Monocular Differentiable Rendering for Self-Supervised 3D Object Detection

Deniz Beker[1], Hiroharu Kato[1], Mihai Adrian Morariu[1], Takahiro Ando[1], Toru Matsuoka[1], Wadim Kehl[2], Adrien Gaidon[3]

[1] Preferred Networks, Inc
[2] Toyota Research Institute - Advanced Development
[3] Toyota Research Institute

Abstract. 3D object detection from monocular images is an ill-posed problem due to the projective entanglement of depth and scale. To overcome this ambiguity, we present a novel self-supervised method for textured 3D shape reconstruction and pose estimation of rigid objects with the help of strong shape priors and 2D instance masks. Our method predicts the 3D location and meshes of each object in an image using differentiable rendering and a self-supervised objective derived from a pretrained monocular depth estimation network. We use the KITTI 3D object detection dataset to evaluate the accuracy of the method. Experiments demonstrate that we can effectively use noisy monocular depth and differentiable rendering as an alternative to expensive 3D ground-truth labels or LiDAR information.

1 Introduction

Autonomous driving relies heavily on 3D object perception for safe navigation. Most existing systems leverage active sensors (e.g., LiDAR, radar) for location estimation, yet they are either prohibitively costly for large-scale deployment or too sparse in spatial coverage. For these reasons, research in monocular 3D object detection has seen rising popularity in recent years.

Ongoing advancements have led to steadily improving detection accuracy [3,25,31]. Despite foregoing active sensing, they still require supervision in the form of 9D cuboid labels which encode 3D location, rotation and metric object dimensions. Most often, such labels are obtained with the help of annotation pipelines and 3D LiDAR point clouds, demanding the usage of costly human labour and expensive sensors. Alternatively, one can leverage self-supervised autolabeling techniques that employ strong shape priors and optimize 3D alignment in stereo point clouds [5,6,35] or mixed RGB/LiDAR setups [40].

In this paper, we explore a novel solution towards tackling 3D location and shape estimation from 2D instance mask detections, requiring only monocular input and geometric priors for self-supervision. Recently, self-supervised depth estimation networks [9,28] have become interesting alternatives to LiDAR sensing. They are generally trained using only video sequences and are able to estimate depth with absolute scale when combined with weak supervision (e.g.,

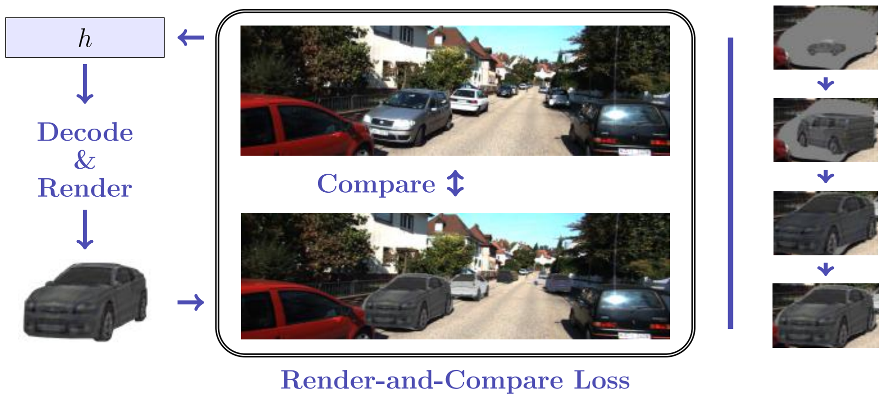


Fig. 1: We propose a self-supervised optimization pipeline for monocular 3D object estimation via analysis-by-synthesis over object pose, metric dimensions, shape and texture. Left: Starting from a 2D instance mask detection, we feed the latent variables h into our decoder network and use differentiable rendering to obtain 2D projections. We use various render-and-compare losses over multiple quantities for comparative analysis and back-propagate the error. Right: The fitting process over multiple iterations. Starting from a random initialization, we can recover the actual object properties quite well.

velocity of the ego-camera [9]). However, they tend to be rather noisy, overall less accurate than LiDAR, and therefore not well-suited for precise 3D object localization and metric dimension estimation. We therefore incorporate additional strong priors over learned textured object shapes for regularization and run comparative scene analysis via differentiable rendering [22,12,21,2]. This allows for optimizing 3D variables against image evidence via self-supervision and back-propagation. While initial work focused mostly on toy examples, recent papers have successfully explored applications of analysis-by-synthesis in the wild [17,16,42,40] and we follow in their path.

We present an example optimization of our pipeline in Fig. 1. Starting from a 2D detector that produces 2D boxes with associated instance masks, we optimize over the 3D location, rotation, metric dimensions as well as the object's shape and texture. In our paper, we handle metric dimension as 3D scaling. For the actual alignment computation, we leverage complimentary cues in the form of instance masks, RGB values as well as monocular depth, and regularize with our textured shape space. While our initial estimation is quite rough, our method is able to converge to a good solution while traversing the possible space of shapes, textures, metric sizes and poses. This process is run sequentially until all 2D detections in the scene have been parsed and transformed into 3D estimates.

To summarize our main contributions: Firstly, we remove the necessity for 3D sensing or ground-truth information, making our 3D object estimation pipeline a truly monocular approach. Secondly, we propose the idea of regularizing noisy monocular depth maps at the instance level by strong geometric object priors. Lastly, we evaluate our pipeline on the KITTI 3D dataset [7] and show that we can achieve comparable accuracy to SoTA methods that rely on 3D supervision.

2 Related Work

Due to the huge body of related 3D detection work, we will focus our survey mostly on monocular methods. Mono3D [3] is an early work which introduces a region proposal method tailored to autonomous driving. This method computes the 3D bounding box by exploiting different cues such as semantic segmentation, instance segmentation, shape, context features and absolute locations in 2D/3D space. Importantly, they incorporate priors to limit the cuboid search space by using pre-defined object sizes and orientations as well as assuming a known ground plane. Deep3DBox [27] estimates cuboids from 2D bounding boxes, assuming that the former should be tightly fit to the latter when projected onto the image plane. This strong assumption can underperform in cases where the actual 2D bounding boxes are not tightly enclosing the objects due to occlusion or truncation. The advent of unsupervised/self-supervised monocular depth estimation [8,24,28] saw their inclusion in recent detection work. In Multi-Fusion [38] 3D region proposals are generated by backprojecting 2D proposals to the 3D space using monocular depth. Similarly, ROI-10D [25] employs such depth maps to robustify their bounding box lifting from the 2D to the 3D space.

