MV/-	0–80 m	0.150	1.124	5.507	0.223	0.806	0.933	0.973
Ranjan et al. [91]	MV/-	0–80 m	0.148	1.149	5.464	0.226	0.815	0.935	0.973
MiniNet et al. [95]	MV/-	0–50 m	0.141	1.080	5.264	0.216	0.825	0.941	0.976
Struct2depth [88]	MV/Cal. Cam. (Only intrinsics matrix)	0–80 m	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Elkerdawy et al. [82]	SP/Cal. Cam.	0–80 m	0.136	–	5.891	–	0.827	–	–
PHN et al. [161]	SI/GT Depth	0–80 m	0.136	–	4.082	0.164	0.864	0.966	0.989
Zhan FullNYU [162]	SP/CM	0–80 m	0.135	1.132	5.585	0.229	0.820	0.933	0.971
Wong et al. [163]	SP/- (Cal. needed in test)	0–80 m	0.133	1.126	5.515	0.231	0.826	0.934	0.969
3Net (ResNet-50) [71]	Trinocular setup/-	0–80 m	0.129	0.996	5.281	0.223	0.831	0.939	0.974
StrAT [164]	SI/Cal. Cam.	0–80 m	0.128	1.019	5.403	0.227	0.827	0.935	0.971
EPC++ [165]	MV/Cal. Cam. (Only intrinsics matrix)	0–80 m	0.128	0.935	5.011	0.209	0.831	0.945	0.979
Gordon et al. [166]	MV/-	0–80 m	0.124	0.930	5.120	0.206	0.851	0.950	0.978
Sparse-to-Continuous [148]	SI/GT Depth	0–80 m	0.123	0.641	4.524	0.199	0.881	0.966	0.986
SA-Attention [129]	MV/Cal. Cam. (Only intrinsics matrix)	0–80 m	0.121	0.837	4.945	0.197	0.853	0.955	0.982
3Net (VGG) [71]	Trinocular setup/-	0–80 m	0.119	1.201	5.888	0.208	0.844	0.941	0.978
SceneNet [167]	SI and SP/SL	0–80 m	0.118	0.905	5.096	0.211	0.839	0.945	0.977
Su et al. [168]	SI/GT Depth	0–80 m	0.117	–	4.251	0.174	0.894	0.971	0.984
SDNet [128]	SI/GT Depth and SL	0–80 m	0.116	0.945	4.916	0.208	0.861	0.952	0.968
Cao et al. [34]	SI/GT Depth	0–80 m	0.115	–	4.712	0.198	0.887	0.963	0.982
Monodepth [27]	SP/Cal. Cam.	0–80 m	0.114	0.898	4.935	0.206	0.861	0.949	0.976
LSIM [126]	SP/Cal. Cam.	0–80 m	0.113	0.898	5.048	0.208	0.853	0.948	0.976
Kuznietsov et al. [61]	SP/Cal. Cam. and GT Depth	0–80 m	0.113	0.741	4.621	0.189	0.862	0.960	0.986
SuperDepth [58]	SP/Cal. Cam.	0–80 m	0.112	0.875	4.958	0.207	0.852	0.947	0.977
DeepLabV3+ (F10) [169]	SI/GT Depth and Cal. Cam.	0–80 m	0.110	0.666	4.186	0.168	0.880	0.966	0.988
Monodepth2 [60]	SP or MV or both/Cal. Cam.	0–80 m	0.106	0.806	4.630	0.193	0.876	0.958	0.980
PackNet-SfM [93]	MV/Velocity term (optional)	0–80 m	0.104	0.758	4.386	0.182	0.895	0.964	0.982
Guizilini et al. [10]	MV/Pre-trained network with SL	0–80 m	0.100	0.761	4.270	0.175	0.902	0.965	0.982
CFA [170]	SI/GT Depth and SL	0–80 m	0.100	0.601	4.298	0.174	0.874	0.966	0.989
Refine&Distill [171]	SP/Cal. Cam.	0–80 m	0.098	0.831	4.656	0.202	0.882	0.948	0.973
DVSO [63]	SP/GT Stereo DSO Depth	0–80 m	0.097	0.734	4.442	0.187	0.888	0.958	0.980
SOM [172]	SI/GT Depth	0–80 m	0.097	0.398	3.007	0.133	0.913	0.985	0.997
Depth Hints [173]	SP/Cal. Cam.	0–80 m	0.096	0.710	4.393	0.185	0.890	0.962	0.981
monoResMatch [67]	SP/Proxy GT with SGM	0–80 m	0.096	0.673	4.351	0.184	0.890	0.961	0.981
VGG16-UNet [72]	SI/Synthetic SP-DM and Cal. Cam.	0–80 m	0.096	0.641	4.095	0.168	0.892	0.967	0.986
SemiDepth [62]	SP/Cal. Cam. and GT Depth	0–80 m	0.096	0.552	3.995	0.152	0.892	0.972	0.992
SVS [66]	SP/Cal. Cam.	0–80 m	0.094	0.626	4.252	0.177	0.891	0.965	0.984
DenseDepth [15]	SI/GT Depth	0–80 m	0.093	0.589	4.170	0.171	0.886	0.965	0.986
VOMonodepth [64]	SP/Sparse Depths and Cal. Cam.	0–80 m	0.091	0.548	3.690	0.181	0.892	0.956	0.979
FAL-netB49 [78]	SP/Cal. Cam.	0–80 m	0.088	0.547	4.004	0.175	0.898	0.966	0.984
Monodepth2-Self [77]	SP or MV or both/Cal. Cam.	0–80 m	0.083	–	3.682	–	0.919	–	–
BA-Net [174]	MV/Cal. Cam.	0–80 m	0.083	–	3.640	0.134	–	–	–
DSN†	SI/GT and SN	0–80 m	0.075	0.363	3.253	0.119	0.934	0.986	0.996
VNL [11]	SI/GT Depth and VN	0–80 m	0.072	–	3.258	0.117	0.938	0.990	**0.998**
DORN [26]	SI/GT Depth	0–80 m	0.072	0.307	**2.727**	0.120	0.932	0.984	0.994
BTS [13]	SI/GT Depth	0–80 m	**0.059**	**0.245**	2.756	**0.096**	**0.956**	**0.993**	**0.998**
Zhou et al. [59]	MV/-	0–50 m	0.190	1.436	4.975	0.258	0.735	0.915	0.968
Kim et al. [175]	SI/GT Depth	0–50 m	0.177	–	6.570	0.254	–	–	–
Garg et al. [57]	SP/CM	1 – 50 m	0.169	1.080	5.104	0.273	0.740	0.904	0.962
Mahjourian et al. [86]	MV/-	0–50 m	0.155	0.927	4.549	0.231	0.781	0.931	0.975
GeoNet [85]	MV/-	0–50 m	0.147	0.936	4.348	0.218	0.810	0.941	0.977
Poggi et al. [80]	SP/-	0–50 m	0.145	1.014	4.608	0.227	0.813	0.934	0.972
Pilzer et al. [79]	SP/Cal. Cam.	0–50 m	0.144	1.007	4.660	0.240	0.793	0.923	0.968
MiniNet et al. [95]	MV/-	0–50 m	0.135	0.839	4.067	0.205	0.838	0.947	0.978
Wong et al. [163]	SP/- (Cal. needed in test)	0–50 m	0.126	1.832	4.172	0.217	0.840	0.941	0.973
Sparse-to-Continuous [148]	SI/GT Depth	0–50 m	0.119	0.444	3.097	0.180	0.893	0.974	0.990
Monodepth [27]	SP/Cal. Cam.	0–50 m	0.108	0.657	3.729	0.194	0.873	0.954	0.979
Kuznietsov et al. [61]	SP/Cal. Cam. and GT Depth	1 – 50 m	0.108	0.595	3.518	0.179	0.875	0.964	0.988
Cao et al. [34]	SI/GT Depth	0–50 m	0.107	–	3.605	0.187	0.898	0.966	0.984

(*continued on next page*)

Table 9 (*continued*).

Method	Training Input/Auxiliar Info	Cap	Abs Rel↓	Sqr Rel↓	RMSE↓	RMSE *log*↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Su et al. [168]	SI/GT Depth	0–50 m	0.107	–	3.440	0.172	0.900	0.976	0.990
LSIM et al. [126]	SP/Cal. Cam.	0–50 m	0.106	0.653	3.790	0.195	0.867	0.954	0.979
CFA [170]	SI/GT Depth and SL	0–50 m	0.096	0.482	3.338	0.166	0.886	0.980	0.995
Gan et al. [176]	SI/GT Depth	0–50 m	0.094	0.552	3.133	0.165	0.898	0.967	0.986
DSN†	SI/GT and SN	0–50 m	0.071	0.267	2.351	0.111	0.944	0.989	0.997
DORN [26]	SI/GT Depth	0–50 m	0.071	0.268	2.271	0.116	0.936	0.985	0.995
BTS [13]	SI/GT Depth	0–50 m	**0.056**	**0.169**	**1.925**	**0.087**	**0.964**	**0.994**	**0.999**

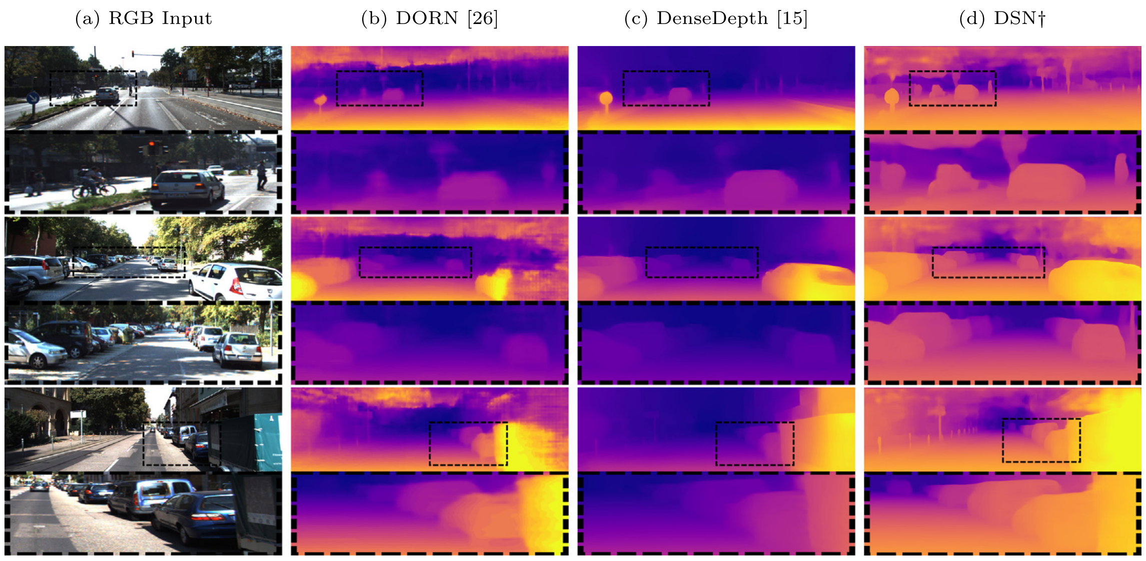


Fig. 4. Differences between the depth maps retrieved by the best DSN configuration (DSN†) and by SIDE approaches in the state-of-the-art with a supervised training pipeline. The input RGB images are part of the KITTI Depth dataset [17].

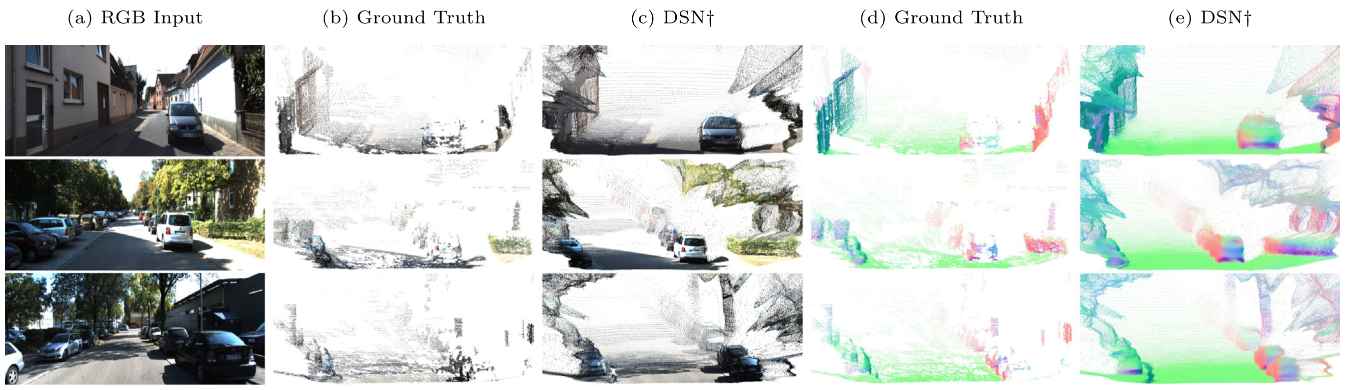


Fig. 5. Qualitative analysis of the point clouds reconstructed from the DSN predictions. DSN† indicates the DSN setup that leads to the most accurate overall metrics. The RGB input and the reference point clouds with depth data are taken from the KITTI Depth dataset [17]. However, the reference surface normals information are generated by the authors with the aid of the algorithm introduced by Fouhey et al. [53].

denser distribution than the point clouds created with the sparse and noisy ground truth. The 3D maps from Fig. 5 also show that our framework is robust to data noise and it is capable of modeling the shape of the objects with few errors.

