Towards Generalization Across Depth for Monocular 3D Object Detection

Andrea Simonelli[2,3], Samuel Rota Buló[1], Lorenzo Porzi[1], Elisa Ricci[2,3], and Peter Kontschieder[1]

[1] Facebook
{rotabulo, porzi, pkontschieder}@fb.com
[2] University of Trento, Trento, Italy
{andrea.simonelli, e.ricci}@unitn.it
[3] Fondazione Bruno Kessler, Trento, Italy

Abstract. While expensive LiDAR and stereo camera rigs have enabled the development of successful 3D object detection methods, monocular RGB-only approaches lag much behind. This work advances the state of the art by introducing *MoVi-3D*, a novel, single-stage deep architecture for monocular 3D object detection. *MoVi-3D* builds upon a novel approach which leverages geometrical information to generate, both at training and test time, virtual views where the object appearance is normalized with respect to distance. These virtually generated views facilitate the detection task as they significantly reduce the visual appearance variability associated to objects placed at different distances from the camera. As a consequence, the deep model is relieved from learning depth-specific representations and its complexity can be significantly reduced. In particular, in this work we show that, thanks to our virtual views generation process, a lightweight, single-stage architecture suffices to set new state-of-the-art results on the popular KITTI3D benchmark.

1 Introduction

With the advent of autonomous driving, significant attention has been devoted in the computer vision and robotics communities to the semantic understanding of urban scenes. In particular, object detection is one of the most prominent challenges that must be addressed in order to build autonomous vehicles able to drive safely over long distances. In the last decade, thanks to the emergence of deep neural networks and to the availability of large-scale annotated datasets, the state of the art in 2D object detection has improved significantly [25,18,23,24,15,11], reaching near-human performance [16]. However, detecting objects in the image plane and, in general, reasoning in 2D, is not sufficient for autonomous driving applications. Safe navigation of self-driving cars requires accurate 3D localization of vehicles, pedestrians and, in general, any object in the scene. As a consequence, depth information is needed. While depth can be obtained from expensive LiDAR sensors or stereo camera rigs, recently,

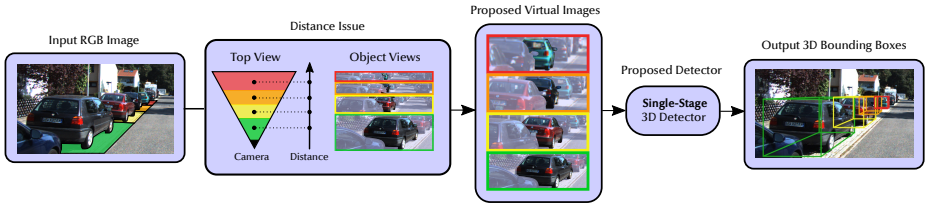

Fig. 1: We aim at predicting a 3D bounding box for each object given a single image (left). In this image, the scale of an object heavily depends on its distance with respect to the camera. For this reason the complexity of the detection increases as the distance grows. Instead of performing the detection on the original image, we perform it on virtual images (middle). Each virtual image presents a cropped and and scaled version of the original image that preserves the scale of objects as if the image was taken at a different, given depth. Colors and object masks have been used for illustrative purposes only.

there has been an increasing interest in replacing them with cheaper sensors, such as RGB cameras. Unsurprisingly, state-of-the-art 3D detection methods exploit a multi-modal approach, combining data from RGB images with LiDAR information [13,32,29,28]. However, recent works have attempted to recover the 3D location and pose of objects from a monocular RGB input [30,1,9], with the ultimate goal of replacing LiDAR with cheaper sensors such as off-the-shelf cameras. Despite the ill-posed nature of the problem, these works have shown that it is possible to infer the 3D position and pose of vehicles in road scenes given a single image with a reasonable degree of accuracy.

This work advances the state of the art by introducing *MoVi-3D*, a novel, *single-stage* architecture for ***Mo****nocular* ***3D*** *object detection*, and new training and inference schemes, which enable the possibility for the model to generalize across depth by exploiting ***Vi****rtual views*. A virtual view is an image transformation that uses geometrical prior knowledge to factor out the variability in the scale of objects due to depth. Each transformation is related to a predefined 3D viewport in space with some prefixed size in meters, *i.e.* a 2D window parallel to the image plane that is ideally positioned in front of an object to be detected, and provides a virtual image of the scene visible through the viewport from the original camera, re-scaled to fit a predetermined resolution (see Fig. 1). By doing so, no matter the depth of the object, its appearance in the virtual image will be consistent in terms of scale. This allows to partially sidestep the burden of learning depth-specific features that are needed to distinguish objects at different depths, thus enabling the use of simpler models. Also we can limit the range of depths where the network is supposed to detect objects, because we will make use of multiple 3D viewports both at training and inference time. For this reason, we can tackle successfully the 3D object detection problem with a lightweight, single-stage architecture in the more challenging multi-class setting.

We evaluate the proposed virtual view generation procedure in combination with our *single-stage* architecture on the KITTI 3D Object Detection benchmark [5], comparing with state-of-the-art methods, and perform an extensive ablation study to assess the validity of our architectural choices. Thanks to our novel training strategy, despite its simplicity, our method is currently the best performing monocular 3D object detection method on KITTI3D that makes no use of additional information at both training and inference time.

2 Related Work

In this section we provide an overview of the most recent works regarding monocular 3D object detection, grouping methods by their methodological similarity.