Methods based on the idea of Pseudo-LiDAR [36,23] leverage reprojected monocular depth from off-the-shelf models and run detection networks that have been originally designed for LiDAR input with impressive accuracy improvements. All the mentioned approaches benefit greatly from the employed monocular depth modules but their analysis shows that their overall performance is heavily dominated by the accuracy of depth estimation. For this reason, regularization from additional priors is desirable.

In terms of analysis-by-synthesis, much novel work has been presented in respect to differentiable rendering. The works [22,12,2,18] propose different ways to produce gradients for the rasterization of triangle meshes. Building on such ideas, both [17] as well as [40] present render-and-compare optimization frameworks with learned shape spaces for automotive scenarios. The former uses 2D instance masks for estimating the shape and up-to-scale pose whereas the latter leverages LiDAR observations for full 3D estimation.

There exists other slightly-related work that leverages learned shape spaces for automotive object retrieval that we discuss for completeness. In [5] a detector initializes object instances which are further optimized for pose and shape priors over stereo depth. The authors later extended it [6] for temporal priors to also recover pose trajectories. In a similar vein, [35] runs full 3D object pose and shape recovery over stereo depth. The authors of [33] explore probabilistic 3D object completion via a shape space and LiDAR as weak supervision. Their work assumes correct localization and focuses solely on the reconstruction quality.

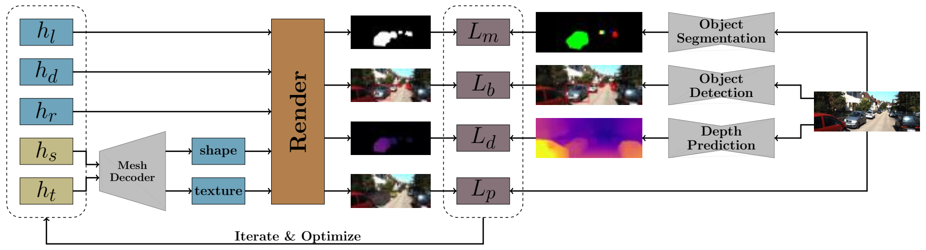


Fig. 2: Overall pipeline of our method where h_l, h_d, h_r are the variables for location, dimension and rotation, and h_s, h_t are the latent variables for shape and texture. We denote with L_m, L_b, L_d, L_p the losses for the silhouette, bounding box, depth map and pixel colors respectively.

3 Method

As mentioned, most existing monocular 3D detection methods rely on supervised learning, requiring 3D cuboid information. Instead, we propose to estimate all required 3D object properties via self-supervision and differentiable rendering, avoiding any form of 3D annotation. We depict our pipeline in Fig. 2 and will provide an overview before going into more detailed descriptions in the next subsections.

We initially run off-the-shelf object detectors on an image to produce masked 2D detections. For each detection, we place an initial estimate into 3D space by instantiating all object properties. Concretely, we capture metric location $h_{\text{loc}} \in \mathbb{R}^3$, metric dimension $h_{\text{d}im} \in \mathbb{R}^3$, rotation along the yaw axis $h_{\text{rot}} \in \mathbb{R}$, shape $h_{\text{s}h} \in \mathbb{R}^{D_{\text{s}h}}$ and texture $h_{\text{t}x} \in \mathbb{R}^{D_{\text{tx}}}$. Following the parameterization introduced in ROI-10D [25], we formulate the location h_{loc} as the difference between the center of 2D bounding box and the projected 3D centroid position $(h_{\text{u}}, h_{\text{v}}) \in \mathbb{R}^2$, together with the depth of the centroid $h_{\text{d}ep}$. We initialize $h_{\text{u}} = h_{\text{v}} = 0$, whereas $h_{\text{d}ep}$ are set $h_{\text{d}im}$ to the mean depth and dimensions. We also initialize $h_{\text{r}ot}, h_{\text{s}h}, h_{\text{t}x}$ to random numbers sampled from a Gaussian with mean $\mu = 0$ and standard deviation $\sigma = 0.1$.

After this initialization, we run an iterative pipeline of 1) rendering, 2) projective loss computation over multiple different cues and 3) backpropagation to our latent object variables. To produce a rendering, we feed $h_{\text{s}h}$ and $h_{\text{t}x}$ through generative models to produce object shape and texture. We then rescale and rotate using $h_{\text{d}im}$ and h_{rot}, place the object in the 3D space using h_{loc} and finally render an RGB/D image pair with the differentiable renderer implementation from [12]. Note that we work at metric scale and therefore require known camera intrinsics during optimization.

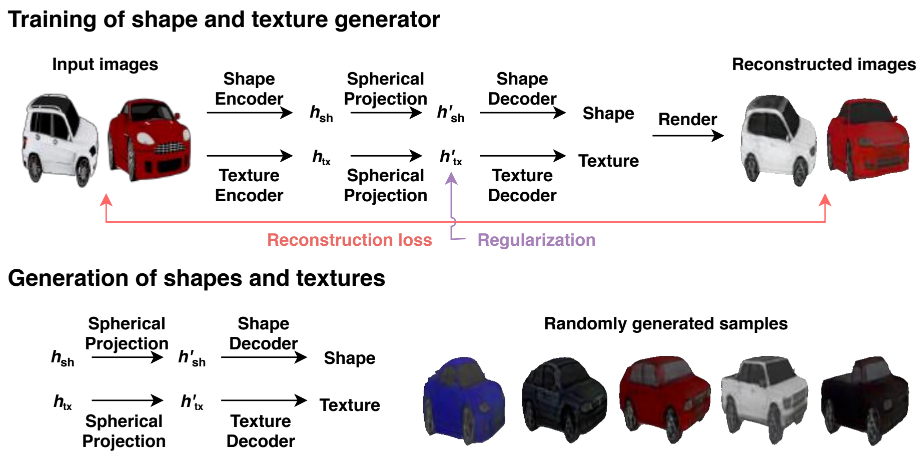


Fig. 3: Illustration of our shape and texture generator. During training, images of cars are encoded to shape and texture vectors that live on hyperspheres. These vectors can then be decoded to 3D shapes and texture maps, which in turn are rendered and used to compute reconstruction losses. After training, the decoder part can be used independently to generate novel textured shapes.

2D Object Detection and Segmentation To identify all instances in the scene, we use an off-the-shelf Mask R-CNN [10] model with the X-152 backbone from detectron2 [37], trained on COCO [19] and thresholded at 0.1.

To compute our rendering losses, we need a clear foreground/background separation. In order to produce a background mask, we compute a union over all detection masks to first obtain a foreground mask, and then invert it. We will evaluate the majority of our experiments with this approach. Alternatively, one can also leverage semantic segmentation for the separation and we present an ablative comparison in the experimental section.