4.4.4. Depth completion studies

To study the performance of the DSN on depth completion applications, we employed the sampling technique based on the categorical distribution of Eq. (19) to feed the CNN with RGBD maps. The results of such experiments, exposed in Table 10, enforce the theory that the more LiDAR points are presented to the network input, the more accurate the retrieved depth maps are. This can be visualized in the graphs of Fig. 6, in which the error metrics decrease and the precision metrics increase as more sparse depth data is added to the network input.

Based on Table 10 metrics, we can conclude that the 3D space cues yielded by the surface normals reference maps cause a slightly positive influence in the densification process. Moreover, the best performance of our depth completion pipeline is compared to the results from other approaches in the literature in Table 11.

Although our CNN is capable of densifying depth maps at a high frame rate, it also generates dense maps with a quality that surpasses that of other state-of-the-art works. Our most accurate depth completion results are also depicted in Fig. 7, in which both the experiments regarding 200 and 500 sparse depth data, fed into the DSN, achieve high-quality dense depth maps.

From the graphs in Fig. 8, we analyze the effects caused on our depth completion CNN when it is fed with sparse data generated by the Monodepth2 pre-trained with the mono +

Table 10
Evaluation of our depth completion pipeline according to Eigen Split applied to the KITTI Depth dataset [17]. The DSN 3 + Pyramid[1] + CS configuration, besides the $\mathcal{L}^{+}_{BerHu}$ function, is taken as default for the first experiments. The DSN 3 + Pyramid[1] + CS + SN setup, along with the edge detector, is analyzed afterward.

Method	Cap	Samples	Abs Rel↓	RMSE↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
DSN 3 + Pyramid[1] + CS	80	20	0.049	2.829	0.021	0.961	0.991	0.997
DSN 3 + Pyramid[1] + CS	80	50	0.035	2.397	0.015	0.976	0.994	0.998
DSN 3 + Pyramid[1] + CS	80	100	0.029	2.184	0.013	0.981	0.995	0.998
DSN 3 + Pyramid[1] + CS	80	200	0.024	1.926	0.010	0.987	0.996	0.999
DSN 3 + Pyramid[1] + CS	80	500	**0.019**	1.672	**0.008**	**0.991**	**0.997**	**0.999**
DSN 3 + Pyramid[1] + CS + SN	80	200	0.024	1.863	0.010	0.987	0.996	0.999
DSN 3 + Pyramid[1] + CS + SN	80	500	**0.019**	**1.588**	**0.008**	**0.991**	**0.997**	**0.999**
DSN 3 + Pyramid[1] + CS	50	20	0.045	1.922	0.019	0.970	0.993	0.998
DSN 3 + Pyramid[1] + CS	50	50	0.032	1.595	0.014	0.982	0.995	0.999
DSN 3 + Pyramid[1] + CS	50	100	0.026	1.432	0.011	0.986	0.996	0.999
DSN 3 + Pyramid[1] + CS	50	200	0.021	1.252	0.009	0.990	0.997	0.999
DSN 3 + Pyramid[1] + CS	50	500	**0.017**	1.075	**0.007**	**0.993**	**0.998**	**0.999**
DSN 3 + Pyramid[1] + CS + SN	50	200	0.022	1.247	0.09	0.990	0.997	0.999
DSN 3 + Pyramid[1] + CS + SN	50	500	0.018	**1.051**	0.008	**0.993**	**0.998**	**0.999**

Table 11
Quantitative comparison of the results from our framework and the results from other depth completion techniques. The metrics are computed considering the Eigen Split and the KITTI Depth dataset [17]. We apply cap 80 in all the analysis.

Method	Samples	Abs Rel↓	RMSE↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
full-MAE [177]	∼650	0.179	7.14	0.709	0.888	0.956
Liao et al. [35]	225	0.113	4.50	0.874	0.960	0.984
Ma et al. [7]	200	0.083	3.851	0.919	0.970	0.986
Ma et al. [7]	500	0.073	3.378	0.935	0.976	0.989
MC + SPN [112]	500	0.059	3.248	0.944	0.977	0.989
SPN [178]	500	0.063	3.243	0.943	0.978	0.991
Mirror connection (MC) [112]	500	0.051	3.049	0.953	0.979	0.990
CSPN [112]	500	0.049	3.029	0.955	0.980	0.990
MC + CSPN [112]	500	0.044	2.977	0.957	0.980	0.991
MC + CSPN + ACSPF [112]	500	0.042	2.843	0.961	0.982	0.992
DSN 3 + Pyramid[1] + CS + SN	200	0.024	1.863	0.987	0.996	**0.999**
DSN 3 + Pyramid[1] + CS + SN	500	**0.019**	**1.588**	**0.991**	**0.997**	**0.999**

Table 12
Evaluation of DSN predictions applied to the CNN-SVO framework according to the RMSE metric. We used the first 11 sequences of the KITTI Odometry dataset [146] to generate the results. Legend: SVO — Semi-Direct Visual Odometry; DSO — Direct Sparse Odometry; DSN† — DSN setup, without the surface normals module, that produces the best results; BA — Bundle Adjustment; DSN‡ — DSN setup, with the surface normals module, that produces the most accurate results.

Sequence	SVO [179]	CNN-SVO + BA [5]	CNN-SVO [5] + BA + DSN‡	CNN-SVO [5] + BA + DSN†	CNN-SVO [5] + DSN†	DSO [180]	ORB-SLAM [181] (without loop closure)
00	–	17.5269	**16.9438**	19.7020	42.9432	113.1838	77.9502
01	–	–	–	–	–	–	–
02	–	50.5119	**14.9513**	15.7180	45.0136	116.8108	41.0064
03	–	3.4588	1.9309	3.4340	12.4293	1.3943	**1.0182**
04	58.3970	2.4414	2.3090	4.1912	5.6721	0.422	**0.9302**
05	–	8.1513	**7.1224**	9.4541	30.2511	47.4605	40.3542
06	–	11.5091	**8.7421**	11.4059	22.5910	55.6173	52.2282
07	–	6.5141	**1.9292**	2.3099	8.0324	16.7192	16.546
08	–	10.9755	**8.3792**	9.4699	13.3337	111.0832	51.6215
09	–	**10.6873**	10.9941	15.0341	28.6044	52.2251	58.1742
10	–	4.8354	**2.3163**	2.8371	4.1040	11.090	18.4765

stereo configuration. The behavior of the curve related to the Monodepth2 approach is increasing for the Abs Rel metric and decreasing for the $\delta < 1.25[\%]$ since the more sparse data are sampled from the depth maps produced by Monodepth2, the more the errors of these maps influence the DSN results.

However, the graphs of Fig. 8 also indicate that the performance of our network (without the surface normals module) is improved by using fewer amounts of depth data sampled from the Monodepth2 results. Therefore, the DSN may be successfully applied in real-world situations, where there is no ground truth available.

4.4.5. Visual odometry results

Once depth data are estimated, they can be incorporated into VO/SLAM systems to improve the quality of visual odometry and the reconstructed map. With respect to Semi-Direct Visual Odometry (SVO) methods, the accurate depth information obtained from SIDE approaches can be added to enhance the invariance of the tracking algorithm to illumination changes in the input overexposed or underexposed images [5]. Moreover, with the aid of such additional cues, the mapping points step of the SVO framework is performed with a smaller uncertainty, leading to an increase in the matching features accuracy and to a decrease in the convergence time of this feature correspondence mechanism [5].

CNN-SLAM and CNN-SVO are examples of works in which the incorporation of the depth maps, predicted by a deep neural network, into the processing pipeline is done successfully. More specifically, the former work used a ResNet-based FCN [29] whereas the latter one used the Monodepth [27] to generate the depth estimates. Since the predictions from the DSN are better than those produced by the cited methods, we fed the CNN-SVO framework with our depth estimations to better determine

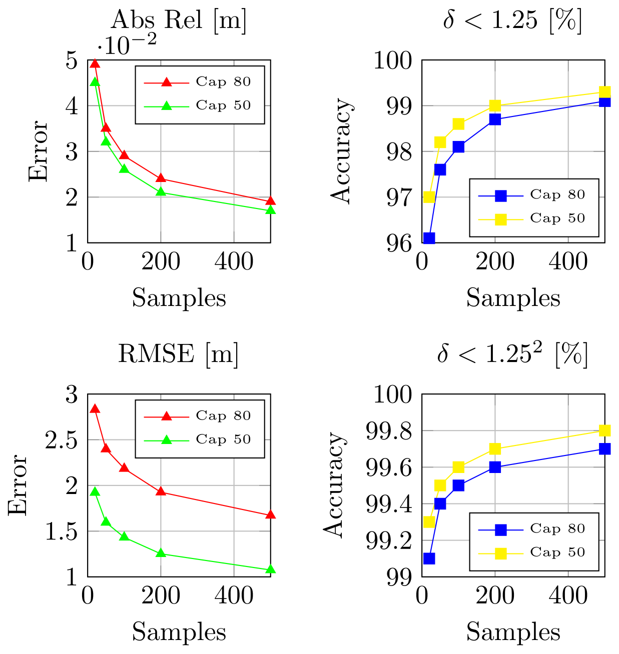


Fig. 6. Errors and accuracy evolution of the DSN predictions according to the number of input sparse samples. We used the ground truth of the KITTI Depth dataset [17] to present the results.

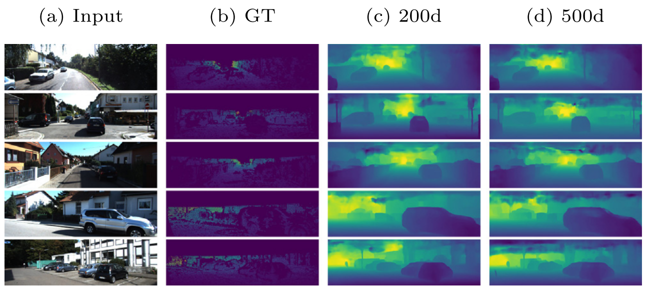


Fig. 7. Qualitative evaluation of our depth completion framework based on the noisy and sparse ground truth of the KITTI Depth dataset [17]. Legend: Input — input RGB image; GT — sparse ground truth; d — sparse input data.

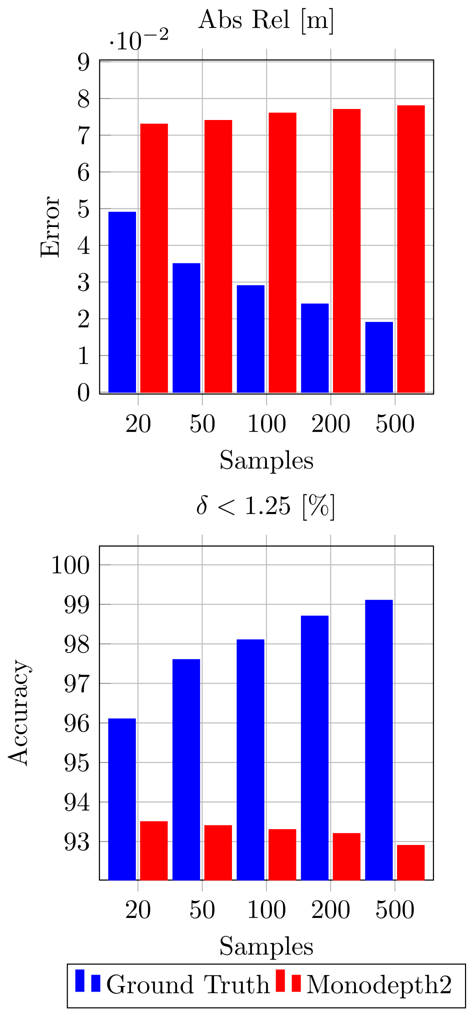


Fig. 8. Comparison between the errors and precision obtained by our depth completion framework (without the surface normals module) when it is introduced to sparse data from the ground truth and the depth maps generated by the Monodepth2 [60].

corresponding features. Using different setups of our CNN, we evaluated the improvements that this modification in mapping the points of consecutive keyframes brings to the SVO task.