A first category of methods exploits geometric constraints, geometric priors or key-points. Deep3DBox [21] given an initial 2D box proposal exploits translation constraints and independent regressions to recover object position, orientation and dimensions. GS3D [12] also employs a 2D bounding box to determine a coarse 3D cuboid which is later refined via a a classification-based and quality-aware loss based on the visual features of the visible surfaces of the object. FQNet [17] performs an initial estimate of object orientation and dimensions to create a large set of 3D bounding box proposals. After this initial stage, the proposals are ranked via a fitting degree scoring mechanism to select the ones with the potentially highest 3D Intersection-over-Union (IoU) with respect to the target object. SMOKE [19] argues that the 2D detection, usually part of a 3D detection model, could introduce instability. It therefore proposes to solve the 3D detection via a dense key-point and 3D bounding box regression. ROI-10D [20] proposes to solve the 3D detection task by relying on a two-stage detection network which determines 2D proposals at its first stage and lately lifts them in 3D in its second stage. In addition, it also proposes to apply the regression loss directly on the 3D coordinates of the 3D bounding box corners. MonoDIS [30] takes a similar approach to ROI-10D and introduces a novel disentangling loss which greatly reduces the instability given by the regression of multiple groups of parameters during training. M3D-RPN [1] and SS3D [8] are the most closely-related approaches to ours. They also implement a single-stage multi-class model. In particular, the former proposes an end-to-end region proposal network using canonical and depth-aware convolutions to generate the predictions, which are then fed to a post-optimization module. SS3D [8] proposes to detect 2D key-points as well as predict object characteristics with their corresponding uncertainties. Similarly to M3D-RPN, the predictions are subsequently fed to an optimization procedure to obtain the final predictions. Both M3D-RPN and SS3D apply a post-optimization phase and, differently from our approach, these methods benefit from a multi-stage training procedure.

Another sub-category is represented by the methods which choose to convert the RGB input into alternative representations. OFTNet [26] proposes a so-called Orthographic Feature Transform which translates the RGB 2D information in a 3D voxel map. This 3D map is further reduced to a 2D bird's eye view

representation on which the detection is performed. Pseudo-Lidar [31] converts the input RGB image into a 3D point-cloud and later perform the 3D detection with state-of-the-art LiDAR approaches. The transformation from RGB to point-cloud is done by exploiting an off-the-shelf depth estimation network.

A different sub-category is defined by methods which exploit multi-task learning. Mono3D [4] focuses on the 3D bounding box proposal generation, which is carried out via an energy minimization approach. The final box scoring is assigned via the fusion of multiple tasks such as semantic segmentation, contextual information, dimensions and object location. Mono3D++ [7] combines the use of 3D-2D consistency with task priors such as depth, ground plane and shape in order to perform the 3D detection via a joint optimization. MonoGRNet [22] proposes a method to perform geometric reasoning in both the observed 2D projection and the unobserved depth dimension. For that, it combines the information of four tasks namely 2D detection, instance depth estimation, 3D location estimation and local corner regression. MonoPSR [9] exploits 2D detection methods to initialize a set of 2D proposals. Given these proposals, the method performs a dense point-cloud estimation in order to learn scale and shape information via aggregate losses including a projection alignment loss. 3D-RCNN [10] targets object shape reconstruction, which is done by exploiting class-specific shape priors learned from CAD models. The estimated shape is then used to solve the 3D detection task via an inverse-graphics framework employing a render-and-compare loss with additional refinements. Deep-MANTA [2] performs a coarse-to-fine detection which exploits the learning of 2D proposals, object part localization and visibility, as well as the similarity between an object and a set of templates.

3 Problem Description

In this work we address the problem of monocular 3D object detection, illustrated in Fig. 2. Given a single RGB image, the task consists in predicting 3D bounding boxes and an associated class label for each visible object. The set of object categories is predefined and we denote by n_c their total number.

In contrast to other methods in the literature, our method makes no use of additional information such as pairs of stereo images, or depth derived from LiDAR

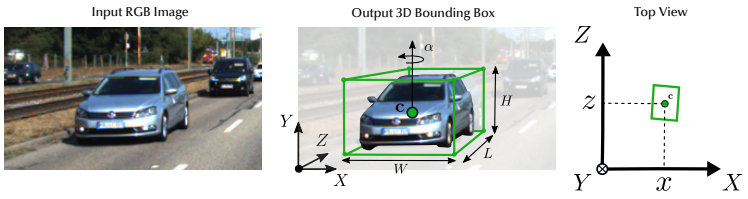

Fig. 2: Illustration of the Monocular 3D Object Detection task. Given an input image (left), the model predicts a 3D box for each object (middle). Each box has its 3D dimensions $\mathbf{s} = (W, H, L)$, 3D center $\mathbf{c} = (x, y, z)$ and rotation (α).

or obtained from monocular depth predictors (supervised or self-supervised). In order to boost their performance, the latter approaches tend to use depth predictors that are pre-trained on the same dataset where monocular 3D object detection is going to be run. Accordingly, the setting we consider is the hardest and in general ill-posed. The only training data we rely on consists of RGB images with annotated 3D bounding boxes. Nonetheless, we assume that per-image camera calibrations are available at both training and test time.

4 Details of our contributions

We will now go into the details of our contributions. We will start by explaining how to generate, both in training and inference, our proposed virtual views. Finally, we will provide the details of our proposed single-stage architecture.

4.1 Proposed Virtual Views

A deep neural network that is trained to detect 3D objects in a scene from a single RGB image is forced to build multiple representations for the same object, in a given pose, depending on the distance of the object from the camera. This is, on one hand, inherently due to the scale difference that two objects positioned at different depths in the scene exhibit when projected on the image plane. On the other hand, it is the scale difference that enables the network to regress the object's depth. In other words, the network has to build distinct representations devoted to recognize objects at specific depths and there is a little margin of generalization across different depths. As an example, if we train a 3D car detector by limiting examples in a range of maximum 20m and then at test time try to detect objects at distances larger than 20m, the detector will fail to deliver proper predictions. We conducted this and other similar experiments and report results in Tab. 4, where we show the performance of a state-of-the-art method MonoDIS [30] against the proposed approach, when training and validating on different depth ranges. Standard approaches tend to fail in this task as opposed to the proposed approach, because they lack the ability to generalize across depths. As a consequence, when we train the 3D object detector we need to scale up the network's capacity as a function of the depth ranges we want to be able to cover and scale up accordingly the amount of training data, in order to provide enough examples of objects positioned at several possible depths.