Rendering We use the publicly available differentiable renderer from Kato et al. [12] and although they introduced their method for color and silhouettes only, we extend it for rendering depth maps in a differentiable way. We exploit the simple fact that a depth map is differentiable with respect to the z-coordinates of a surface without any approximations because the depth value at a pixel is computed by a weighted sum of z-coordinates of the surfaces at the pixel.

Shape and Texture Generation Differentiable rasterization allows us to establish losses between renderings and an input image (*render-and-compare*). This approach was initially adopted by 3D-RCNN [17] in which shapes were projected to 2D silhouettes. Unfortunately, silhouettes retain only a fraction of the original information and have an especially strong impact on rotation estimation.

To leverage image information as much as possible, we therefore additionally reconstruct textures and render a joint representation of textured shapes.

To encode a variety of object shapes and appearances, we employ a generative model that is able to produce a mesh and a texture map for given inputs. Concretely, we use the single-image 3D object reconstruction method presented in [14] and provide an illustration in Figure 3. This method adopts an encoder-decoder architecture that transforms an image into a low-dimensional latent space which in turn can be used to generate a 3D object. Even though we train the entire encoder-decoder architecture beforehand, we discard the encoder and only use the decoder later on.

We would like our generator to produce plausible shapes and textures for any decoding input h_{sh} and h_{tx}, ensuring that our latent spaces are smoothly traversable without collapsing. Therefore, unlike [14], we project the hidden vectors onto a unit hypersphere, similar to [40]. To distribute h'_{sh} and h'_{tx} uniformly on the hypersphere, we employ a regularization technique where we take random samples on the sphere and pull the closest latent vectors towards those random samples. Let r_{sh} and r_{tx} be random samples uniformly distributed on the hyperspheres of D_{sh} and D_{tx} dimensions. We add the following term to the loss function:

$$L_h = \frac{1}{N_b}\left(\sum_{i=1}^{N_b} \min_j |r^i_{\text{sh}} - h^j_{\text{sh}}|_1 + \sum_{i=1}^{N_b} \min_j |r^i_{\text{tx}} - h^j_{\text{tx}}|_1\right). \tag{1}$$

N_b is the size of minibatch, and $h'^i_{\text{sh}}, h'^i_{\text{tx}}, r^i_{\text{sh}}, r^i_{\text{sh}}$ represent i-th sample in the minibatch. The encoder-decoder architecture is trained using synthetic views of 3D car models taken from the ShapeNet dataset [1]. We found a latent dimensionality of 8 for the shape D_{sh} and texture D_{tx} spaces to work quite well.

Monocular Depth Estimation To weaken the projective entanglement between object dimensions, locations and 2D appearance, we employ depth maps extracted by a pre-trained, state-of-the-art monocular depth estimator that is trained with self-supervision (PackNet [9]). The scale of the estimated depth is calibrated from weak supervision in the form of ego-camera velocity.

3.1 Loss Functions

For self-supervision we propose four loss functions between renderings and image evidence: L_p which penalizes pixel color deviations, L_b to maximize 2D bounding box overlap, L_m for silhouette alignment,and L_d for depth map differences. We notate an input image of $H \times W$ pixels as $I^{\text{R}GB} \in \mathbb{R}^{H\times W\times 3}$, the foreground map and depth map estimated from the image as $I^{\text{F}} \in \mathbb{R}^{H\times W}$ and $I^{\text{D}} \in \mathbb{R}^{H\times W}$, respectively. In addition, assuming that the number of detected objects is N_o, we notate the cropped and resized region of the i-th object from these maps as $I_i^{\text{R}GB} \in \mathbb{R}^{H_c\times W_c\times 3}$, $I_i^{\text{F}} \in \mathbb{R}^{H_c\times W_c}$, and $I_i^{\text{D}} \in \mathbb{R}^{H_c\times W_c}$. For the reconstructed counterparts, we use the same notations with *hat*.

Additionally, we use a regularization term $L_{\mathrm{d}im}$ to penalize unrealistic dimensions. We define it as the L1 distance between the estimated dimension and the mean dimension of objects in the dataset. The overall loss function is defined as the weighted sum of all the loss functions and the regularization term:

$$L = \lambda_p L_p + \lambda_b L_b + \lambda_m L_m + \lambda_d L_d + \lambda_{\mathrm{d}im} L_{\mathrm{d}im}. \tag{2}$$

Pixel Map Difference The pixel map difference loss is defined as L1 distance between rendered image and input image by using equation

$$L_p = \frac{1}{H_c W_c} \sum_{i=1}^{N_b} \sum_{j=1}^{H_c} \sum_{k=1}^{W_c} |I_{ijk}^{\mathrm{R}GB} - \hat{I}_{ijk}^{\mathrm{R}GB}|. \tag{3}$$

Bounding Box Difference We compute the bounding boxes of the objects from the rendered image. Let IoU_i^B be the intersection over union between input and rendered 2D bounding boxes of i-th object. We define the bounding box loss as

$$L_b = \sum_{i=1}^{N_b} \mathrm{maximum}(1 - \mathrm{IoU}_i^B - t_b, 0). \tag{4}$$

As the predicted bounding boxes may be noisy, we allow the IoU to have a small error margin of up to t_b. This loss is differentiable with respect to the vertices of the reconstructed shapes as the bounding box is computed by differentiable projection of vertices to image coordinates. Fitting estimated 3D bounding boxes to detected 2D bounding boxes is a well-known approach [27], and some works [26] use IoU as a metric. Different from these works, we compute projected bounding boxes using 3D shapes instead of 3D bounding boxes.

Silhouette Difference Concretely, assuming that $I_{ijk}^{\mathrm{F}} = 1$ if the pixel jk of the i-th image is foreground and $I_{ijk}^{\mathrm{F}} = 0$ if background, the IoU of i-th image is

$$\mathrm{IoU}_i^F = \frac{\sum_{j=1}^{H_c} \sum_{k=1}^{W_c} I_{ijk}^{\mathrm{F}} \hat{I}_{ijk}^{\mathrm{F}}}{\sum_{j=1}^{H_c} \sum_{k=1}^{W_c} I_{ijk}^{\mathrm{F}} + \hat{I}_{ijk}^{\mathrm{F}} - I_{ijk}^{\mathrm{F}} \hat{I}_{ijk}^{\mathrm{F}}}, \tag{5}$$

$$L_m = \sum_{i=1}^{N_o} L_{m_i} = \sum_{i=1}^{N_o} (1 - \mathrm{IoU}_i^F). \tag{6}$$

Depth Map Difference The depth map loss is defined as the L1 difference between the rendered depth and model-predicted depth by using the formula

$$L_d = \frac{1}{H_c W_c} \sum_{i=1}^{N_b} \sum_{j=1}^{H_c} \sum_{k=1}^{W_c} I_{ijk}^{\mathrm{F}} \hat{I}_{ijk}^{\mathrm{F}} |I_{ijk}^{\mathrm{D}} - \hat{I}_{ijk}^{\mathrm{D}}|. \tag{7}$$

Note that during the optimization, all the pre-trained networks are used with fixed weights and are not optimized. We disable back-propagation for $\hat{I}^{\mathrm{F}}_{ijk}$ and back-propagate gradients only through $\hat{I}^{\mathrm{D}}_{ijk}$.