Particularly, we compared the results obtained by the joint application of the CNN-SVO and variations of the DSN with the CNN-SVO alone [5], the Direct Sparse Odometry (DSO) [180], the SVO [179] and the ORB-SLAM without loop closure [181] through Table 12. Such experiments are conducted with the help of the 11 sequences with ground truth trajectories provided by the KITTI Odometry dataset [146]. Also, we employed the Absolute Trajectory Error (ATE) and the RMSE metric to expose the aforementioned results. In Table 12, the spaces that are not filled with metric values in meters are related to the techniques that are not able to compute the entire sequence due to failures in the tracking algorithm. As it is proposed in the work of Loo et al. [5], we stored the ATEs with a median of 5 runs.

Analyzing the metrics from Table 12, it is plausible to conclude that when the CNN-SVO technique, along with the Bundle Adjustment algorithm (BA), employs the depth maps retrieved by the DSN, it is capable of tracking most of the addressed sequences in a more precise way. This occurs due to the initialization of new map points in the CNN-SVO pipeline with lower depth uncertainties. Furthermore, the results from Table 12 also indicate that when the CNN-SVO is combined with the DSN configuration that leverages surface normals cues, it provides the best overall metric values.

The trajectories illustrated in Fig. 9 are the aligned qualitative results of our best visual odometry approach in the KITTI Odometry sequences. Such trajectories confirm the metrics exposed in Table 12. The scale drift values from Fig. 10 show that our tracking results have a scale close to the ground truth. However, the small lags of scale occur due to the distortions suffered by the output depth maps of the DSN, which are resized to be applied in the CNN-SVO framework.

4.5. *Indoor ablation studies*

4.5.1. *Structural studies*

To analyze the behavior of the DSN 3 when it is exposed to other types of scenes, we also conducted ablation studies with indoor datasets, which are composed of images with different scales than those present in the outdoor datasets. At first, we modified the loss function and the decoder structure of our framework to determine an optimized combination. Regarding the RMSE metric, Table 13 indicates that both the attention term and the modification in the constant κ of the BerHu function do

Table 13
Performance of the proposed SIDE method for different architecture configurations and training strategies. The feature extraction network of the DSN 3 is the developed Model 3. Both pyramid modules, besides the loss function $\mathcal{L}^+_{BerHu}$, are implemented as in the analyzes involving the outdoor datasets. The terms Sem and Indoor indicate the pre-training of the DSN 3 with semantic maps and extra depth maps, respectively, provided by the indoor datasets.

Method	Loss	Size	Speed	Abs Rel↓	Sqr Rel↓	RMSE↓	RMSE (log)↓	SILog↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
DSN 3	$\mathcal{L}_{BerHu}$	**12 M**	**51 fps**	0.169	0.136	0.523	0.208	17.143	0.070	0.768	0.934	0.978
DSN 3	$\mathcal{L}_a + \mathcal{L}_{BerHu}$	**12 M**	**51 fps**	0.179	0.144	0.540	0.216	17.315	0.073	0.746	0.926	0.977
DSN 3	$\mathcal{L}^+_{BerHu}$	**12 M**	**51 fps**	0.179	0.151	0.538	0.215	17.277	0.073	0.750	0.924	0.976
DSN 3	$\mathcal{L}_1$	**12 M**	**51 fps**	0.180	0.151	0.548	0.217	17.588	0.073	0.746	0.927	0.978
DSN 3 + Pyramid^1	$\mathcal{L}_{BerHu}$	**12 M**	41 fps	0.171	0.133	0.521	0.207	16.605	0.070	0.758	0.936	0.982
DSN 3 + Pyramid^1	$\mathcal{L}^+_2$	**12 M**	41 fps	0.175	0.138	0.520	0.207	16.462	0.071	0.757	0.934	0.981
DSN 3 + Pyramid^1	$\mathcal{L}^+_1$	**12 M**	41 fps	0.167	0.130	0.519	0.207	16.859	0.069	0.763	0.933	0.981
DSN 3 + Pyramid^2	$\mathcal{L}^+_1$	**12 M**	32 fps	0.174	0.143	0.529	0.210	16.896	0.071	0.756	0.933	0.979
DSN 3 + Pyramid^2 + Sem	$\mathcal{L}^+_1$	**12 M**	32 fps	0.174	0.139	0.522	0.211	17.231	0.071	0.760	0.932	0.977
DSN 3 + Pyramid^1 + Indoor	$\mathcal{L}^+_1$	**12 M**	41 fps	**0.142**	**0.098**	**0.445**	**0.177**	**14.197**	**0.059**	**0.823**	**0.954**	**0.987**

Table 14
Analysis of our pipeline that leverages surface normals cues. The configuration DSN 3 + Pyramid^1 + Indoor + SN, along with the $\mathcal{L}^+_1$ loss, is maintained in all experiments. Legend: DSN 3 — DSN with Model 3 as encoder; Pyramid^1 — decoder's pyramid-shaped module; Indoor — pre-training with indoor datasets; SN — surface normals ground truth; Edge — Sobel edge detector; Blur — convolution with Gaussian kernel; ψ_1 and ψ_2 — constants of the 2.5D loss function used in the pre-training and training steps respectively.

Method	Size	Speed	Edge	Blur	ψ_1	ψ_2	Abs Rel	Sqr Rel	RMSE	RMSE (log)	SILog	log10	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	**35 fps**	✗	✗	✗	10^0	0.138	0.096	0.444	0.177	14.227	0.058	0.825	0.954	0.986
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	**35 fps**	✗	✗	✗	10^6	0.137	0.096	0.441	0.176	14.171	0.058	0.825	0.954	0.986
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	34 fps	✓	✗	✗	10^6	0.137	0.094	0.439	0.175	14.134	0.058	0.826	0.954	0.986
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	34 fps	✓	✗	✗	10^7	0.138	0.095	0.442	0.176	14.171	0.058	0.825	0.956	0.986
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	34 fps	✓	✗	✗	10^5	0.139	0.096	0.443	0.177	14.289	0.059	0.822	0.953	0.986
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	34 fps	✓	✗	10^6	10^6	0.134	0.091	0.435	0.173	13.917	0.057	0.831	0.958	0.986
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	34 fps	✓	✗	10^7	10^7	0.134	0.089	0.434	0.172	13.841	0.057	0.831	0.957	0.987
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	34 fps	✓	✗	10^6	10^7	**0.132**	0.088	**0.429**	**0.171**	**13.732**	**0.056**	**0.834**	**0.959**	0.987
DSN 3 + Pyramid^1 + Indoor + SN	**12 M**	33 fps	✓	✓	10^6	10^7	**0.132**	**0.086**	0.431	**0.171**	13.793	**0.056**	0.833	**0.959**	**0.988**

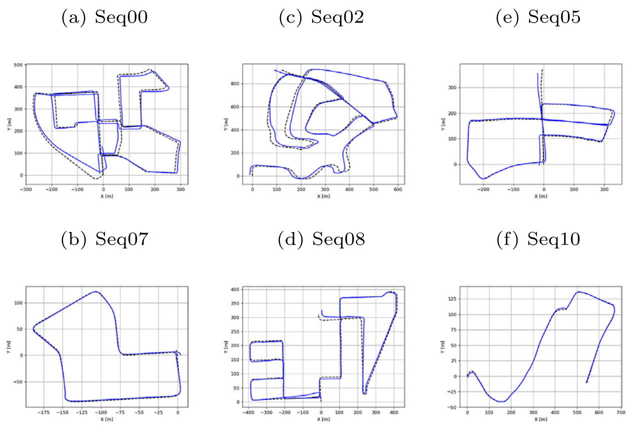


Fig. 9. Qualitative results of the camera trajectory estimations obtained on the KITTI Odometry [146] sequences 00, 02, 05, 07, 08 and 10 by the CNN-SVO + BA jointly with the DSN and the surface normals module. The black dotted and solid blue trajectories represent the ground truth and the predictions respectively.

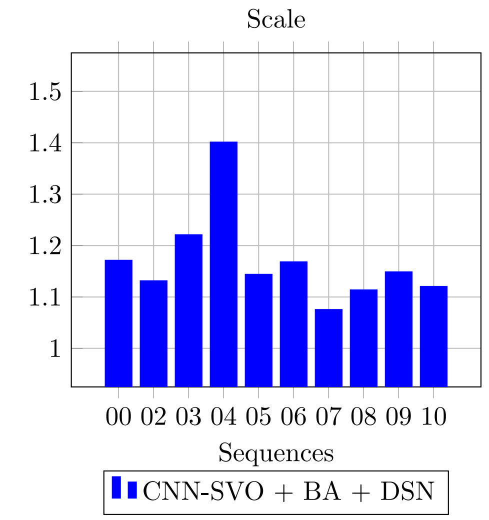


Fig. 10. Scale drift profile of the tracking results generated by the CNN-SVO + BA + DSN (with the surface normals module). The sequences are taken from the KITTI Odometry dataset [146].

not bring benefits to the network predictions since, in comparison to the KITTI Depth [17], the NYU Depth V2 dataset [18] is denser both in regions closer and further from the scene. Due to the denser profile of the ground truth of indoor datasets, simple loss functions, such as $\mathcal{L}^+_1$ and $\mathcal{L}^+_2$, are capable of handling the pixel distribution of the reference maps and update the network weights in a faster and more efficient way.

As also can be inferred from Table 13, the Pyramid^1 module enhances the value of most of the metrics. However, the block Pyramid^2 is not advantageous for the DSN 3 predictions once, for the NYU Depth V2 [18], the upsampled feature maps from the initial convolution of the network decoder do not present the necessary monocular cues to support the estimation of the output depth map. Moreover, we utilized the semantic maps, with 13 fine annotated labels, of the NYU Depth V2 [18] and SUNRGBD [136] datasets, as well as the depth ground truth of all indoor datasets covered in this work to pre-train the proposed framework. As in the experiments addressing outdoor data, we conclude that the pre-trained weights for semantic segmentation help our approach to reason about high-frequency regions of the image, yet the features learned from the pre-training of the DSN 3 on the same task leads to more accurate results.

4.5.2. Surface normals analysis

Considering indoor scenes, the results obtained from the experiments involving our approach that is trained with both depth and surface normals reference maps are presented in Table 14. From the RMSE metric, it is possible to evidence that the application of the edge detector, implemented in the surface normals

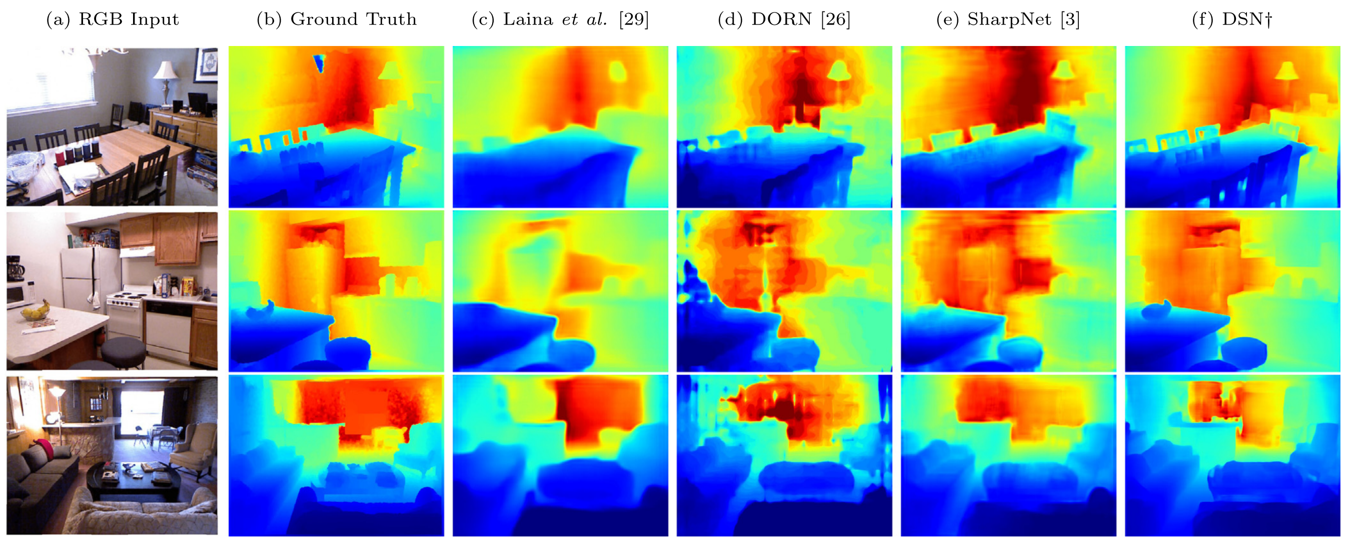


Fig. 11. Comparison between the depth predictions of indoor scenes from our SIDE pipeline and other supervised methods in the state-of-the-art. The DSN† approach is related to the DSN configuration that produces the most accurate overall results. The input images and the ground truth come from the NYU Depth V2 dataset [18]. The ground truth is interpolated for visualization purposes.