Our goal is to devise a training and inference procedure that enables generalization across depth, by indirectly forcing the models to develop representations for objects that are less dependent on their actual depth in the scene. The idea is to feed the model with transformed images that have been put into a canonical form that depends on some query depth. To illustrate the idea, consider a car in the scene and assume to virtually put a 2D window in front of the car. The window is parallel to the image plane and has some pre-defined size in meters. Given an output resolution, we can crop a 2D region from the original image corresponding to the projection of the aforementioned window on the image plane

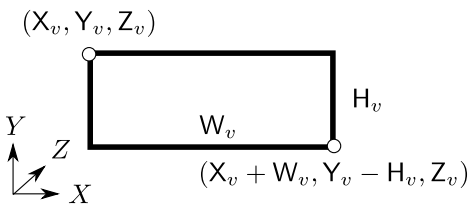

Fig. 3: Notations about the 3D viewport.

and rescale the result to fit the desired resolution. After this transformation, no matter where the car is in space, we obtain an image of the car that is consistent in terms of the scale of the object. Clearly, depth still influences the appearance, *e.g.* due to perspective deformations, but by removing the scale factor from the nuisance variables we are able to simplify the task that has to be solved by the model. In order to apply the proposed transformation we need to know the location of the 3D objects in advance, so we have a chicken-egg problem. In the following, we will show that this issue can be easily circumvented by exploiting geometric priors about the position of objects while designing the training and inference stages.

Image transformation. The proposed transformation is applied to the original image given a desired *3D viewport*. A 3D viewport is a rectangle in 3D space, parallel to the camera image plane and positioned at some depth Z_v. The top-left corner of the viewport in 3D space is given by $(\mathsf{X}_v, \mathsf{Y}_v, \mathsf{Z}_v)$ and the viewport has a pre-defined height H_v thus spanning the range $[\mathsf{Y}_v - \mathsf{H}_v, \mathsf{Y}_v]$ along the Y-axis (see Fig. 3). We also specify a desired resolution $\mathsf{h}_v \times \mathsf{w}_v$ for the images that should be generated.

The size W_v of the viewport along the X-axis can then be computed as $\mathsf{W}_v = \mathsf{w}_v \frac{\mathsf{H}_v}{\mathsf{h}_v} \frac{f_y}{f_x}$, where $\mathsf{f}_{x/y}$ are the x/y focal lengths. In practice, given an image captured with the camera and the viewport described above, we can generate a new image as follows. We compute the top-left and bottom-right corners of the viewport, namely $(\mathsf{X}_v, \mathsf{Y}_v, \mathsf{Z}_v)$ and $(\mathsf{X}_v + \mathsf{W}_v, \mathsf{Y}_v - \mathsf{H}_v, \mathsf{Z}_v)$ respectively, and project them to the image plane of the camera, yielding the top-left and bottom-right corners of a *2D* viewport. We crop it and rescale it to the desired resolution $\mathsf{w}_v \times \mathsf{h}_v$ to get the final output. We call the result a *virtual image* generated by the given 3D viewport.

Training. The goal of the training procedure is to build a network that is able to make correct predictions within a limited depth range given an image generated from a 3D viewport. Accordingly, we define a depth resolution parameter Z_{res} that is used to delimit the range of action of the network. Given a training image from the camera and a set of ground-truth 3D bounding boxes, we generate n_v virtual images from random 3D viewports. The sampling process however is not uniform, because objects occupy a limited portion of the image and drawing 3D viewports blindly in 3D space would make the training procedure very inefficient. Instead, we opt for a ground-truth-guided sampling procedure, where we repeatedly draw (without replacement) a ground-truth object and then sample

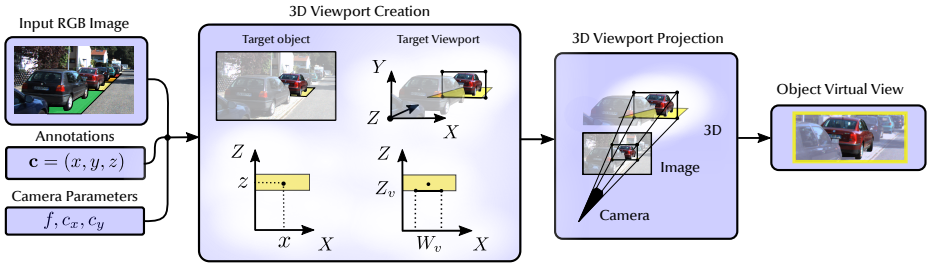

Fig. 4: Training virtual image creation. We randomly sample a target object (dark-red car). Given the input image, object position and camera parameters, we compute a 3D viewport that we place at $z = Z_v$. We then project the 3D viewport onto the image plane, resulting in a 2D viewport. We finally crop the corresponding region and rescale it to obtain the target *virtual view* (right). Colors and object masks have been used for illustrative purposes only.