3.2 Escaping Rotational Local Minima

Estimating rotation with render-and-compare approaches can easily lead to local minima [13,34] due to non-linear objectives and visual ambiguities. To deal with this issue, we explore several rotations at every step of the optimization. We sample 4 different variations (h_{rot}, $h_{rot} \pm 15°$, -h_{rot}) as well as a random angle sampled from a uniform distribution. If an angle r_0 gives a lower error with the perceptual metric [41] than the one used at current iteration, we set h_{rot} to r_0. Also, the size of the input and the reconstructed images are adjusted to ensure that the area of the object bounding boxes is the same, and the background region is masked out using the estimated segmentation mask.

3.3 Detection Confidence Score

MonoDIS [31] proposes to learn confidences of 3D detections by weighting confidences of corresponding 2D detections. As their method requires ground-truth 3D bounding boxes for confidence training, it is not applicable in a straightforward manner to our problem. Instead, we propose to weight 2D confidences using our reconstruction errors after optimization. To this end, we compute the silhouette reconstruction loss L_{m_i} of the i-th image, and the ratio of protrusion b_i between rendered bounding box and detection bounding box. Our confidence score c_{3D} is then defined by using the formula

$$c_{3D} = c_{2D} e^{-\alpha_m L_{m_i}} e^{-\alpha_b b_i}, \tag{8}$$

with two hyperparameters α_m, α_b and the original 2D detection confidence c_{2D}.

4 Experiments

We evaluate the proposed method on KITTI 3D [7], one of the most popular benchmarks for 3D object detection in autonomous driving. KITTI 3D is composed of synchronized RGB images/LiDAR frames, for which annotations of 2D bounding boxes, 3D object location, 3D object dimension and 1D object rotation angle along the y-axis (yaw) are provided. Unlike other approaches, our method does not require 3D ground-truth or LiDAR observations in any part of the pipeline. For a fair comparison to similar work, we use the same training and validation splits proposed in [4] and focus on the *car* category.

Evaluation Metrics KITTI 3D uses different AP metrics to measure 2D and 3D detection accuracy at different cut-off thresholds (usually 0.5 and 0.7). For each detection, the IoU overlap with a ground truth is computed either on the image plane (2D), in bird's eye view (BEV), or in volumetric 3D space. Following a recent suggestion by [31], we use the $AP|_{R_{40}}$ metric that is currently used in the official KITTI 3D benchmark leaderboard, instead of the older $AP|_{R_{11}}$ metric. For reference and a fair comparison to previously established work, we also evaluate against the $AP|_{R_{11}}$ metric and share the results in the supplement.

System Configuration All experiments are performed on a Linux-based cluster with 8 V100 (32GB) GPUs. Running the full pipeline on the KITTI 3D *validation* dataset takes approximately one day.

Hyperparameters Tuning The hyperparameter values are set to $\lambda_b = \lambda_m = \lambda_p = 1$, $\lambda_d = 0.2$, $\lambda_{dim} = 0.1$, and $t_b = 0.1$. We use the Adam optimizer [15] with $\alpha = 0.03$, $\beta_1 = 0.5$ and $\beta_2 = 0.9$. For each detected object, we run the optimization pipeline for 150 iterations. The hyperparameters for confidence weighting are set to $\alpha_m = 0.1$ and $\alpha_b = 0.03$.

4.1 Comparison to SoTA

As a first experiment, we run our pipeline on the validation set and show our quantitative results in Table 1 where we compare our inferred 3D boxes against the ground truth labels. As can be seen, we position ourselves between the two leading monocular 3D detectors MonoDIS [31] and MoVI-3D [32] for the easy instances and fall slightly behind for the moderate and difficult cases. This shows that our self-supervised 3D estimation method can be competitive with fully supervised detectors trained on manually labeled 3D ground truth cuboids. For more difficult instances, the monocular depth maps become noisier at longer ranges, there is an increasing amount of occlusion, and there are smaller objects, all factors that challenge our losses for bounding box, silhouette, and pixel differences. The autolabeling solution from [40] leverages shape priors similar to us, yet has access to LiDAR observations that greatly benefit their scale and 3D location estimation. Nonetheless, especially for the more challenging 3D AP metric, we can retrieve much more accurate object estimates overall. One of our major differences over LiDAR equipped SDFLabel paper is the scale regression method. SDFLabel often fail the strict 3D AP metric due to small deviations in 3D scale regression, especially at range where very few LiDAR points are on the object. On the other hand, we leverage the dimension statistics of the cars as a strong additional prior over generic driving scene instances. While SDFLabel regress directly on metric scale, we regressed the deviation from the statistics.

We also present some qualitative results in Figure 4. It can clearly be seen that our priors provide enough guidance to estimate the correct 3D locations and rotations of the objects even if the predicted depth map is noisy. The depth map provides weak supervision for the 2D loss to estimate the correct object scale and rough 3D location but is ultimately regularized by shape and texture. It is also evident that optimizing for the 2D bounding box IoU loss L_b results in 3D shape projections that tightly fit into the predicted 2D bounding boxes. Besides, the regularization loss L_{dim} on dimensions prevents the predicted shapes from being too different from the mean object shape.

4.2 Ablation Studies

Reconstruction Loss Firstly, we analyze the contribution of each individual loss component on the final outcome. Table 2 shows that all losses have a significant impact. Among them, L_d is the most important, as the accuracy sharply drops by over 60% when it is removed from the loss function. This shows that using 2D loss functions alone is not as effective, and that even noisy knowledge of scene depth (either implicitly or explicitly) is essential for monocular object estimation tasks. On the other hand, optimizing only for L_d leads to an accuracy close to zero. This signifies that using depth alone is apparently not enough for accurate reconstruction. In such cases, the optimization is disregarding object boundaries and physical extents. For example, the detected and reconstructed silhouettes should overlap, but the loss function does not enforce this constraint. Additionally, the objects can warp into any possible size.

We experimented to integrate depth information by backprojecting depth maps into 3D point clouds similar to Pseudo-LiDAR [39]. We optimized the Hausdorff distance between predicted meshes and the point clouds, and also experimented with Procrustes variants to support rotation, translation and scale. However, both results were very similar to our current *depth-only* results.