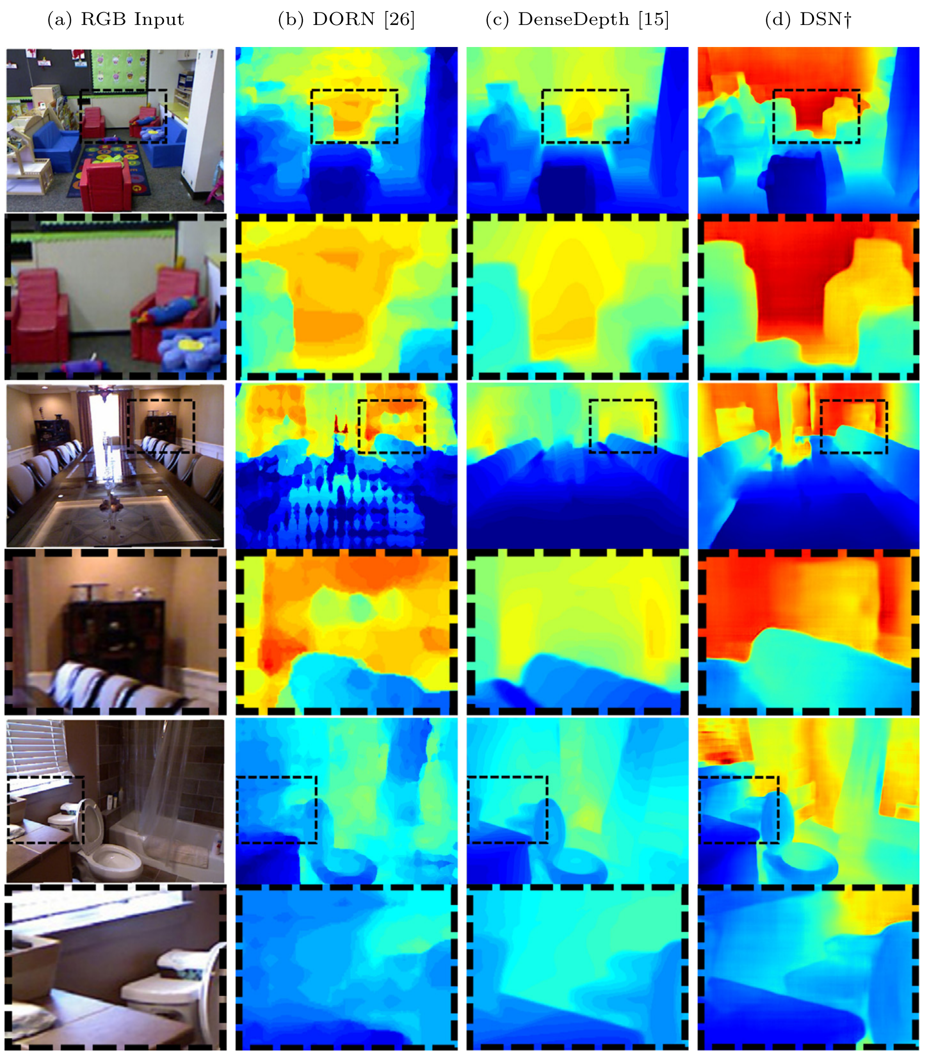


Fig. 12. Detailed evaluation of the predictions of our method and the ones retrieved by recent supervised approaches. DSN† refers to the best DSN setup and the RGB inputs compose the NYU Depth V2 dataset [18].

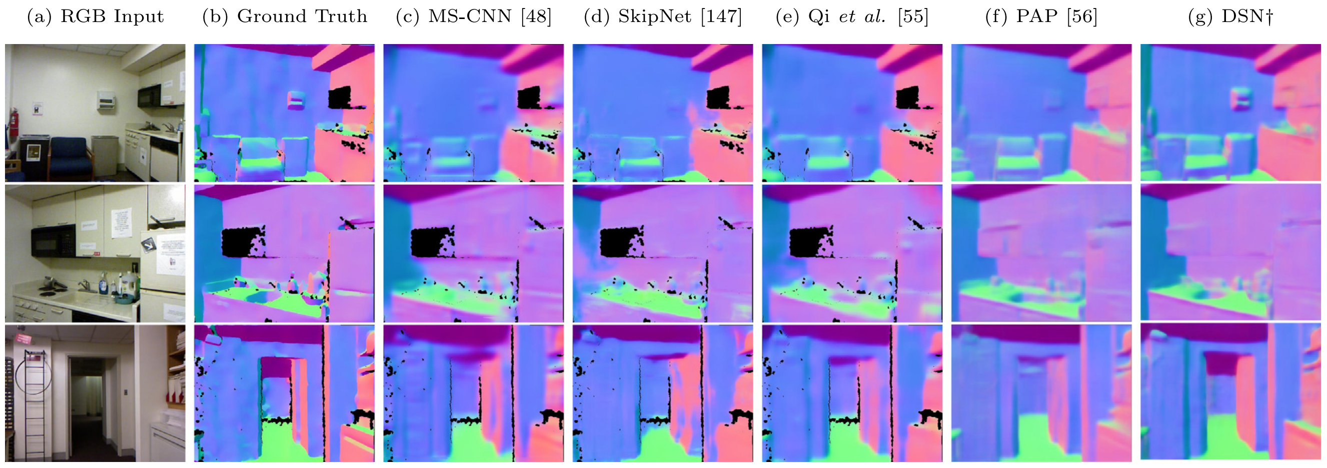


Fig. 13. Qualitative comparison of the surface normals maps retrieved by our framework and by state-of-the-art techniques. DSN† corresponds to the best DSN configuration. The RGB inputs are part of the NYU Depth V2 dataset [18]. We produced the ground truth images through the method of Fouhey et al. [53].

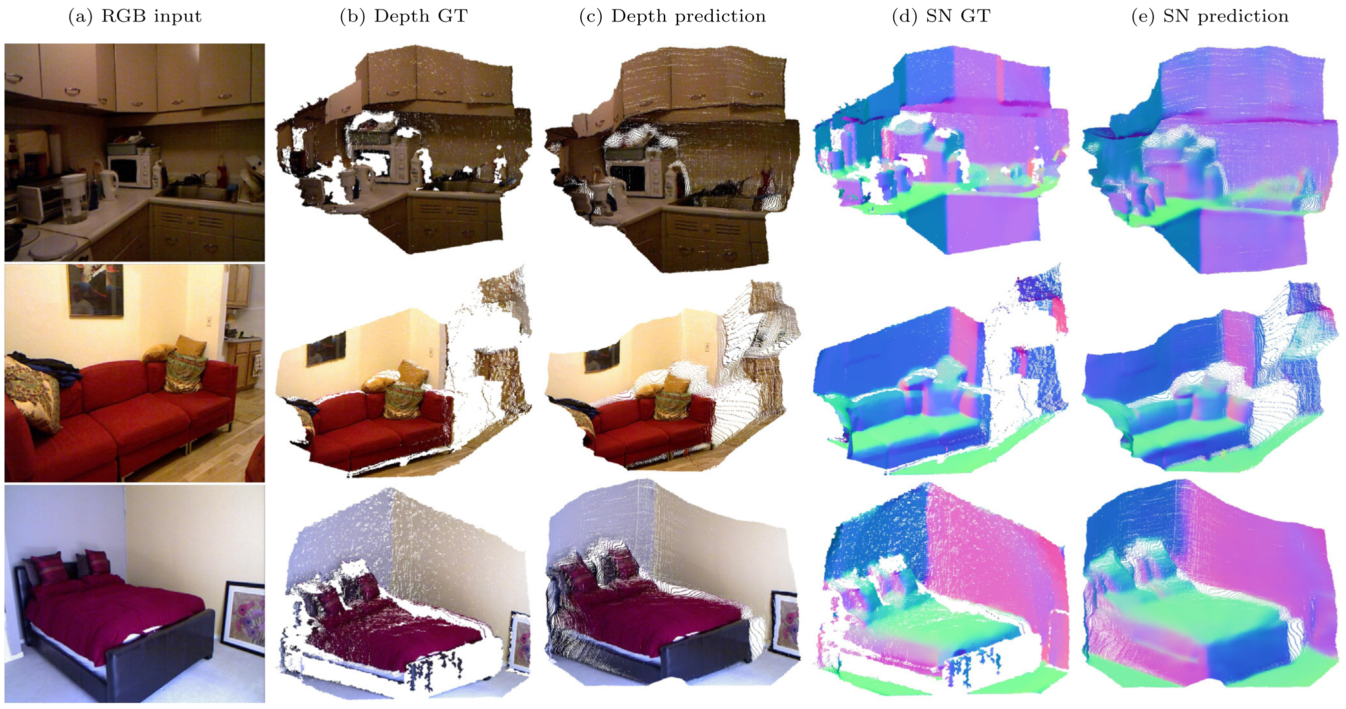


Fig. 14. Qualitative evaluation of the point clouds reconstructed from the most accurate DSN predictions. The input images and the depth ground truth compose the KITTI Depth dataset [17]. However, we created the reference surface normals data through the algorithm developed by Fouhey et al. [53].

module, improves the results once it supports the proposed CNN in the process of predicting more accurate boundaries. The edge detection operation reduces the network's prediction speed by 1 fps and does not change its number of trainable parameters.

When the DSN 3 is pre-trained on the additional surface normals ground truth, with the help of the geometric 2.5D penalty, the results are also enhanced. This occurs since our learning-based method is introduced to a greater amount of 3D geometric features which help it to better generalize. Although the presence of the Gaussian filter leads to better values for the Sqr Rel and $\delta < 1.25^3$ metrics, its absence is related to the configuration of the proposed framework that generated the most accurate overall results. This proves the efficiency and robustness of the surface normals module, which can be successfully applied to our SIDE approach at a high frame rate.

Comparing the most accurate results of the DSN, on the NYU Depth V2 dataset [18], with the metrics obtained by different works in the state-of-the-art, Table 15 shows that our method reaches the top-3 of the benchmark ranking. However, among the approaches in the top-3, ours has the lowest runtime and total size, as shown in Table 1.

Also, we evaluated the predictions from the surface normals module of the DSN and compared the acquired results with the ones generated by recent works through Table 16. Masking out the invalid pixels of the reference surface normals maps, we achieve the most accurate results presented in Table 16.

On the other hand, considering all the pixels of the surface normals ground truth, we still get the best results of the Median and the accuracy metrics. Therefore, this demonstrates that our surface normals module is more advantageous than the other techniques since, besides not consisting of an entire branch or

Table 15
Quantitative comparison between our method and others that address the single-view depth estimation task. The results are evaluated on the NYU Depth V2 dataset [18]. DSN† refers to the DSN setup that provides the most accurate overall results.