a 3D viewport in a neighborhood thereof so that the object is completely visible in the virtual image. In Fig. 4 we provide an example of such a sampling result. The location of the 3D viewport is perturbed with respect to the position of the target ground-truth object in order to obtain a model that is robust to depth ranges up to the predefined depth resolution Z_{res}, which in turn plays an important role at inference time. Specifically, we position the 3D viewport in a way that $\mathsf{Y}_v = \hat{Y}$ and $\mathsf{Z}_v = \hat{Z}$, where $\hat{Y}$ and $\hat{Z}$ are the upper and lower bounds of the target ground-truth box along the Y- and Z-axis, respectively. From there, we shift Z_v by a random value in the range $[-\frac{\mathsf{Z}_{\text{res}}}{2}, 0]$, perturb randomly X_v in a way that the object is still entirely visible in the virtual image and perturb Y_v within some pre-defined range. The ground-truth boxes with $\hat{Z}$ falling outside the range of validity $[0, \mathsf{Z}_{\text{res}}]$ are set to *ignore*, *i.e.* there will be no training signal deriving from those boxes but at the same time we will not penalize potential predictions intersecting with this area. Our goal is to let the network focus exclusively on objects within the depth resolution range, because objects out of this range will be captured by moving the 3D viewport as we will discuss below when we illustrate the inference strategy. Every other ground-truth box that is still valid will be shifted along the Z-axis by $-\mathsf{Z}_v$, because we want the network to predict a depth value that is relative to the 3D viewport position. This is a key element to enforce generalization across depth. In addition, we let a small share of the n_v virtual images to be generated by 3D viewports randomly positioned in a way that the corresponding virtual image is completely contained in the original image. Finally, we have also experimented a class-uniform sampling strategy which allows to get an even number of virtual images for each class that is present in the original image.

Inference. At inference time we would ideally put the 3D viewport in front of potential objects in order to have the best view for the detector. Clearly, we do not know in advance where the objects are, but we can exploit the special

training procedure that we have used to build the model and perform a complete sweep over the input image by taking depth steps of $\frac{\mathsf{Z}_{\text{res}}}{2}$ and considering objects lying close to the ground, *i.e.* we set $\mathsf{Y}_v = 0$. Since we have trained the network to be able to predict at distances that are twice the depth step, we are reasonably confident that we are not missing objects, in the sense that each object will be covered by at least a virtual image. Also, due to the convolutional nature of the architecture we adjust the width of the virtual image in a way to cover the entire extent of the input image. By doing so we have virtual images that become wider as we increase the depth, following the rule $\mathsf{w}_v = \frac{\mathsf{h}_v}{\mathsf{H}_v} \frac{\mathsf{Z}_v}{\mathsf{f}_y} \mathsf{W}$, where W is the width of the input image. We finally perform NMS over detections that have been generated from the same virtual image.

4.2 Proposed Single-Stage Architecture

We propose a *single-stage*, fully-convolutional architecture for 3D object detection (*MoVi-3D*), consisting of a small backbone to extract features and a simple 3D detection head providing dense predictions of 3D bounding boxes.

Backbone The backbone we adopt is a ResNet34 [6] with a Feature Pyramid Network (FPN) [14] module on top. The structure of the FPN network differs from the original paper [15] for we implement only 2 scales, connected to the output of modules conv4 and conv5 of ResNet34, corresponding to downsampling factors of $\times 16$ and $\times 32$, respectively. Moreover, our implementation of ResNet34 differs from the original one by replacing BatchNorm+ReLU layers with synchronized InPlaceABN (iABN$^{\text{sync}}$) activated with LeakyReLU with negative slope 0.01 as proposed in [27]. This change allows to free up a significant amount of GPU memory, which can be exploited to scale up the batch size and, therefore, improve the quality of the computed gradients. In Fig. 5 we depict our backbone, where white rectangles in the FPN module denote 1×1 or 3×3 convolution layers with 256 output channels, each followed by iABN$^{\text{sync}}$.
Inputs. The backbone takes in input an RGB image x. If used with our proposed virtual views (Sec. 4.1), the input is represented by the set of virtual views.
Outputs. The backbone provides 2 output tensors, namely $\{f_1, f_2\}$, corresponding to the 2 different scales of the FPN network with downsampling factors of $\times 16$ and $\times 32$, each with 256 feature channels (see, Fig. 5).

3D Detection Head We build the 3D detection head by modifying the single-stage 2D detector implemented in RetinaNet [15]. We apply the detection module independently to each output f_i of our backbone, thus operating at a different scale of the FPN as described above. The detection modules share the same parameters and provide dense 3D bounding boxes predictions. In addition, we let the module regress 2D bounding boxes similar to [30,1], but in contrast to those works, we will not use the predicted 2D bounding boxes but rather consider this as a regularizing side task. Akin to RetinaNet, this module makes use of so-called *anchors*, which implicitly provide some pre-defined 2D bounding boxes

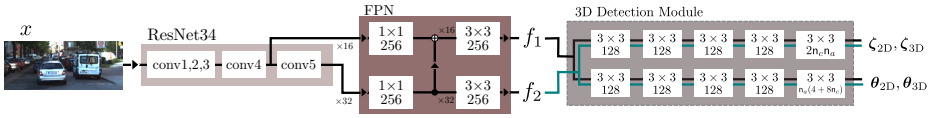

Fig. 5: Our architecture. The backbone consists of a ResNet34 with a reduced FPN module covering only 2 scales at $\times 16$ and $\times 32$ downsampling factors. The 3D detection head is run independently on f_1 and f_2. Rectangles in FPN and the 3D detection head denote convolutions followed by iABN$^{\text{sync}}$. See Sec. 4.2 for a description of the different outputs.

that the network can modify. The number of anchors per spatial location is given by n_a. Fig. 5 shows the architecture of our 3D detection head. It consists of two parallel branches, the top one devoted to providing confidences about the predicted 2D and 3D bounding boxes, while the bottom one is devoted to regressing the actual bounding boxes. White rectangles denote 3×3 convolutions with 128 output channels followed by iABN$^{\text{sync}}$. More details about the input and outputs of this module are given below, by following the notation in [30].