Another strong impact can be seen from the silhouette loss L_m. This loss back-propagates 2D silhouette mismatches to penalize the reconstruction of the 3D shape. Along with the regularization loss of dimensions L_{dim}, it helps converging towards a realistic shape that matches the predicted mask.

On the other hand, the impact of bounding box IoU loss L_b and pixel color loss L_p have less impact on the final accuracy. L_b strongly enforces 2D projection

Method	Supervised	LiDAR	3D detection			Birds eye view		
			Easy	Moderate	Hard	Easy	Moderate	Hard
MonoDIS [31]	✓		11.06	7.60	6.37	18.45	12.58	10.66
MoVI-3D [32]	✓		14.28	11.13	9.68	22.36	17.87	15.73
SDFLabel [40]		✓	1.23	0.54	n/a	15.7	10.52	n/a
MonoDR (ours)			12.50	7.34	4.98	19.49	11.51	8.72

Table 1: Evaluation of different monocular 3D detection methods: We report $AP_{|R_{40}}$ on the KITTI 3D *validation* set. The values are calculated assuming an intersection-over-union (IoU) between the predicted and ground-truth bounding boxes (in bird's-eye view) of at least 0.7.

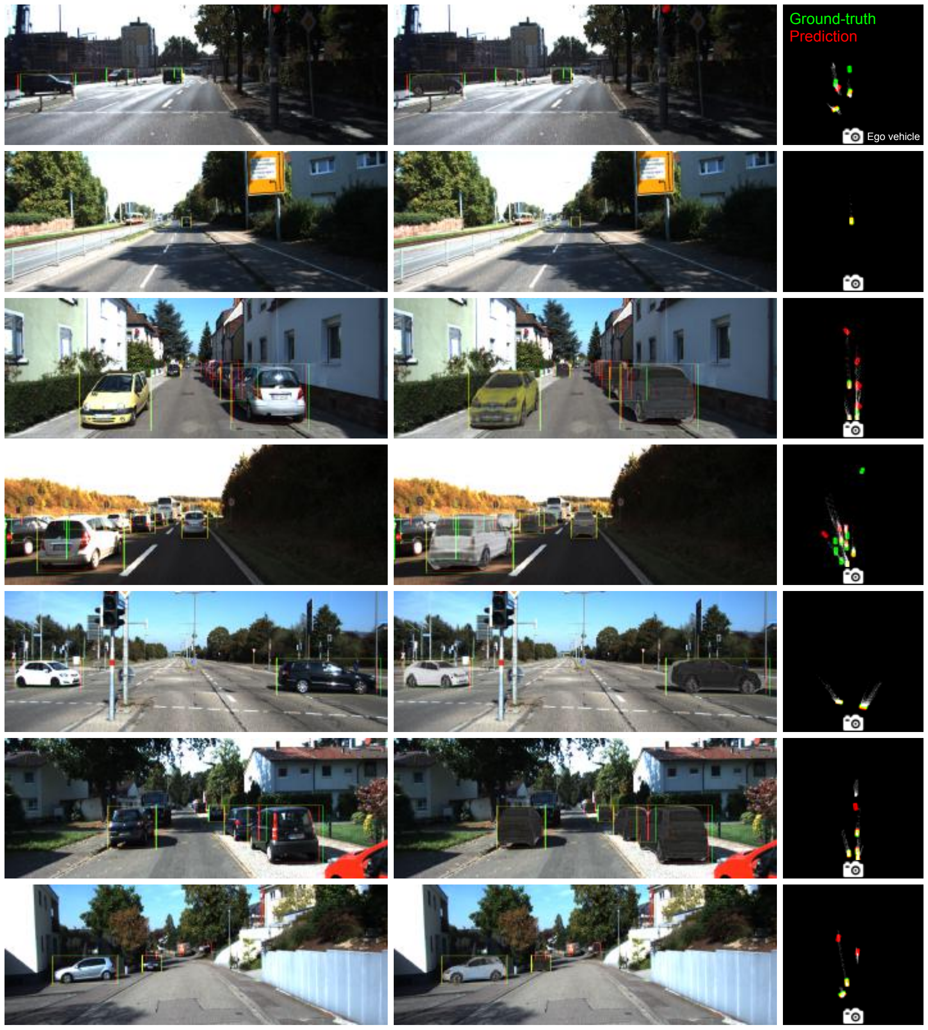


Fig. 4: Representative results obtained through our method: Raw images with predicted 2D bounding boxes and instance masks are shown in the left column. Reconstructed images (with the projection of an estimated 3D shape onto the image plane) are shown in the middle column. Ground-truth (green)/predicted (red) 3D bounding boxes in the bird's-eye view (BEV) are shown in the right column. The white points in BEV represent projections of object point clouds generated from predicted depth maps. The ground truth information is only used for visualization purposes.

of the reconstruction to be within the bounding box limits, which can be interpreted as an auxiliary support for the silhouette loss L_m. Similarly, the pixel loss

Reconstruction loss	3D Detection			Birds eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard
full (L)	12.50	7.34	4.98	19.49	11.51	8.72
w/o depth ($L - \lambda_d L_d$)	4.82	2.88	1.91	7.96	4.96	3.64
depth only (L_d)	0.00	1.00	0.00	0.00	0.00	0.00

Table 2: Evaluation of different reconstruction losses on KITTI 3D *val* set.

Confidence weighting	3D detection			Birds eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard
full (c_{3D})	12.50	7.34	4.98	19.49	11.51	8.72
w/o silhouette recon. ($c_{2D}e^{-\alpha_b b_i}$)	11.19	6.81	4.61	17.72	10.80	8.20
w/o box protrusion ($c_{2D}e^{-\alpha_m L_{m_i}}$)	12.43	7.29	4.89	19.29	11.34	8.64
w/o both (c_{2D})	10.74	6.48	4.40	17.19	10.36	7.95

Table 3: Evaluation of different confidence weighting schemes on the *val* set.

impacts primarily the texturing of objects and helps mostly for minor textural misalignments but can hardly recover larger displacements or affect the shape.

Confidence Weighting In Table 3, we illustrate the impact of our proposed confidence weighting schemes. In the last row, we present the accuracies calculated by using the scores obtained from 2D bounding box detection without applying any confidence weighting. In the first row, we present the result after applying the confidence weighting over all scores. Applying both weights improves the accuracy by 11% - 16%. The second row shows that weighting the scores by using the silhouette reconstruction loss value significantly impacts the increase.