Method	Abs Rel↓	RMSE↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Saxena et al. [157]	0.349	1.214	–	0.447	0.745	0.897
Karsch et al. [182]	0.349	1.210	1.131	–	–	–
Liu et al. [183]	0.335	1.060	1.127	–	–	–
Ladicky et al. [184]	–	–	–	0.542	0.829	0.941
Eigen et al. [28]	0.214	0.877	0.285	0.611	0.887	0.971
Wang et al. [44]	0.220	0.824	–	0.605	0.890	0.970
HCRF [185]	0.232	0.821	0.094	0.621	0.886	0.968
DCNF [158]	0.213	0.759	0.087	0.650	0.906	0.976
NR forest [186]	0.187	0.744	0.078	–	–	–
MS-CNN [48]	0.158	0.641	–	0.769	0.950	0.988
Chakrabarti et al. [187]	0.149	0.620	–	0.806	0.958	0.987
Li et al. [188]	0.152	0.611	0.064	0.789	0.955	988
SOM [172]	0.136	0.604	0.067	0.814	0.959	0.990
Xu et al. [41]	0.125	0.593	–	0.806	0.952	0.986
MS-CRF [14]	0.121	0.586	0.052	0.811	0.954	0.987
PAD-Net [189]	0.120	0.582	–	0.817	0.954	0.987
DeepLabV3+ (F10) [169]	0.162	0.575	–	0.772	0.942	0.984
FCRN [29]	0.127	0.573	0.055	0.811	0.953	0.988
Lee et al. [190]	0.139	0.572	–	0.815	0.963	0.991
Qi et al. [55]	0.128	0.569	0.057	0.834	0.960	0.990
Index Network [191]	–	0.565	–	0.786	–	–
RF-LW [192]	0.149	0.565	0.105	0.790	0.955	0.990
Cao et al. [34]	0.141	0.540	0.060	0.790	0.955	0.990
RelativeDepth [193]	0.131	0.538	0.087	0.837	0.971	0.994
SC-SfMLearner-ResNet18 [194]	0.147	0.536	0.062	0.804	0.950	0.986
SENet-154 [195]	0.115	0.530	0.050	0.866	0.975	0.993
Kim et al. [175]	0.117	0.525	–	0.825	0.976	0.993
SARPN [196]	0.111	0.514	0.048	0.846	0.968	0.994
DORN [26]	0.115	0.509	0.051	0.828	0.965	0.992
AdaDepth [160]	0.114	0.506	–	0.856	0.966	0.991
TRL [197]	0.144	0.501	0.181	0.815	0.962	0.992
Zhang et al. [197]	0.144	0.501	0.181	0.815	0.962	0.992
PHN et al. [161]	0.144	0.501	–	0.835	0.962	0.992
Su et al. [168]	0.137	0.498	0.058	0.826	0.967	0.995
SDC-Depth [198]	0.128	0.497	0.174	0.845	0.966	0.990
PAP [56]	0.121	0.497	0.175	0.846	0.968	0.994
ACAN [42]	0.138	0.496	0.101	0.826	0.964	0.990
SharpNet [3]	0.139	0.495	**0.047**	**0.888**	**0.979**	**0.995**
DenseDepth [15]	0.123	0.465	0.053	0.846	0.974	0.994
DSN†	0.132	0.429	0.056	0.834	0.959	0.987
VNL [11]	**0.108**	0.416	0.048	0.875	0.976	0.994
BTS [13]	0.110	**0.392**	**0.047**	0.885	0.978	0.994

another model, it works mutually with the monocular depth estimation step.

Through the indoor scenes depicted in Figs. 11–13, we visually compare the most accurate depth and surface normals predictions of the proposed CNN with those obtained by other supervised pipelines from the SIDE literature. By Fig. 11(f), we can notice that our method is capable of better estimating the depth of finer objects and the geometric structure of the elements that compose the scenes, making the predictions closer to the ground truth.

Comparing the depth maps of Fig. 12(b) with those illustrated in Figs. 12(b) and 12(c), it is viable to conclude that our results present greater details for both the closest and the most distant artifacts from the camera. Moreover, Fig. 13 shows that the DSN completes the ground truth pixels without information and outputs surface normals vectors more consistent with the local and global layout of the scene.

The point clouds illustrated in Figs. 14(c) and 14(e) reveal that our predictions are denser than the reference point clouds and that the estimated pixels do not obey a distorted distribution of the 3D structure which portrays the environments exposed in Fig. 14(a).

4.5.3. Depth completion studies

To analyze the performance of our depth completion framework in indoor scenarios, we conducted studies with the NYU Depth V2 dataset [18] wherein we fed the DSN with sparse depth data, sampled from the ground truth, and RGB images to retrieve dense depth maps. The results of our CNN are presented in Table 17. We may observe a similar behavior of the errors and accuracy metrics than those acquired in the outdoor studies with the same depth completion network. Fig. 15 illustrates the DSNs response to the application of the increasing amount of sparse samples in its input.

Table 16
Performance of the proposed CNN and other techniques on the surface normals ground truth of the NYU Depth V2 [18]. Legend: DSN† — DSN configuration that provides the most accurate overall results; DSN‡ — same DSN configuration as the DSN†, yet with the Gaussian filter; Mask — mask out the invalid pixels related to the surface normals ground truth.

Method	Mean↓	Median↓	RMSE↓	11.25° ↑	22.50° ↑	30° ↑
Fouhey et al. [53]	36.3	19.2	–	16.4	36.6	48.2
UIOW [54]	35.2	17.9	–	40.5	54.1	58.9
Ladicky et al. [199]	33.5	23.1	–	27.7	49.0	58.7
MS-CNN [48]	23.7	15.5	–	39.2	62.0	71.1
Deep3D [200]	26.9	14.8	–	42.0	61.2	68.2
SkipNet [147]	19.8	12.0	28.2	47.9	70.0	77.8
SURGE [2]	20.6	12.2	–	47.3	68.9	76.6
Qi et al. [55]	19.0	11.8	26.9	48.4	71.5	79.5
PAP [56]	18.6	11.7	25.5	48.8	72.2	79.8
DSN† + Mask	**8.6**	**5.5**	**12.2**	72.2	**91.0**	96.8
DSN†	18.7	6.9	33.7	63.2	79.7	84.8
DSN‡ + Mask	**8.6**	**5.5**	**12.2**	**72.3**	**91.0**	**96.9**
DSN‡	18.7	6.8	33.7	63.3	79.7	84.8

Table 17
NYU Depth V2 [18] metrics obtained by our depth completion pipeline. The pattern DSN 3 + Pyramid[1] + Indoor, jointly with the $\mathcal{L}_1^+$ function, is selected for all the experiments that do not involve the surface normals ground truth. DSN 3 + Pyramid[1] + Indoor + SN† is the DSN setup with the surface normals module which yields the most accurate results.

Method	Cap	Samples	Abs Rel↓	RMSE↓	log10↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
DSN 3 + Pyramid[1] + Indoor	10	20	0.046	0.221	0.020	0.964	0.992	0.998
DSN 3 + Pyramid[1] + Indoor	10	50	0.032	0.179	0.014	0.979	0.995	0.998
DSN 3 + Pyramid[1] + Indoor	10	100	0.024	0.151	0.010	0.986	0.996	0.999
DSN 3 + Pyramid[1] + Indoor	10	200	0.018	0.132	0.008	0.990	0.997	0.999
DSN 3 + Pyramid[1] + Indoor	10	500	0.013	0.106	0.006	**0.994**	0.998	0.999
DSN 3 + Pyramid[1] + Indoor + SN†	10	200	0.017	0.128	0.007	0.991	0.998	0.999
DSN 3 + Pyramid[1] + Indoor + SN†	10	500	**0.012**	**0.102**	**0.005**	**0.994**	**0.999**	**1.000**
DSN 3 + Pyramid[1] + Indoor	5	20	0.045	0.191	0.019	0.965	0.992	0.998
DSN 3 + Pyramid[1] + Indoor	5	50	0.031	0.153	0.013	0.980	0.995	0.999
DSN 3 + Pyramid[1] + Indoor	5	100	0.023	0.129	0.010	0.987	0.997	0.999
DSN 3 + Pyramid[1] + Indoor	5	200	0.018	0.111	0.008	0.990	0.998	0.999
DSN 3 + Pyramid[1] + Indoor	5	500	0.013	0.090	**0.005**	**0.994**	0.998	**1.000**
DSN 3 + Pyramid[1] + Indoor + SN†	5	200	0.017	0.107	0.007	0.991	0.998	0.999
DSN 3 + Pyramid[1] + Indoor + SN†	5	500	**0.012**	**0.086**	**0.005**	**0.994**	**0.999**	**1.000**

Table 18
Quantitative evaluation of our best depth completion framework (DSN†) and other techniques in the state-of-the-art. The NYU Depth V2 [18] is used to compute all the metrics.

Method	Samples	Abs Rel↓	RMSE↓	$\delta < 1.25$ ↑	$\delta < 1.25^2$ ↑	$\delta < 1.25^3$ ↑
Ma et al. [7]	200	0.044	0.230	0.971	0.994	0.998
NConv [106]	200	0.027	0.173	0.982	0.996	0.999
GuideNet [111]	200	0.024	0.142	0.988	**0.998**	**1.000**
DSN†	200	**0.017**	**0.128**	**0.991**	**0.998**	0.999
Silberman et al. [18]	500	0.084	0.479	0.924	0.976	0.989
TGV [201]	500	0.123	0.635	0.819	0.930	0.968
Zhang et al. [202]	500	0.042	0.228	0.971	0.993	0.997
Ma et al. [7]	500	0.043	0.204	0.978	0.996	0.999
Ma et al. [7] + Bilateral [203]	500	0.084	0.479	0.924	0.976	0.989
Ma et al. [7] + SPN [178]	500	0.031	0.172	0.983	0.997	0.999
NConv [106]	500	0.018	0.129	0.990	0.998	**1.000**
CSPN [112]	500	0.016	0.117	0.992	**0.999**	**1.000**
Imran et al. [151]	500	0.013	0.118	0.994	**0.999**	–
CSPN++ [96]	500	–	0.116	–	–	–
DeepLiDAR [103]	500	0.022	0.115	0.993	**0.999**	**1.000**
Xu et al. [118]	500	0.018	0.112	0.995	**0.999**	**1.000**
GuideNet [111]	500	0.015	0.101	0.995	**0.999**	**1.000**
DSN†	500	**0.012**	0.102	0.994	**0.999**	**1.000**
NLSPN [113]	500	**0.012**	**0.092**	**0.996**	**0.999**	**1.000**

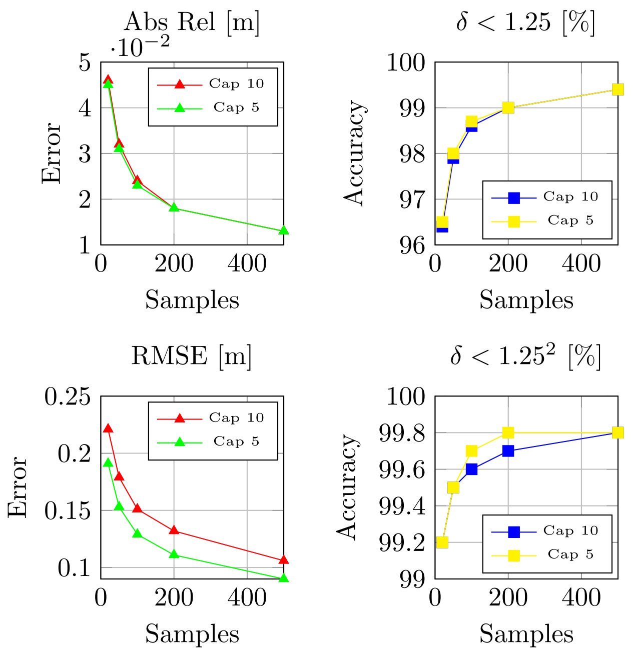


Fig. 15. Evolution of the depth completion metrics obtained by the DSN. We employed the reference depth maps of the NYU Depth V2 dataset [18] to plot the results.

We compare the results generated by the best DSN setup (DSN†) with the ones acquired from recent depth completion methods through Table 18. Considering both Abs Rel and RMSE metrics, we achieve the top-2 in the benchmark ranking of Table 18. However, the Non-local CSPN technique [113] has a higher number of trainable parameters (25 M) and it applies a more complex feature extraction CNN (ResNet-34).

Fig. 16 shows the predictions produced by the DSN and by state-of-the-art techniques that address the depth completion task. Through this figure, it is plausible to state that our results are the ones that come closest to the reference. We also present the surface normals maps estimated by the DSN which are visually consistent with the output depth maps. Furthermore, the metrics presented in Table 19 indicate that, as with depth predictions, the quality of the surface normals maps increases as more sparse depth data is added to the network's input.

5. Conclusions

Through the conducted surveys, we can imply that the monocular depth estimation and depth completion areas have significantly advanced in recent years due to the emergence of Deep Learning methods. These methods may overcome the performance of classical Computer Vision techniques and also generate better predictions for other fundamental tasks such as semantic segmentation, surface normals estimation and VO.

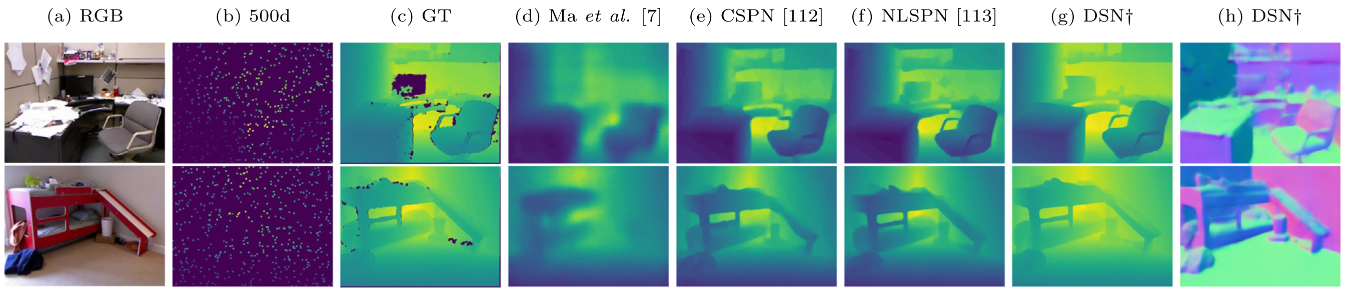


Fig. 16. Comparison between the dense depth maps retrieved by our best depth completion CNN (DSN†) and by other methods in the literature. We also present the surface normals maps estimated by the DSN†. The NYU Depth V2 [18] is employed by all approaches. Legend: RGB — input image; 500d - sparse input data; GT — sparse ground truth.