Inputs. The 3D detection head takes f_i, $i \in \{1, 2\}$, *i.e.* an output tensor of our backbone, as input. Each tensor f_i has a spatial resolution of $w_i \times h_i$.

Outputs. The detection head outputs a 2D bounding box and n_c 3D bounding boxes (with confidences) for each anchor a and spatial cell g of the $w_i \times h_i$ grid of f_i. Each anchor a provides a reference size (w_a, h_a) for the 2D bounding box. The 2D bounding box is given in terms of $\boldsymbol{\theta}_{\text{2D}} = (\delta_u, \delta_v, \delta_w, \delta_h)$ and $\boldsymbol{\zeta}_{\text{2D}} = (\zeta^1_{\text{2D}}, \ldots, \zeta^{\mathsf{n}_c}_{\text{2D}})$ from which we can derive

- $p^c_{\text{2D}} = (1 + e^{-\zeta^c_{\text{2D}}})^{-1}$, *i.e.* the probability that the 2D bbox belongs to class c,
- $(u_b, v_b) = (u_g + \delta_u w_a, v_g + \delta_v h_a)$, *i.e.* the bounding box's center, where (u_g, v_g) are the image coordinates of cell g, and
- $(w_b, h_b) = (w_a e^{\delta_w}, h_a e^{\delta_h})$, *i.e.* the size of the bounding box.

In addition to the 2D bounding box the head returns, for each class $1 \leq c \leq \mathsf{n}_c$, a 3D bounding box in terms of $\boldsymbol{\theta}_{\text{3D}} = (\Delta_u, \Delta_v, \delta_z, \delta_W, \delta_H, \delta_D, r_x, r_z)$ and ζ_{3D} (we omitted the superscript c). Indeed, from those outputs we can compute

- $p^c_{\text{3D|2D}} = (1 + e^{-\zeta_{\text{3D}}})^{-1}$, *i.e.* the per-class 3D bbx confidence,
- $\boldsymbol{c} = (u_b + \Delta_u, v_b + \Delta_v)$, *i.e.* the 3D bbox center projected on the image plane,
- $z = \mu^c_z + \sigma^c_z \delta_z$, *i.e.* the depth of the bounding box center, where μ^c_z and σ^c_z are class- and Z_{res}-specific depth mean and standard deviation,
- $\boldsymbol{s} = (W^c_0 e^{\delta_W}, H^c_0 e^{\delta_H}, D^c_0 e^{\delta_D})$, *i.e.* the 3D bounding box dimensions, where (W^c_0, H^c_0, D^c_0) is a reference size for 3D bounding boxes belonging to class c,
- $\alpha = \text{atan2}(r_x, r_z)$ is the rotation angle on the XZ-plane with respect to an allocentric coordinate system.

The actual confidence of each 3D bounding box is computed by combining the 2D and 3D bounding box probabilities into $p^c_{\text{3D}} = p^c_{\text{3D|2D}} p^c_{\text{2D}}$.

Losses. The losses we employ to regress the 2D bounding boxes and to learn the 2D class-wise confidence are inherited from the RetinaNet 2D detector [15]. Also the logic for the assignment of ground-truth boxes to anchors is taken

from the same work, but we use it in a slightly different way, since we have 3D bounding boxes as ground-truth. The idea is to extract the 2D bounding box from the projected 3D bounding box and use this to guide the assignment of the ground-truth box to anchors. As for the losses pertaining to the 3D detection part, we exploit the lifting transformation combined with the loss disentangling strategy as proposed in [30]. Indeed, the lifting transformation allows to sidestep the issue of finding a proper way of balancing losses for the different outputs of the network, which inherently operate at different scales, by optimizing a single loss directly at the 3D bounding box level. However, this loss entangles the network's outputs in a way that renders the training dynamics unstable, thus harming the learning process. Nonetheless, this can be overcome by employing the disentangling transformation [30]. We refer to the latter work for details.

5 Experiments

In this section we validate our contributions on the KITTI3D dataset [5]. After providing some details about the implementation of our method, we give a description of the dataset and its metrics. Then, we show the results obtained comparing our single-stage architecture *MoVi-3D* against state-of-the-art methods on the KITTI3D benchmark. Finally, to better highlight the importance of our novel technical contribution, we perform an in-depth ablation study.

5.1 Implementation details

In this section we provide the details about the implementation of the virtual views as well as relevant information about the optimization. Due to the limited space we report only the most relevant details, and refer to the supplementary material for further details about 3D head and model hyperparameters.

Virtual Views. We implement our approach using a parametrization that provides good performances without compromising the overall speed of the method. During training we generate a total of $\mathsf{n}_v = 8$ virtual views per training image, by using a class-uniform, ground-truth-oriented sampling strategy with probability $p_v = 0.7$, random otherwise (see Sec. 4.1). We set the depth resolution Z_{res} to $5m$. During inference we limit the search space along depth to $[4.5m, 45m]$. We set the dimensions of all the generated views to have height $\mathsf{h}_v = 100$ pixels and width $\mathsf{w}_v = 331$ pixels. We set the depth statistics as $\mu_z = 3m$ and $\sigma_z = 1m$.

Optimization. Our network is optimized in an end-to-end manner and in a *single* training phase, not requiring any multi-step or warm-up procedures. No form of augmentation (*e.g.* multi-scale voting) has been applied during inference.