Model	Box AP	Mask AP	2D detection			3D detection			Bird's Eye View		
	COCO	COCO	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mask R-CNN X152	50.2	44.0	82.24	74.82	59.34	12.50	7.34	4.98	19.49	11.51	8.72
Panoptic R101-FPN	42.4	-	78.59	72.17	59.66	9.26	5.84	4.35	15.42	9.97	7.44

Table 4: Different segmentation networks, on KITTI 3D *validation* set.

Effect of the Segmentation Masks In Table 4, we present the impact of the chosen 2D object detection and segmentation network on the final accuracy. We compared the results of Mask R-CNN X-152 [10] and Panoptic Segmentation R101-FPN, both taken from detectron2[37] and pretrained on COCO [19]. Mask R-CNN has significantly better mAP than the panoptic segmentation model, especially for the easy and moderate instances. Even though panoptic segmentation provides better semantic masks, it provides worse 2D detection performance for occluded objects. This is due the general nature of panoptic segmentation, since it derives its bounding boxes tightly around the segmentations. As our method is strongly dependent on the tightness of 2D bounding boxes, panoptic segmentation has an overall negative impact on the accuracy.

4.3 Limitations and Failure Cases

The quality of the estimated bounding boxes and segmentation masks clearly impacts the rendering used for computing the loss. Inaccuracies in their estimation will be reflected in the quality of the estimated 3D shape and object pose. Figure 5a shows two example of failure cases due to stronger car occlusions. Because the 2D bounding boxes used in this work are not *amodal*, they only confine the visible parts of the cars. In this case, optimization is done by projecting the 3D shape onto a partial view of the car, which leads to a large error in the rotation estimation. This problem could be solved by fine-tuning the detector on a dataset with annotated amodal 2D bounding boxes.

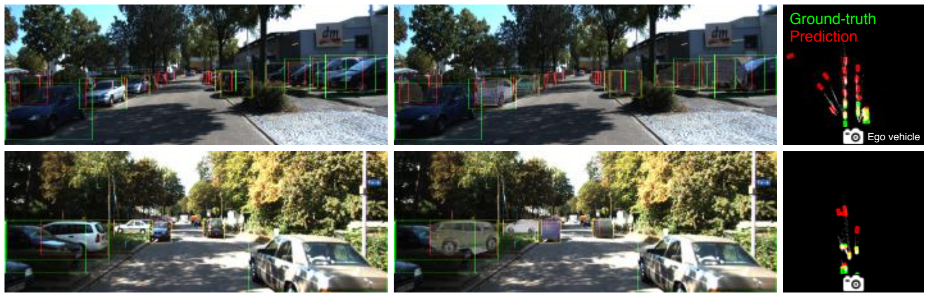
The size of the estimated 3D shape is also constrained by the size of the estimated 2D bounding box. Figure 5b shows an example of a failure case where the 2D bounding box is larger than the object (non-tight). Because the optimization requires the projection of the 3D shape to fit the 2D bounding box, the shape is constrained to be larger than it should be. In this case, a large translation error can be seen due to a shift in the center of mass.

The computational cost is currently a bottleneck of our method. The current computation time is proportional to the number of detected objects and the mean computation time per image on the KITTI *validation* set is three minutes. The major time is spent on escaping the local minima of the rotation 3.2 as we had to render multiple times per image.

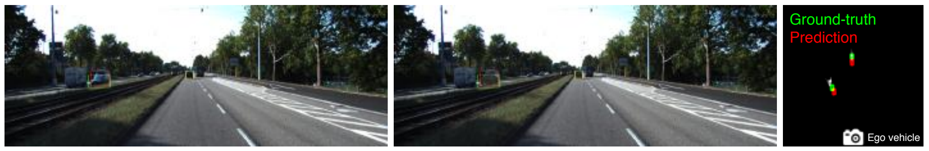
5 Conclusion

We presented a self-supervised approach, based on differentiable rendering, for 3D shape reconstruction and localization of rigid objects from monocular images. Although used in the context of autonomous driving, our method is generally applicable to many categories and scenarios. We showed that it is possible to use noisy monocular depth and differentiable rendering in conjunction with learned object priors as an alternative to expensive 3D ground-truth labels or LiDAR information.

Future work should investigate alternative approaches to estimating depth from monocular images and registering point clouds. Although significantly improved over the last years, accurate long-range depth estimation is still a challenging problem. This information is of vital importance in autonomous driving,



(a) Example of failures caused by occlusion. The cars on the right side (top row) and the car on the left side (bottom row) are partially visible and the estimation of their 2D bounding boxes is not accurate, which leads to large errors in rotation estimation.



(b) Example of a failure caused by a non-tight 2D bounding box. The inaccurate 2D bounding box estimation leads to a large error in 3D shape estimation and pose translation.

Fig. 5: Two examples of cases where our method fails to estimate accurate object pose translations or rotations. Ground-truth (green)/predicted(red) 3D bounding boxes in the birds-eye view (BEV) are shown in the right column. The white points in BEV represent projections of object point clouds generated from predicted depth maps. The ground truth information is only used for visualization purposes.

where the system often needs to make quick decisions. Having accurate information about the surrounding environment, as early as possible, allows for better risk assessment, planning and control.

Accurate and fast point cloud registration, on the other hand, is necessary in our pipeline for achieving self-supervision. Many of the current techniques are either slow or require an initial guess for accurate registration. This makes them impractical for applications where real-time inference is required.

6 Implementation Details

6.1 Parameterization and Initialization

An object's properties are defined as follows:

- Location: $h_{\text{loc}} \in \mathbb{R}^3$
- Dimensions: $h_{\text{dim}} \in \mathbb{R}^3$

- Rotation: $h_{\text{rot}} \in \mathbb{R}$
- Shape encoding: $h_{\text{sh}} \in \mathbb{R}^{D_{\text{sh}}}$
- Texture encoding: $h_{\text{tx}} \in \mathbb{R}^{D_{\text{tx}}}$

Let $b_{\text{top}}, b_{\text{left}}, b_{\text{bottom}}, b_{\text{right}}$ be the coordinates of a detected 2D bounding box. We define the following:

$$c_u = \frac{b_{\text{top}} + b_{\text{bottom}}}{2}, \tag{9}$$

$$c_v = \frac{b_{\text{left}} + b_{\text{right}}}{2}, \tag{10}$$

$$s_u = \frac{b_{\text{bottom}} - b_{\text{top}}}{2}, \tag{11}$$

$$s_v = \frac{b_{\text{right}} - b_{\text{left}}}{2}, \tag{12}$$

$$u = c_u + s_u h_u, \tag{13}$$

$$v = c_v + s_v(h_v + 0.5), \tag{14}$$

$$z = \mu_z + h_z \sigma_z \tag{15}$$

The location of the object, h_{loc}, is then expressed as

$$h_{\text{loc}} = \text{proj}(u, v, z) \tag{16}$$

Instead of directly optimizing for h_{loc}, we optimize for h_u, h_v, and h_d. μ_z and σ_z are the mean and standard deviation of the z-coordinates of objects from the KITTI dataset. proj is the projection operator defined in [25]. We initialize h_u, h_v, and h_d to zero.