Table 19
Analysis of the surface normals predictions from our best depth completion CNN (DSN†) when it receives different amounts of sparse samples as input. Mask stands for the operation of masking out the invalid pixels corresponding to the surface normals ground truth.

Method	Samples	Mean↓	Median↓	RMSE↓	11.25° ↑	22.50° ↑	30° ↑
DSN† + Mask	200	7.6	3.5	12.2	77.1	91.0	95.6
DSN†	200	17.8	4.6	33.7	67.6	79.7	83.7
DSN† + Mask	500	**6.1**	**3.0**	**9.6**	**82.9**	**94.9**	**98.1**
DSN†	500	16.5	3.9	33.0	72.6	83.2	86

Particularly, considering all the experimental analysis, we designed a lightweight CNN architecture, whose number of trainable parameters can vary from 2 M up to 12 M. Our framework can also be used in real-world applications since it achieves prediction speeds from 32 fps up to 88 fps.

For the tests involving outdoor scenarios, the DenseSIDENet configuration that includes the proposed surface normals module, the 2.5D geometric loss function, the pyramid block and the pre-training with the cityscapes [135] dataset provided results in the state-of-the-art for the SIDE ($Abs\ Rel = 0.075$), depth completion ($Abs\ Rel = 0.019$) and VO ($seq00\ RMSE = 16.9438$) tasks, regarding the KITTI Benchmark datasets [17,146,156].

On the other hand, for the experiments involving indoor environments, the same DSN setup aforementioned, with the exception of the pre-training step that is conducted with the aid of indoor datasets [136–138], produced promising results for the single-view depth prediction ($RMSE = 0.429$), depth completion ($RMSE = 0.102$) and surface normals estimation ($RMSE = 12.2$) tasks on the NYU Benchmark datasets [18].

Therefore, we were able to determine optimal training conditions, using different deep learning techniques, as well as optimized network structures that enabled the generation of high-quality predictions in reduced processing time. As future work, we intend to employ the developed CNNs for other tasks such as robotic grasping and visual servoing control.

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.

Acknowledgments

This research was financed in part by the Brazilian National Council for Scientific and Technological Development — CNPq, Brazil (grants 465755/2014-3 and 130602/2018-3), by the Coordination of Improvement of Higher Education Personnel – Brazil – CAPES (Finance Code 001, grants 88887.136349/2017-00 and 88887.339070/2019-00), and the São Paulo Research Foundation — FAPESP, Brazil (grant 2014/50851-0).

References

[1] M. Mancini, G. Costante, P. Valigi, T.A. Ciarfuglia, J. Delmerico, D. Scaramuzza, Toward domain independence for learning-based monocular depth estimation, IEEE Robot. Autom. Lett. 2 (3) (2017) 1778–1785.
[2] P. Wang, X. Shen, B. Russell, S. Cohen, B. Price, A.L. Yuille, Surge: Surface regularized geometry estimation from a single image, in: Advances in Neural Information Processing Systems, 2016, pp. 172–180.
[3] M. Ramamonjisoa, V. Lepetit, Sharpnet: Fast and accurate recovery of occluding contours in monocular depth estimation, in: Proceedings of the IEEE International Conference on Computer Vision Workshops, 2019.
[4] J. Chang, G. Wetzstein, Deep optics for monocular depth estimation and 3d object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 10193–10202.
[5] S.Y. Loo, A.J. Amiri, S. Mashohor, S.H. Tang, H. Zhang, CNN-SVO: Improving the mapping in semi-direct visual odometry using single-image depth prediction, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 5218–5223.
[6] K. Tateno, F. Tombari, I. Laina, N. Navab, Cnn-slam: Real-time dense monocular slam with learned depth prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6243–6252.
[7] F. Ma, S. Karaman, Sparse-to-dense: Depth prediction from sparse depth samples and a single image, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 1–8.
[8] W. Van Gansbeke, D. Neven, B. De Brabandere, L. Van Gool, Sparse and noisy lidar completion with rgb guidance and uncertainty, in: 2019 16th International Conference on Machine Vision Applications, MVA, IEEE, 2019, pp. 1–6.
[9] F. Ma, G.V. Cavalheiro, S. Karaman, Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 3288–3295.
[10] V. Guizilini, R. Hou, J. Li, R. Ambrus, A. Gaidon, Semantically-guided representation learning for self-supervised monocular depth, 2020, arXiv preprint arXiv:2002.12319.
[11] W. Yin, Y. Liu, C. Shen, Y. Yan, Enforcing geometric constraints of virtual normal for depth prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 5684–5693.
[12] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, A.L. Yuille, Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2017) 834–848.
[13] J.H. Lee, M.-K. Han, D.W. Ko, I.H. Suh, From big to small: Multi-scale local planar guidance for monocular depth estimation, 2019, arXiv preprint arXiv:1907.10326.
[14] D. Xu, E. Ricci, W. Ouyang, X. Wang, N. Sebe, Multi-scale continuous crfs as sequential deep networks for monocular depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5354–5362.
[15] I. Alhashim, P. Wonka, High quality monocular depth estimation via transfer learning, 2018, arXiv preprint arXiv:1812.11941.
[16] N. Silberman, R. Fergus, Indoor scene segmentation using a structured light sensor, in: 2011 IEEE International Conference on Computer Vision Workshops, ICCV Workshops, IEEE, 2011, pp. 601–608.
[17] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, A. Geiger, Sparsity invariant cnns, in: 2017 International Conference on 3D Vision, 3DV, IEEE, 2017, pp. 11–20.
[18] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: European Conference on Computer Vision, Springer, 2012, pp. 746–760.

[19] D. Hoiem, A.A. Efros, M. Hebert, Closing the loop in scene interpretation, in: 2008 IEEE Conference on Computer Vision and Pattern Recognition, IEEE, 2008, pp. 1–8.

[20] D. Hoiem, A.A. Efros, M. Hebert, Recovering surface layout from an image, Int. J. Comput. Vis. 75 (1) (2007) 151–172.

[21] V. Hedau, D. Hoiem, D. Forsyth, Recovering the spatial layout of cluttered rooms, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 1849–1856.

[22] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[23] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.

[24] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1492–1500.

[25] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.

[26] H. Fu, M. Gong, C. Wang, K. Batmanghelich, D. Tao, Deep ordinal regression network for monocular depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2002–2011.

[27] C. Godard, O. Mac 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. 270–279.

[28] D. Eigen, C. Puhrsch, R. Fergus, Depth map prediction from a single image using a multi-scale deep network, in: Advances in Neural Information Processing Systems, 2014, pp. 2366–2374.

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

[30] 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, IEEE, Las Vegas, USA, 2016, pp. 770–778.

[31] L. Zwald, S. Lambert-Lacroix, The berhu penalty and the grouped effect, 2012, arXiv preprint arXiv:1207.6868.

[32] R. Girshick, Fast r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.

[33] F. Liu, C. Shen, G. Lin, Deep convolutional neural fields for depth estimation from a single image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 5162–5170.

[34] Y. Cao, Z. Wu, C. Shen, Estimating depth from monocular images as classification using deep fully convolutional residual networks, IEEE Trans. Circuits Syst. Video Technol. 28 (11) (2017) 3174–3182.

[35] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, Y. Liu, Parse geometry from a line: Monocular depth estimation with partial laser observation, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 5059–5066.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[37] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[38] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: Advances in Neural Information Processing Systems, 2015, pp. 802–810.

[39] A. CS Kumar, S.M. Bhandarkar, M. Prasad, Depthnet: A recurrent neural network architecture for monocular depth prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 283–291.

[40] R. Wang, S.M. Pizer, J.-M. Frahm, Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5555–5564.

[41] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, E. Ricci, Structured attention guided convolutional neural fields for monocular depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3917–3925.

[42] Y. Chen, H. Zhao, Z. Hu, Attention-based context aggregation network for monocular depth estimation, 2019, arXiv preprint arXiv:1901.10137.

[43] J. Jiao, Y. Cao, Y. Song, R. Lau, Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 53–69.

[44] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, A.L. Yuille, Towards unified depth and semantic prediction from a single image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2800–2809.

[45] D. Hoiem, A.A. Efros, M. Hebert, Recovering occlusion boundaries from an image, Int. J. Comput. Vis. 91 (3) (2011) 328–346.

[46] B. Liu, S. Gould, D. Koller, Single image depth estimation from predicted semantic labels, in: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE, 2010, pp. 1253–1260.

[47] S. Gupta, R. Girshick, P. Arbeláez, J. Malik, Learning rich features from RGB-D images for object detection and segmentation, in: European Conference on Computer Vision, Springer, 2014, pp. 345–360.

[48] D. Eigen, R. Fergus, Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2650–2658.

[49] A. Mousavian, H. Pirsiavash, J. Košecká, Joint semantic segmentation and depth estimation with deep convolutional networks, in: 2016 Fourth International Conference on 3D Vision, 3DV, IEEE, 2016, pp. 611–619.

[50] R. Zhang, P.-S. Tsai, J.E. Cryer, M. Shah, Shape-from-shading: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 21 (8) (1999) 690–706.

[51] T. Binford, Visual Perception by Computer. IEEE Cont. on Systems and Control, Miami, 1971.

[52] A. Saxena, M. Sun, A.Y. Ng, Learning 3-d scene structure from a single still image, in: 2007 IEEE 11th International Conference on Computer Vision, IEEE, 2007, pp. 1–8.

[53] D.F. Fouhey, A. Gupta, M. Hebert, Data-driven 3D primitives for single image understanding, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 3392–3399.

[54] D.F. Fouhey, A. Gupta, M. Hebert, Unfolding an indoor origami world, in: European Conference on Computer Vision, Springer, 2014, pp. 687–702.

[55] X. Qi, R. Liao, Z. Liu, R. Urtasun, J. Jia, Geonet: Geometric neural network for joint depth and surface normal estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 283–291.

[56] Z. Zhang, Z. Cui, C. Xu, Y. Yan, N. Sebe, J. Yang, Pattern-Affinitive Propagation across Depth, Surface Normal and Semantic Segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 4106–4115.

[57] R. Garg, V.K. BG, G. Carneiro, I. Reid, Unsupervised cnn for single view depth estimation: Geometry to the rescue, in: European Conference on Computer Vision, Springer, 2016, pp. 740–756.

[58] S. Pillai, R. Ambruş, A. Gaidon, Superdepth: Self-supervised, super-resolved monocular depth estimation, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 9250–9256.

[59] T. Zhou, M. Brown, N. Snavely, D.G. Lowe, Unsupervised learning of depth and ego-motion from video, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1851–1858.

[60] C. Godard, O. Mac Aodha, M. Firman, G.J. Brostow, Digging into self-supervised monocular depth estimation, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 3828–3838.

[61] Y. Kuznietsov, J. Stuckler, B. Leibe, Semi-supervised deep learning for monocular depth map prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 6647–6655.

[62] A.J. Amiri, S.Y. Loo, H. Zhang, Semi-supervised monocular depth estimation with left-right consistency using deep neural network, in: 2019 IEEE International Conference on Robotics and Biomimetics, ROBIO, IEEE, 2019, pp. 602–607.

[63] N. Yang, R. Wang, J. Stuckler, D. Cremers, Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 817–833.

[64] L. Andraghetti, P. Myriokefalitakis, P.L. Dovesi, B. Luque, M. Poggi, A. Pieropan, S. Mattoccia, Enhancing self-supervised monocular depth estimation with traditional visual odometry, in: 2019 International Conference on 3D Vision, 3DV, IEEE, 2019, pp. 424–433.

[65] P.Z. Ramirez, M. Poggi, F. Tosi, S. Mattoccia, L. Di Stefano, Geometry meets semantics for semi-supervised monocular depth estimation, in: Asian Conference on Computer Vision, Springer, 2018, pp. 298–313.

[66] Y. Luo, J. Ren, M. Lin, J. Pang, W. Sun, H. Li, L. Lin, Single view stereo matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 155–163.

[67] F. Tosi, F. Aleotti, M. Poggi, S. Mattoccia, Learning monocular depth estimation infusing traditional stereo knowledge, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9799–9809.

[68] R. Li, S. Wang, Z. Long, D. Gu, Undeepvo: Monocular visual odometry through unsupervised deep learning, in: 2018 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2018, pp. 7286–7291.