5.2 Dataset and Experimental Protocol

Dataset. The KITTI3D dataset is arguably the most influential benchmark for monocular 3D object detection. It consists of 7481 training and 7518 test images. Since the dataset does not provide an official validation set, it is common

practice to split the training data into 3712 training and 3769 validation images as proposed in [3] and then report validation results. For this reason it is also mandatory not to limit the analysis of the results to the validation set but instead to provide results on the official test set obtained via the KITTI3D benchmark evaluation server[4]. The dataset annotations are provided in terms of 3D bounding boxes, each one characterized by a *category* and a *difficulty*. The possible object categories are *Car*, *Pedestrian* and *Cyclist*, while the object difficulties are chosen among *Easy*, *Moderate* and *Hard* depending on the object distance, occlusion and truncation. It is also relevant to note that the number of per-class annotations is profoundly different, causing the dataset to have a fairly high class imbalance. On a total of 28,8k annotations, 23.0k (79.8%) are *Car* objects, while 4.3k (15.0%) are *Pedestrian* and only 1.5k (5.2%) are *Cyclist*.
Experimental Protocol. In order to provide a fair comparison, we followed the experimental protocol of M3D-RPN [1] and SS3D [8], *i.e.* the only other available multi-class, monocular, RGB-only methods. To this end, we show results on all the KITTI3D classes obtained by means of a *single multi-class* model. For completeness we also report results of other methods (*e.g.* single-class or RBG+LiDAR), but we remark that a fair comparison is only possible with [1,8].
Evaluation Protocol. Our results follow the Official KITTI3D protocol [4]. In particular, we report scores in terms of the official 3D Average Precision (AP) metric and Bird's Eye View (BEV) AP metric. These scores have been computed with the official class-specific thresholds which are 0.7 for *Car* and 0.5 for *Pedestrian* and *Cyclist*. Recently, there has been a modification in the KITTI3D metric computation. The previous $AP|_{R_{11}}$ metric, which has been demonstrated to provide biased comparisons [30], has been deprecated in favour of the $AP|_{R_{40}}$. We therefore invite to refer only to $AP|_{R_{40}}$ and to disregard any $AP|_{R_{11}}$ score.

5.3 3D Detection

In this section we show the results of our approach, providing a comparison with state-of-the-art 3D object detection methods. As previously stated in Sec. 5.2, we would like to remind that some of the reported methods do not adopt the same experimental protocol as ours. Furthermore, due to the formerly mentioned redefinition of the metric computation, the performances reported by some previous methods which used a potentially biased metric cannot be taken into consideration. For this reason we focus our attention on the performances on the *test* split, reporting official results computed with the updated metric.
Performances on class Car. In Tab. 1 we show the results on class *Car* of the KITTI3D test set. It is evident that our approach outperforms all baselines on both 3D and BEV metrics, often by a large margin. In particular, our method achieves better performances compared to *single class* models (*e.g.* MonoDIS [30], MonoGRNet [22], SMOKE [19]) and to methods which use LiDAR information during training (MonoPSR [9]). Our method also outperforms the other *single-*

[4]Official KITTI3D benchmark http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d.

Table 1: Test set SOTA results on *Car* (0.7 IoU threshold)

Method	# classes	Training data	3D detection Easy	Moderate	Hard	Bird's eye view Easy	Moderate	Hard
OFTNet [26]	single	RGB	1.61	1.32	1.00	7.16	5.69	4.61
FQNet [17]	single	RGB	2.77	1.51	1.01	5.40	3.23	2.46
ROI-10D [20]	single	RGB+Depth	4.32	2.02	1.46	9.78	4.91	3.74
GS3D [12]	single	RGB	4.47	2.90	2.47	8.41	6.08	4.94
MonoGRNet [22]	single	RGB	9.61	5.74	4.25	18.19	11.17	8.73
MonoDIS [30]	single	RGB	10.37	7.94	6.40	17.23	13.19	11.12
MonoPSR [9]	single	RGB+LiDAR	10.76	7.25	5.85	18.33	12.58	9.91
SS3D [8]	multi	RGB	10.78	7.68	6.51	16.33	11.52	9.93
SMOKE [19]	single	RGB	14.03	9.76	7.84	20.83	14.49	12.75
M3D-RPN [1]	multi	RGB	14.76	9.71	7.42	21.02	13.67	10.23
Ours	multi	RGB	**15.19**	**10.90**	**9.26**	**22.76**	**17.03**	**14.85**

Table 2: Test set SOTA results on *Pedestrian* and *Cyclist* (0.5 IoU threshold)

Method	# classes	Training data	Pedestrian 3D Detection Easy	Moderate	Hard	Bird's eye view Easy	Moderate	Hard	Cyclist 3D Detection Easy	Moderate	Hard	Bird's eye view Easy	Moderate	Hard
OFTNet [26]	single	RGB	0.63	0.36	0.35	1.28	0.81	0.51	0.14	0.06	0.07	0.36	0.16	0.15
SS3D [8]	multi	RGB	2.31	1.78	1.48	2.48	2.09	1.61	2.80	1.45	1.35	3.45	1.89	1.44
M3D-RPN [1]	multi	RGB	4.92	3.48	2.94	5.65	4.05	3.29	0.94	0.65	0.47	1.25	0.81	0.78
MonoPSR [9]	single	RGB+LiDAR	6.12	4.00	3.30	7.24	4.56	4.11	**8.37**	**4.74**	**3.68**	**9.87**	**5.78**	**4.57**
Ours	multi	RGB	**8.99**	**5.44**	**4.57**	**10.08**	**6.29**	**5.37**	1.08	0.63	0.70	1.45	0.91	0.93

stage, *multi-class* competitors (M3D-RPN [1], SS3D [8]). This is especially remarkable considering the fact that M3D-RPN relies on a fairly deeper backbone (DenseNet-121) and, similarly to SS3D, it also uses a post-optimization process and a multi-stage training. It is also worth noting that our method achieves the largest improvements on *Moderate* and *Hard* sets where object are in general more distant and occluded: on the 3D AP metric we improve with respect to the best competing method by **+12.3%** and **+24.8%** respectively while for the BEV AP metric improves by **+24.6%** and **+33.5%**, respectively.