The dimensions of the object, h_{dim}, are represented by

$$h_{\text{dim}} = \mu_d + h'_{\text{dim}} \sigma_d, \tag{17}$$

Instead of directly optimizing for h_{dim}, we optimize for h'_{dim}. μ_d and σ_d are the mean and standard deviation of dimensions of objects from the KITTI dataset. We initialize h'_{dim} to zero.

Following [32], the rotation h_{rot} is defined as

$$h_{\text{rot}} = \text{arctan2}(h^{\text{sin}}_{\text{rot}}, h^{\text{cos}}_{\text{rot}}). \tag{18}$$

We initialize $h^{\text{sin}}_{\text{rot}}$ and $h^{\text{cos}}_{\text{rot}}$ by sampling from a Gaussian distribution with zero mean and a standard deviation of 0.1.

The shape and texture encoding, h_{sh} and h_{tx}, are directly optimized. We initialize them by sampling from a Gaussian distribution with zero mean and a standard deviation of 0.01.

6.2 Filtering of 2D Bounding Boxes

We filter out 2D bounding boxes with a height smaller than 20 pixels because the minimum height of objects that are used for evaluation, on the KITTI dataset, is 25 pixels. We also remove 2D bounding boxes near the image boundary because the bounding box reconstruction loss is not meaningful if the bounding box is truncated.

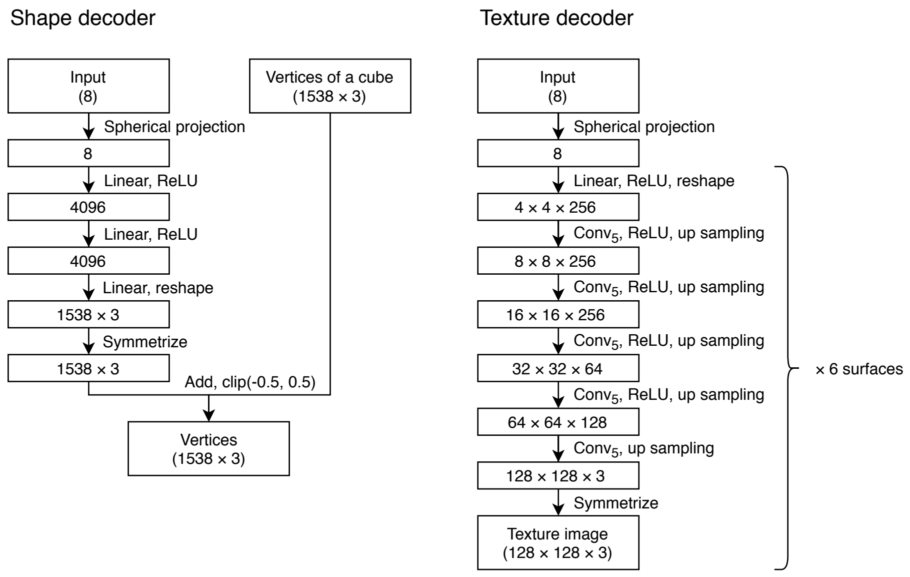


Fig. 6: We deform the vertices of a unit cube in order to generate the shape of an object. The cube is uniformly sampled to form a $17 \times 17 \times 17$ grid. The total number of vertices on the surfaces is, therefore, $17^3 - 15^3 = 1538$. Six texture images of 128×128 pixels are mapped onto the surfaces of the cube. Conv_5, in this figure, represents a 2D convolution operation that uses a 5×5 kernel.

6.3 Network Architecture for Shape Generation

We use the ResNet-18 architecture for encoding shape and texture. Fig. 6 shows the network architecture used for decoding them.

7 Additional Experimental Results

Table 5 shows a comparison with other methods on the KITTI *validation* set, using the $AP_{|R_{11}}$ metric. Table 6 shows a quantitative evaluation of our method on the KITTI *validation* set using different metrics and detection thresholds.

To confirm the effectiveness of shape reconstruction, we conducted two additional experiments. The first one uses randomly initialized object shapes and does not do any further shape optimization. The second one approximates object shapes with cuboids without shape optimization, as previously done in other works [27] that tackle the 3D detection problem.

Table 7 shows a quantitative evaluation of these approaches. When the shape is not optimized, the detection accuracy drops by about 20%-30%. When cuboids are used as shapes, the detection accuracy decreases to nearly zero. Fig. 7 shows the difference between the two approaches. If the shapes are not optimized, the (projected) object silhouettes negatively impact the estimation of rotation, when using the render-and-compare approach. As we optimize for the silhouette loss along with fitting within the bounding box area, if the shapes are cuboids, optimization results in trying to fill the area inside the bounding boxes and silhouette loss becomes ineffective. These experimental results demonstrate that optimizing shapes is essential for our method to work effectively.

Method	Supervised	3D detection			Birds eye view		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D[3]	✓	2.53	2.31	2.31	5.22	5.19	4.13
OFTNet[30]	✓	4.07	3.27	3.29	11.06	8.79	8.91
FQNet[20]	✓	5.98	5.50	4.75	9.50	8.02	7.71
ROI-10D[25]	✓	9.61	6.63	6.29	14.50	9.91	8.73
Mono3D++[11]	✓	10.60	7.90	5.70	16.70	11.50	10.10
MonoGRNet[29]	✓	13.88	10.19	7.62	-	-	-
MonoDIS [31]	✓	18.05	14.98	13.42	24.26	18.43	16.95
MonoDR		13.90	14.17	12.12	21.20	17.35	15.25

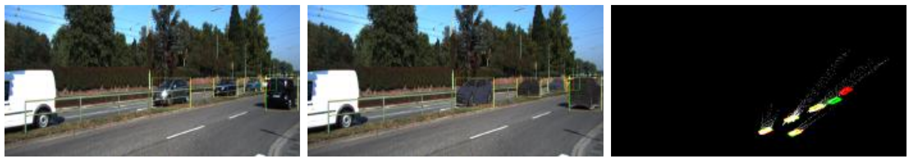
Table 5: Evaluation of different monocular 3D detection methods: We report $AP|_{R_{11}}$ on the KITTI 3D *validation* set. The values are calculated assuming an intersection-over-union (IoU) between the predicted and ground truth bounding boxes of at least 0.7.