[69] V.M. Babu, K. Das, A. Majumdar, S. Kumar, Undemon: Unsupervised deep network for depth and ego-motion estimation, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 1082–1088.

[70] P. Charbonnier, L. Blanc-Feraud, G. Aubert, M. Barlaud, Two deterministic half-quadratic regularization algorithms for computed imaging, in: Proceedings of 1st International Conference on Image Processing, Vol. 2, IEEE, 1994, pp. 168–172.

[71] M. Poggi, F. Tosi, S. Mattoccia, Learning monocular depth estimation with unsupervised trinocular assumptions, in: 2018 International Conference on 3D Vision, 3DV, IEEE, 2018, pp. 324–333.

[72] X. Guo, H. Li, S. Yi, J. Ren, X. Wang, Learning monocular depth by distilling cross-domain stereo networks, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 484–500.

[73] A. CS Kumar, S.M. Bhandarkar, M. Prasad, Monocular depth prediction using generative adversarial networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 300–308.

[74] F. Aleotti, F. Tosi, M. Poggi, S. Mattoccia, Generative adversarial networks for unsupervised monocular depth prediction, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018.

[75] H. Gupta, K. Mitra, Unsupervised single image underwater depth estimation, in: 2019 IEEE International Conference on Image Processing, ICIP, IEEE, 2019, pp. 624–628.

[76] L. Chen, W. Tang, T.R. Wan, N.W. John, Self-supervised monocular image depth learning and confidence estimation, Neurocomputing 381 (2020) 272–281.

[77] M. Poggi, F. Aleotti, F. Tosi, S. Mattoccia, On the uncertainty of self-supervised monocular depth estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 3227–3237.

[78] J.L. GonzalezBello, M. Kim, Forget about the LiDAR: Self-supervised depth estimators with MED probability volumes, Adv. Neural Inf. Process. Syst. 33 (2020).

[79] A. Pilzer, D. Xu, M. Puscas, E. Ricci, N. Sebe, Unsupervised adversarial depth estimation using cycled generative networks, in: 2018 International Conference on 3D Vision, 3DV, IEEE, 2018, pp. 587–595.

[80] M. Poggi, F. Aleotti, F. Tosi, S. Mattoccia, Towards real-time unsupervised monocular depth estimation on cpu, in: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2018, pp. 5848–5854.

[81] V. Peluso, A. Cipolletta, A. Calimera, M. Poggi, F. Tosi, S. Mattoccia, Enabling energy-efficient unsupervised monocular depth estimation on armv7-based platforms, in: 2019 Design, Automation & Test in Europe Conference & Exhibition, DATE, IEEE, 2019, pp. 1703–1708.

[82] S. Elkerdawy, H. Zhang, N. Ray, Lightweight monocular depth estimation model by joint end-to-end filter pruning, in: 2019 IEEE International Conference on Image Processing, ICIP, IEEE, 2019, pp. 4290–4294.

[83] Z. Yang, P. Wang, W. Xu, L. Zhao, R. Nevatia, Unsupervised learning of geometry from videos with edge-aware depth-normal consistency, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[84] Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, Lego: Learning edge with geometry all at once by watching videos, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 225–234.

[85] Z. Yin, J. Shi, Geonet: Unsupervised learning of dense depth, optical flow and camera pose, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 1983–1992.

[86] R. Mahjourian, M. Wicke, A. Angelova, Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5667–5675.

[87] C. Wang, J. Miguel Buenaposada, R. Zhu, S. Lucey, Learning depth from monocular videos using direct methods, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2022–2030.

[88] V. Casser, S. Pirk, R. Mahjourian, A. Angelova, Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 8001–8008.

[89] Z. Teed, J. Deng, Deepv2d: Video to depth with differentiable structure from motion, 2018, arXiv preprint arXiv:1812.04605.

[90] V. Casser, S. Pirk, R. Mahjourian, A. Angelova, Unsupervised monocular depth and ego-motion learning with structure and semantics, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019.

[91] A. Ranjan, V. Jampani, L. Balles, K. Kim, D. Sun, J. Wulff, M.J. Black, Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 12240–12249.

[92] Y. Chen, C. Schmid, C. Sminchisescu, Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 7063–7072.

[93] V. Guizilini, R. Ambrus, S. Pillai, A. Gaidon, Packnet-sfm: 3d packing for self-supervised monocular depth estimation, 2019, arXiv preprint arXiv:1905.02693.

[94] A. Sharma, J. Ventura, Unsupervised learning of depth and ego-motion from cylindrical panoramic video, 2019, arXiv preprint arXiv:1901.00979.

[95] J. Liu, Q. Li, R. Cao, W. Tang, G. Qiu, MiniNet: An extremely lightweight convolutional neural network for real-time unsupervised monocular depth estimation, ISPRS J. Photogramm. Remote Sens. 166 (2020) 255–267.

[96] X. Cheng, P. Wang, C. Guan, R. Yang, CSPN++: Learning context and resource aware convolutional spatial propagation networks for depth completion, 2019, arXiv preprint arXiv:1911.05377.

[97] D. Herrera, J. Kannala, J. Heikkilä, et al., Depth map inpainting under a second-order smoothness prior, in: Scandinavian Conference on Image Analysis, Springer, 2013, pp. 555–566.

[98] H.-T. Zhang, J. Yu, Z.-F. Wang, Probability contour guided depth map inpainting and superresolution using non-local total generalized variation, Multimedia Tools Appl. 77 (7) (2018) 9003–9020.

[99] X. Zhang, R. Wu, Fast depth image denoising and enhancement using a deep convolutional network, in: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2016, pp. 2499–2503.

[100] J. Shen, S.-C.S. Cheung, Layer depth denoising and completion for structured-light rgb-d cameras, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1187–1194.

[101] L.-F. Yu, S.-K. Yeung, Y.-W. Tai, S. Lin, Shading-based shape refinement of RGB-D images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1415–1422.

[102] J. Lu, D. Forsyth, Sparse depth super resolution, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 2245–2253.

[103] J. Qiu, Z. Cui, Y. Zhang, X. Zhang, S. Liu, B. Zeng, M. Pollefeys, Deeplidar: Deep surface normal guided depth prediction for outdoor scene from sparse lidar data and single color image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3313–3322.

[104] J. Ku, A. Harakeh, S.L. Waslander, In defense of classical image processing: Fast depth completion on the cpu, in: 2018 15th Conference on Computer and Robot Vision, CRV, IEEE, 2018, pp. 16–22.

[105] N. Schneider, L. Schneider, P. Pinggera, U. Franke, M. Pollefeys, C. Stiller, Semantically guided depth upsampling, in: German Conference on Pattern Recognition, Springer, 2016, pp. 37–48.

[106] A. Eldesokey, M. Felsberg, F.S. Khan, Confidence propagation through cnns for guided sparse depth regression, IEEE Trans. Pattern Anal. Mach. Intell. (2019).

[107] N. Chodosh, C. Wang, S. Lucey, Deep convolutional compressed sensing for lidar depth completion, in: Asian Conference on Computer Vision, Springer, 2018, pp. 499–513.

[108] Z. Huang, J. Fan, S. Cheng, S. Yi, X. Wang, H. Li, Hms-net: Hierarchical multi-scale sparsity-invariant network for sparse depth completion, IEEE Trans. Image Process. (2019).

[109] L. Yan, K. Liu, E. Belyaev, Revisiting sparsity invariant convolution: A network for image guided depth completion, IEEE Access (2020).

[110] K. Lu, N. Barnes, S. Anwar, L. Zheng, From Depth What Can You See? Depth Completion via Auxiliary Image Reconstruction, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11306–11315.

[111] J. Tang, F.-P. Tian, W. Feng, J. Li, P. Tan, Learning guided convolutional network for depth completion, 2019, arXiv preprint arXiv:1908.01238.

[112] X. Cheng, P. Wang, R. Yang, Depth estimation via affinity learned with convolutional spatial propagation network, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 103–119.

[113] J. Park, K. Joo, Z. Hu, C.-K. Liu, I.S. Kweon, Non-local spatial propagation network for depth completion, 2020, arXiv preprint arXiv:2007.10042.

[114] Z. Xu, Y. Wang, J. Yao, Deformable spatial propagation network for depth completion, 2020, arXiv preprint arXiv:2007.04251.

[115] Y. Chen, B. Yang, M. Liang, R. Urtasun, Learning Joint 2D-3D Representations for Depth Completion, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 10023–10032.

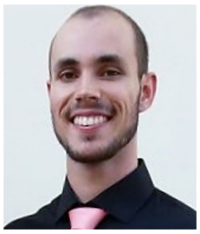
[116] A. Li, Z. Yuan, Y. Ling, W. Chi, C. Zhang, et al., A Multi-Scale Guided Cascade Hourglass Network for Depth Completion, in: The IEEE Winter Conference on Applications of Computer Vision, 2020, pp. 32–40.

[117] M. Dimitrievski, P. Veelaert, W. Philips, Learning morphological operators for depth completion, in: International Conference on Advanced Concepts for Intelligent Vision Systems, Springer, 2018, pp. 450–461.