Performances on the other KITTI3D classes. In Tab. 2 we report the performances obtained on the classes *Pedestrian* and *Cyclist* on the KITTI3D test set. On the class *Pedestrian* our approach outperforms all the competing methods on all levels of difficulty considering both 3D AP and BEV AP. Remarkably, we also achieve better performance than MonoPSR [9] which exploits LiDAR at training time, in addition to RGB images. The proposed method also outperforms the *multi-class* models in [1,8]. On *Cyclist* our method achieves modest improvements with respect to M3D-RPN [1], but it does not achieve better performances than SS3D [8] and MonoPSR [9]. However, we would like to remark that MonoPSR [9] exploits additional source of information (*i.e.* LiDAR) besides RGB images, while SS3D [8] underperforms on *Car* and *Pedestrian* which, as described in Sec. 5.2, are the two most represented classes.

Ablation studies. In Tab. 3,4 we provide three different ablation studies. First, in 1^{st}-4^{th} row of Tab. 3 we put our proposed virtual views in comparison with a bin-based distance estimation approach. To do so, we took a common baseline,

Table 3: Validation set results on all KITTI3D classes. (0.7 IoU threshold on *Car*, 0.5 on *Pedestrian* and *Cyclist*). **V** = Virtual Views, **B** = Bin-based estimation

Method	Z_{res}	Car						Pedestrian						Cyclist					
		3D Detection			Bird's eye view			3D Detection			Bird's eye view			3D Detection			Bird's eye view		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MonoDIS [30]	–	11.06	7.60	6.37	18.45	12.58	10.66	3.20	2.28	1.71	4.04	3.19	2.45	1.52	0.73	0.71	1.87	1.00	0.94
MonoDIS+**V**	5m	**13.40**	**10.89**	**9.67**	**21.90**	**17.38**	**15.71**	**4.98**	**3.31**	**3.06**	**6.83**	**4.33**	3.38	2.09	1.07	1.00	2.70	1.42	1.31
MonoDIS+**B**	5m	7.30	5.34	4.25	12.83	8.77	7.21	3.96	3.10	2.49	4.87	3.65	3.01	0.44	0.31	0.26	0.82	0.39	0.27
MonoDIS+**B**	10m	11.64	8.36	7.25	19.07	12.98	11.39	3.37	3.13	2.53	4.56	4.21	**3.44**	**2.76**	**1.80**	**1.72**	**3.39**	**2.20**	**2.18**
MoVi-3D	5m	**14.28**	**11.13**	**9.68**	**22.36**	**17.87**	**15.73**	**7.86**	**5.52**	**4.42**	**9.25**	**6.63**	**5.06**	**2.63**	**1.27**	**1.13**	**3.10**	**1.57**	**1.30**
MoVi-3D	10m	11.58	9.54	8.54	17.98	15.16	13.98	1.82	1.27	0.94	2.38	1.78	1.34	1.08	0.51	0.51	1.84	0.97	0.89
MoVi-3D	20m	7.68	6.18	5.56	13.35	11.11	10.22	1.55	0.97	0.83	1.97	1.39	1.05	0.25	0.10	0.10	0.36	0.17	0.17

Table 4: Ablation results on *Car* obtained on different distance ranges.

Method	train range	val range	3D detection			Bird's eye view		
			Easy	Mod.	Hard	Easy	Mod.	Hard
MonoDIS [30]	far	near	0.2	0.1	0.1	0.2	0.1	0.1
MoVi-3D	far	near	**4.0**	**1.9**	**1.7**	**5.5**	**2.7**	**2.4**
MonoDIS [30]	near	far	0.2	0.1	0.1	0.2	0.2	0.1
MoVi-3D	near	far	**3.3**	**1.4**	**1.7**	**4.2**	**1.9**	**2.3**
MonoDIS [30]	near+far	middle	0.6	0.4	0.3	0.7	0.5	0.4
MoVi-3D	near+far	middle	**19.2**	**10.6**	**8.8**	**22.9**	**12.8**	**10.4**

MonoDIS [30], and modified it in order to work with both virtual views and bin-based estimation. The 1^{st} row of Tab. 3 shows the baseline results of MonoDIS as reported in [30]. In the 2^{nd} row we report the scores obtained by applying our virtual views to MonoDIS (MonoDIS+**V**). In the 3^{rd}-4^{th} rows we show the results obtained with MonoDIS with the bin-based approach (MonoDIS+**B**). For these last experiments we kept the full-resolution image as input, divided the distance range into Z_{res}-spaced bins, assigned each object to a specific bin, learned this assignment as a classification task and finally applied a regression-based refinement. By experimenting with different Z_{res} values, we found that MonoDIS+**V** performs best with $Z_{res} = 5m$ while MonoDIS+**B** performed best with $Z_{res} = 10m$. With the only exception of the class *Cyclist*, the MonoDIS+**V** outperforms MonoDIS+**B** in both 3D and BEV AP. Second, in the 3^{rd}-6^{th} rows of Tab. 3 we show the results of another ablation study in which we focus on different possible Z_{res} configurations of our proposed *MoVi-3D* detector. In this regard, we show the performances by setting Z_{res} to $5m$ (3^{rd} row), $10m$ (4^{th}) and $20m$ (5^{th}). Among the different settings, the depth resolution $Z_{res} = 5m$ outperforms the others by a clear margin. Finally, in Tab. 4 we conduct another ablation experiment in order to measure the generalization capabilities of our virtual views. We create different versions of the KITTI3D train/val splits, each one of them containing objects included into a specific *depth range*. In particular, we define a *far/near* train/val split, where the depth of the objects in the training split is in [0m, 20m] whereas the depth of the objects included into the validation split is in [20m, 50m]. We then define a *near/far* train/val split by reversing the previous splits, as well as a third train/val split regarded as *near+far/middle*

Fig. 6: Qualitative results obtained with *MoVi-3D* on KITTI3D.

where the training split includes object with depth in [0m,10m] + [20m, 40m] while the validation is in [10m, 20m]. We compare the results on these three train/val splits with the MonoDIS [30] baseline, decreasing the AP IoU threshold to 0.5 in order to better comprehend the analysis. By analyzing the results in Tab. 4 it is clear that our method generalizes better across ranges, achieving performances which are one order of magnitude superior to the baseline.