[118] Y. Xu, X. Zhu, J. Shi, G. Zhang, H. Bao, H. Li, Depth completion from sparse LiDAR data with depth-normal constraints, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 2811–2820.
[119] Y. Wu, V. Boominathan, H. Chen, A. Sankaranarayanan, A. Veeraraghavan, Phasecam3d—learning phase masks for passive single view depth estimation, in: 2019 IEEE International Conference on Computational Photography, ICCP, IEEE, 2019, pp. 1–12.
[120] S. Lee, J. Lee, D. Kim, J. Kim, Deep architecture with cross guidance between single image and sparse lidar data for depth completion, IEEE Access 8 (2020) 79801–79810.
[121] M. Klodt, A. Vedaldi, Supervising the new with the old: learning sfm from sfm, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 698–713.
[122] Y. Zou, Z. Luo, J.-B. Huang, Df-net: Unsupervised joint learning of depth and flow using cross-task consistency, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 36–53.
[123] R. Li, K. Xian, C. Shen, Z. Cao, H. Lu, L. Hang, Deep attention-based classification network for robust depth prediction, in: Asian Conference on Computer Vision, Springer, 2018, pp. 663–678.
[124] S. Kong, C. Fowlkes, Pixel-wise attentional gating for scene parsing, in: 2019 IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2019, pp. 1024–1033.
[125] B. Li, Y. Dai, M. He, Monocular depth estimation with hierarchical fusion of dilated cnns and soft-weighted-sum inference, Pattern Recognit. 83 (2018) 328–339.
[126] M. Goldman, T. Hassner, S. Avidan, Learn stereo, infer mono: Siamese networks for self-supervised, monocular, depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2019.
[127] Z. Zhang, C. Xu, J. Yang, Y. Tai, L. Chen, Deep hierarchical guidance and regularization learning for end-to-end depth estimation, Pattern Recognit. 83 (2018) 430–442.
[128] M. Ochs, A. Kretz, R. Mester, Sdnet: Semantically guided depth estimation network, in: German Conference on Pattern Recognition, Springer, 2019, pp. 288–302.
[129] J. Zhou, Y. Wang, K. Qin, W. Zeng, Unsupervised high-resolution depth learning from videos with dual networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6872–6881.
[130] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[131] E.G. Ribeiro, V. Grassi, Fast convolutional neural network for real-time robotic grasp detection, in: 2019 19th International Conference on Advanced Robotics, ICAR, IEEE, 2019, pp. 49–54.
[132] M. Yang, K. Yu, C. Zhang, Z. Li, K. Yang, Denseaspp for semantic segmentation in street scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3684–3692.
[133] A.F. Agarap, Deep learning using rectified linear units (relu), 2018, arXiv preprint arXiv:1803.08375.
[134] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), 2015, arXiv preprint arXiv:1511.07289.
[135] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, B. Schiele, The cityscapes dataset for semantic urban scene understanding, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 3213–3223.
[136] S. Song, S.P. Lichtenberg, J. Xiao, Sun rgb-d: A rgb-d scene understanding benchmark suite, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 567–576.
[137] A. Janoch, S. Karayev, Y. Jia, J.T. Barron, M. Fritz, K. Saenko, T. Darrell, A category-level 3d object dataset: Putting the kinect to work, in: Consumer Depth Cameras for Computer Vision, Springer, 2013, pp. 141–165.
[138] J. Xiao, A. Owens, A. Torralba, Sun3d: A database of big spaces reconstructed using sfm and object labels, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 1625–1632.
[139] M. Carvalho, B. Le Saux, P. Trouvé-Peloux, A. Almansa, F. Champagnat, On regression losses for deep depth estimation, in: 2018 25th IEEE International Conference on Image Processing, ICIP, IEEE, 2018, pp. 2915–2919.
[140] D. Sun, S. Roth, M.J. Black, A quantitative analysis of current practices in optical flow estimation and the principles behind them, Int. J. Comput. Vis. 106 (2) (2014) 115–137.
[141] P. Chen, G. Chen, S. Zhang, Log hyperbolic cosine loss improves variational auto-encoder, 2018.
[142] M. Nixon, A. Aguado, Feature Extraction and Image Processing for Computer Vision, Academic press, 2019.
[143] H.H. Afshari, S.A. Gadsden, S. Habibi, Gaussian filters for parameter and state estimation: A general review of theory and recent trends, Signal Process. 135 (2017) 218–238.
[144] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv:1412.6980.
[145] A. Geiger, P. Lenz, C. Stiller, R. Urtasun, Vision meets robotics: The kitti dataset, Int. J. Robot. Res. 32 (11) (2013) 1231–1237.
[146] A. Geiger, P. Lenz, R. Urtasun, Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, in: Conference on Computer Vision and Pattern Recognition, CVPR, 2012.
[147] A. Bansal, B. Russell, A. Gupta, Marr revisited: 2d-3d alignment via surface normal prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5965–5974.
[148] N. dos Santos Rosa, V. Guizilini, V. Grassi, Sparse-to-continuous: Enhancing monocular depth estimation using occupancy maps, in: 2019 19th International Conference on Advanced Robotics, ICAR, IEEE, 2019, pp. 793–800.
[149] S.S. Shivakumar, T. Nguyen, I.D. Miller, S.W. Chen, V. Kumar, C.J. Taylor, Dfusenet: Deep fusion of rgb and sparse depth information for image guided dense depth completion, in: 2019 IEEE Intelligent Transportation Systems Conference, ITSC, IEEE, 2019, pp. 13–20.
[150] A. Wong, X. Fei, S. Tsuei, S. Soatto, Unsupervised depth completion from visual inertial odometry, IEEE Robot. Autom. Lett. 5 (2) (2020) 1899–1906.
[151] S. Imran, Y. Long, X. Liu, D. Morris, Depth coefficients for depth completion, in: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, 2019, pp. 12438–12447.
[152] H. Hekmatian, S. Al-Stouhi, J. Jin, Conf-net: Predicting depth completion error-map forhigh-confidence dense 3D point-cloud, 2019, arXiv preprint arXiv:1907.10148.
[153] Y. Zhang, T. Nguyen, I.D. Miller, S.S. Shivakumar, S. Chen, C.J. Taylor, V. Kumar, Dfinenet: Ego-motion estimation and depth refinement from sparse, noisy depth input with rgb guidance, 2019, arXiv preprint arXiv:1903.06397.
[154] M. Jaritz, R. De Charette, E. Wirbel, X. Perrotton, F. Nashashibi, Sparse and dense data with cnns: Depth completion and semantic segmentation, in: 2018 International Conference on 3D Vision, 3DV, IEEE, 2018, pp. 52–60.
[155] Y. Yang, A. Wong, S. Soatto, Dense depth posterior (ddp) from single image and sparse range, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3353–3362.
[156] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger, C. Rother, Augmented reality meets computer vision: Efficient data generation for urban driving scenes, Int. J. Comput. Vis. (2018).
[157] A. Saxena, M. Sun, A.Y. Ng, Make3D: Depth Perception from a Single Still Image, in: AAAI, Vol. 3, 2008, pp. 1571–1576.
[158] F. Liu, C. Shen, G. Lin, I. Reid, Learning depth from single monocular images using deep convolutional neural fields, IEEE Trans. Pattern Anal. Mach. Intell. 38 (10) (2015) 2024–2039.
[159] Z. Yang, P. Wang, W. Xu, L. Zhao, R. Nevatia, Unsupervised learning of geometry with edge-aware depth-normal consistency, 2017, arXiv preprint arXiv:1711.03665.
[160] J. Nath Kundu, P. Krishna Uppala, A. Pahuja, R. Venkatesh Babu, Adadepth: Unsupervised content congruent adaptation for depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2656–2665.
[161] Z. Zhang, C. Xu, J. Yang, J. Gao, Z. Cui, Progressive hard-mining network for monocular depth estimation, IEEE Trans. Image Process. 27 (8) (2018) 3691–3702.
[162] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, I. Reid, Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 340–349.
[163] A. Wong, S. Soatto, Bilateral cyclic constraint and adaptive regularization for unsupervised monocular depth prediction, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 5644–5653.
[164] I. Mehta, P. Sakurikar, P. Narayanan, Structured adversarial training for unsupervised monocular depth estimation, in: 2018 International Conference on 3D Vision, 3DV, IEEE, 2018, pp. 314–323.
[165] C. Luo, Z. Yang, P. Wang, Y. Wang, W. Xu, R. Nevatia, A. Yuille, Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding, 2018, arXiv preprint arXiv:1810.06125.
[166] A. Gordon, H. Li, R. Jonschkowski, A. Angelova, Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8977–8986.
[167] P.-Y. Chen, A.H. Liu, Y.-C. Liu, Y.-C.F. Wang, Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2624–2632.
[168] W. Su, H. Zhang, Q. Zhou, W. Yang, Z. Wang, Monocular depth estimation using information exchange network, IEEE Trans. Intell. Transp. Syst. (2020).

[169] S. Gur, L. Wolf, Single image depth estimation trained via depth from defocus cues, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7683–7692.
[170] A. Gurram, O. Urfalioglu, I. Halfaoui, F. Bouzaraa, A.M. López, Monocular depth estimation by learning from heterogeneous datasets, in: 2018 IEEE Intelligent Vehicles Symposium, IV, IEEE, 2018, pp. 2176–2181.
[171] A. Pilzer, S. Lathuiliere, N. Sebe, E. Ricci, Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9768–9777.
[172] J. Zhu, Y. Shi, M. Ren, Y. Fang, K.-C. Lien, J. Gu, Structure-attentioned memory network for monocular depth estimation, 2019, arXiv preprint arXiv:1909.04594.
[173] J. Watson, M. Firman, G.J. Brostow, D. Turmukhambetov, Self-supervised monocular depth hints, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 2162–2171.
[174] C. Tang, P. Tan, Ba-net: Dense bundle adjustment network, 2018, arXiv preprint arXiv:1806.04807.
[175] Y. Kim, H. Jung, D. Min, K. Sohn, Deep monocular depth estimation via integration of global and local predictions, IEEE Trans. Image Process. 27 (8) (2018) 4131–4144.
[176] Y. Gan, X. Xu, W. Sun, L. Lin, Monocular depth estimation with affinity, vertical pooling, and label enhancement, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 224–239.
[177] C. Cadena, A.R. Dick, I.D. Reid, Multi-modal auto-encoders as joint estimators for robotics scene understanding, in: Robotics: Science and Systems, Vol. 5, 2016, p. 1.
[178] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, J. Kautz, Learning affinity via spatial propagation networks, in: Advances in Neural Information Processing Systems, 2017, pp. 1520–1530.
[179] C. Forster, M. Pizzoli, D. Scaramuzza, SVO: Fast semi-direct monocular visual odometry, in: 2014 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2014, pp. 15–22.
[180] J. Engel, V. Koltun, D. Cremers, Direct sparse odometry, IEEE Trans. Pattern Anal. Mach. Intell. 40 (3) (2017) 611–625.
[181] R. Mur-Artal, J.D. Tardós, Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras, IEEE Trans. Robot. 33 (5) (2017) 1255–1262.
[182] K. Karsch, C. Liu, S.B. Kang, Depth transfer: Depth extraction from video using non-parametric sampling, IEEE Trans. Pattern Anal. Mach. Intell. 36 (11) (2014) 2144–2158.
[183] M. Liu, M. Salzmann, X. He, Discrete-continuous depth estimation from a single image, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 716–723.
[184] L. Ladicky, J. Shi, M. Pollefeys, Pulling things out of perspective, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 89–96.
[185] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, M. He, Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1119–1127.
[186] A. Roy, S. Todorovic, Monocular depth estimation using neural regression forest, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5506–5514.
[187] A. Chakrabarti, J. Shao, G. Shakhnarovich, Depth from a single image by harmonizing overcomplete local network predictions, in: Advances in Neural Information Processing Systems, 2016, pp. 2658–2666.
[188] J. Li, R. Klein, A. Yao, A two-streamed network for estimating fine-scaled depth maps from single rgb images, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 3372–3380.
[189] D. Xu, W. Ouyang, X. Wang, N. Sebe, Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 675–684.
[190] J.-H. Lee, M. Heo, K.-R. Kim, C.-S. Kim, Single-image depth estimation based on fourier domain analysis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 330–339.
[191] H. Lu, Y. Dai, C. Shen, S. Xu, Index network, 2019, arXiv preprint arXiv:1908.09895.
[192] V. Nekrasov, T. Dharmasiri, A. Spek, T. Drummond, C. Shen, I. Reid, Real-time joint semantic segmentation and depth estimation using asymmetric annotations, in: 2019 International Conference on Robotics and Automation, ICRA, IEEE, 2019, pp. 7101–7107.
[193] J.-H. Lee, C.-S. Kim, Monocular depth estimation using relative depth maps, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 9729–9738.
[194] J.-W. Bian, H. Zhan, N. Wang, T.-J. Chin, C. Shen, I. Reid, Unsupervised depth learning in challenging indoor video: Weak rectification to rescue, 2020, arXiv preprint arXiv:2006.02708.
[195] J. Hu, M. Ozay, Y. Zhang, T. Okatani, Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries, in: 2019 IEEE Winter Conference on Applications of Computer Vision, WACV, IEEE, 2019, pp. 1043–1051.
[196] X. Chen, X. Chen, Z.-J. Zha, Structure-aware residual pyramid network for monocular depth estimation, 2019, arXiv preprint arXiv:1907.06023.
[197] Z. Zhang, Z. Cui, C. Xu, Z. Jie, X. Li, J. Yang, Joint task-recursive learning for semantic segmentation and depth estimation, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 235–251.
[198] L. Wang, J. Zhang, O. Wang, Z. Lin, H. Lu, SDC-Depth: Semantic Divide-and-Conquer Network for Monocular Depth Estimation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 541–550.
[199] L. Ladickỳ, B. Zeisl, M. Pollefeys, Discriminatively trained dense surface normal estimation, in: ECCV, Vol. 2, No. 3, 2014, p. 4.
[200] X. Wang, D. Fouhey, A. Gupta, Designing deep networks for surface normal estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 539–547.
[201] D. Ferstl, C. Reinbacher, R. Ranftl, M. Rüther, H. Bischof, Image guided depth upsampling using anisotropic total generalized variation, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 993–1000.
[202] 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.
[203] J.T. Barron, B. Poole, The fast bilateral solver, in: European Conference on Computer Vision, Springer, 2016, pp. 617–632.

Raul de Queiroz Mendes received a bachelor's degree in Electrical Engineering from the Federal University of São Carlos (UFSCar), Campus São Carlos/Brazil, in 2018. In 2019, he started his master's degree from the Department of Electrical and Computer Engineering (SEL) of the São Carlos School of Engineering (EESC) of the University of São Paulo (USP) in Brazil. His main line of research is Dynamic Systems with a focus on Robotic Perception for autonomous vehicles. His interests encompass the field and subareas of Artificial Intelligence and Computer Vision.



Eduardo Godinho Ribeiro received his B.Sc. in Control and Automation Engineering from the Federal University of Lavras (Brazil) and his M.Sc. in Electrical Engineering from the University of São Paulo (Brazil), in 2018 and 2020, respectively. He is a Ph.D. student at the Laboratory of Intelligent Systems in the São Carlos campus of the University of São Paulo. His main research interests include intelligent control and deep learning applied to enhance perception and grasp ability of robotic manipulators.

Nícolas dos Santos Rosa received a bachelor's degree in Electrical Engineering with emphasis on Electronics from the São Carlos School of Engineering at the University of São Paulo (EESC/USP), in 2016. During 2014 and 2015, he received complementary undergraduate formation at the University of Michigan. In 2017, he received his master's degree in Dynamic Systems from the Department of Electrical and Computer Engineering (SEL-EESC/USP). In 2019, he started his doctor's degree at the same institution. His research addresses Dynamic Systems with a focus on Robotic Perception for autonomous vehicle. His interests encompass the field of Machine Learning and Computer Vision.

Valdir Grassi Jr. is an Assistant Professor at the Department of Electrical Engineering of the São Carlos School of Engineering at the University of São Paulo (EESC-USP). He received his Ph.D. degree in Mechanical Engineering from the Escola Politécnica at the University of São Paulo (EPUSP). His current research interests include Autonomous Mobile Robot Navigation, Motion Planning and Control, and Computer Vision applied to Robotics.