Inference Time. Our method demonstrates to achieve real-time performances reaching, under the best configuration with $\mathsf{Z}_{\text{res}} = 5m$, an average inference time of 45ms. We found the inference time to be inversely proportional to the discretization of the distance range Z_{res}. In fact, we observe that inference time goes from 13ms with $\mathsf{Z}_{\text{res}} = 20m$, to 25ms (10m), 45ms (5m).

Qualitative results. We provide some qualitative results in Fig. 6. We also provide full-size qualitative results in the supplementary material.

6 Conclusions

We introduced new training and inference schemes for 3D object detection from single RGB images, designed with the purpose of injecting depth invariance into the model. At training time, our method generates virtual views that are positioned within a small neighborhood of the objects to be detected. This yields to learn a model that is supposed to detect objects within a small depth range independently from where the object was originally positioned in the scene. At inference time, we apply the trained model to multiple virtual views that span the entire range of depths at a resolution that relates to the depth tolerance considered at training time. Due to the gained depth invariance, we also designed a novel, lightweight, single-stage deep architecture for 3D object detector that does not make explicit use of regressed 2D bounding boxes at inference time, as opposite to many previous methods. Overall, our approach achieves state-of-the-art results on the KITTI3D benchmark. Future research will focus on devising data-driven methods to adaptively generate the best views at inference time.

Acknowledgements. We acknowledge financial support from H2020 EU project SPRING - Socially Pertinent Robots in Gerontological Healthcare. This work was carried out under the *Vision and Learning joint Laboratory* between FBK and UNITN.

References

1. Brazil, G., Liu, X.: M3D-RPN: Monocular 3d region proposal network for object detection. In: ICCV. pp. 9287–9296 (2019) 2, 3, 8, 11, 12
2. Chabot, F., Chaouch, M., Rabarisoa, J., Teuliere, C., Chateau, T.: Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In: CVPR (July 2017) 4
3. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z.: Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. CoRR **abs/1512.01274** (2015) 11
4. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: CVPR (2016) 4
5. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The kitti vision benchmark suite. In: CVPR (2012) 3, 10
6. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015) 8
7. He, T., Soatto, S.: Mono3d++: Monocular 3d vehicle detection with two-scale 3d hypotheses and task priors. CoRR **abs/1901.03446** (2019) 4
8. Jorgensen, E., Zach, C., Kahl, F.: Monocular 3d object detection and box fitting trained end-to-end using intersection-over-union loss. In: CVPR (2019) 3, 11, 12
9. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: CVPR (2019) 2, 4, 11, 12
10. Kundu, A., Li, Y., Rehg, J.M.: 3D-RCNN: Instance-level 3d object reconstruction via render-and-compare. In: (CVPR) (June 2018) 4
11. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: ECCV (September 2018) 1
12. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: Gs3d: An efficient 3d object detection framework for autonomous driving. In: CVPR (2019) 3, 12
13. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3d object detection. In: CVPR (2019) 2
14. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. CoRR **abs/1612.03144** (2016) 8
15. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. CoRR **abs/1708.02002** (2017) 1, 8, 9
16. Liu, L., Ouyang, W., Wang, X., Fieguth, P.W., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. CoRR **abs/1809.02165** (2018) 1
17. Liu, L., Lu, J., Xu, C., Tian, Q., Zhou, J.: Deep fitting degree scoring network for monocular 3d object detection. CoRR **abs/1904.12681** (2019) 3, 12
18. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV (2016) 1
19. Liu, Z., Wu, Z., Tóth, R.: Smoke: Single-stage monocular 3d object detection via keypoint estimation. CoRR **abs/2002.10111** (2020) 3, 11, 12
20. Manhardt, F., Kehl, W., Gaidon, A.: Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In: CVPR (2019) 3, 12
21. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR (July 2017) 3
22. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A geometric reasoning network for 3d object localization. In: (AAAI) (2019) 4, 11, 12

23. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: CVPR (June 2016) 1
24. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. In: CVPR (2017) 1
25. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real-time object detection with region proposal networks. In: NIPS (2015) 1
26. Roddick, T., Kendall, A., Cipolla, R.: Orthographic feature transform for monocular 3d object detection. CoRR **abs/1811.08188** (2018) 3, 12
27. Rota Bulò, S., Porzi, L., Kontschieder, P.: In-place activated batchnorm for memory-optimized training of DNNs. In: CVPR (2018) 8
28. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: CVPR (2019) 2
29. Shin, K., Kwon, Y.P., Tomizuka, M.: Roarnet: A robust 3d object detection based on region approximation refinement. CoRR **abs/1811.03818** (2018) 2
30. Simonelli, A., Rota Bulò, S., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: ICCV (2019) 2, 3, 5, 8, 9, 10, 11, 12, 13, 14
31. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: CVPR (2019) 4
32. Wang, Z., Jia, K.: Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. CoRR **abs/1903.01864** (2019) 2