Metric	Threshold	3D detection			Birds eye view		
		Easy	Moderate	Hard	Easy	Moderate	Hard
$AP\|_{R_{40}}$	0.7	12.50	7.34	4.98	19.49	11.51	8.72
$AP\|_{R_{40}}$	0.5	43.37	29.50	22.72	48.53	33.90	25.85
$AP\|_{R_{40}}$	0.3	65.58	52.02	42.38	68.62	54.94	45.29
$AP\|_{R_{11}}$	0.7	18.86	14.04	12.05	24.79	17.10	15.01
$AP\|_{R_{11}}$	0.5	45.76	32.31	26.19	51.13	37.29	30.20
$AP\|_{R_{11}}$	0.3	66.44	52.09	44.30	68.94	57.00	45.66

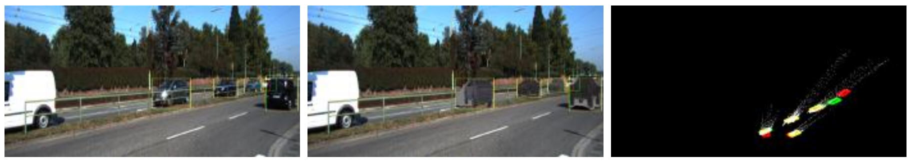
Table 6: Quantitative evaluation of different metrics and thresholds on the KITTI *validation* set.

Design choice	3D detection			Birds eye view		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Optimizing shapes	12.50	7.34	4.98	19.49	11.51	8.72
Using random shapes	9.10	5.51	4.01	16.28	10.26	7.48
Using cuboids	0.30	0.17	0.16	0.79	0.48	0.44

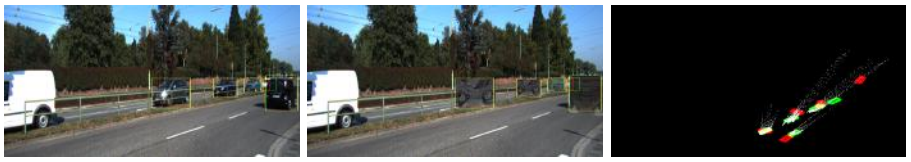
Table 7: Effectiveness of shape reconstruction on the KITTI *validation* set.



(a) 3D detection with shape optimization.



(b) 3D detection without shape optimization (the shapes are randomly initialized).



(c) 3D detection using cuboids as object shapes.

Fig. 7: The difference between 3D detection with and without shape optimization: Raw images with predicted 2D bounding boxes are shown in the left column. Reconstructed images (with the projection of an estimated 3D shape onto the image plane) are shown in the middle column. Ground-truth (green)/predicted (red) 3D bounding boxes in the bird's-eye view (BEV) are shown in the right column. The white points in BEV represent projections of object point clouds generated from predicted depth maps. The ground truth information is only used for visualization purposes.

8 Acknowledgements

This work was supported by Toyota Research Institute Advanced Development, Inc. The authors would like to thank Richard Calland, Karim Hamzaoui, Rares Ambrus, Vitor Guizilini for their helpful comments and suggestions.

References

1. Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An Information-Rich 3D Model Repository. In *CoRR*, 2015.
2. Wenzheng Chen, Jun Gao, Huan Ling, Edward J. Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer. In *NeurIPS*, 2019.
3. Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3D Object Detection for Autonomous Driving. In *CVPR*, 2016.
4. Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3D Object Proposals for Accurate Object Class Detection. In *NIPS*, 2015.
5. Francis Engelmann, Jörg Stückler, and Bastian Leibe. Joint Object Pose Estimation and Shape Reconstruction in Urban Street Scenes Using 3D Shape Priors. In *GCPR*, 2016.
6. Francis Engelmann, Jörg Stückler, and Bastian Leibe. SAMP: Shape and Motion Priors for 4D Vehicle Reconstruction. In *WACV*, 2017.
7. Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.
8. Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised Monocular Depth Estimation with Left-Right Consistency. In *CVPR*, 2017.
9. Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. PackNet-SfM: 3D Packing for Self-Supervised Monocular Depth Estimation. In *CoRR*, 2019.
10. Kaiming He, Georgia Gkioxari, Piotr Dollr, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017.
11. Tong He and Stefano Soatto. Mono3D++: Monocular 3D Vehicle Detection with Two-Scale 3D Hypotheses and Task Priors. In *ICCV*, 2019.
12. Yoshitaka Ushiku Hiroharu Kato and Tatsuya Harada. Neural 3D Mesh Renderer. In *CVPR*, 2018.
13. Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised Learning of Shape and Pose with Differentiable Point Clouds. In *NIPS*, 2018.
14. Hiroharu Kato and Tatsuya Harada. Learning View Priors for Single-view 3D Reconstruction. In *CVPR*, 2019.
15. Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2015.
16. Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical Surface Mapping via Geometric Cycle Consistency. In *ICCV*, 2019.
17. Abhijit Kundu, Yin Li, and James M. Rehg. 3D-RCNN: Instance-level 3D Object Reconstruction via Render-and-Compare. In *CVPR*, 2018.
18. Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.
19. Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014.
20. Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep Fitting Degree Scoring Network for Monocular 3D Object Detection. In *CVPR*, 2019.
21. Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. In *ICCV*, 2019.

22. Matthew M. Loper and Michael J. Black. OpenDR: An Approximate Differentiable Renderer. In *ECCV*, 2014.
23. Xinzhu Ma, Zhihui Wang, Haojie Li, Wanli Ouyang, and Pengbo Zhang. Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving. In *ICCV*, 2019.
24. Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised Learning of Depth and Egomotion from Monocular Video Using 3D Geometric Constraints. In *CVPR*, 2018.
25. Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. ROI-10D: Monocular Lifting of 2D Detection to 6D Pose and Metric Shape. In *CVPR*, 2019.
26. Hee Min Choi, Hyoa Kang, and Yoonsuk Hyun. Multi-View Reprojection Architecture for Orientation Estimation. In *ICCVW*, 2019.
27. Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *CVPR*, 2017.
28. Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon. SuperDepth: Self-Supervised, Super-Resolved Monocular Depth Estimation. In *ICRA*, 2019.
29. Zengyi Qin, Jinglu Wang, and Yan Lu. MonoGRNet: A Geometric Reasoning Network for Monocular 3D Object Localization. In *AAAI*, 2019.
30. Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic Feature Transform for Monocular 3D Object Detection. In *BMVC*, 2019.
31. Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling Monocular 3D Object Detection. In *ICCV*, 2019.
32. Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Elisa Ricci, and Peter Kontschieder. Single-Stage Monocular 3D Object Detection with Virtual Cameras. In *CoRR*, 2019.
33. David Stutz and Andreas Geiger. Learning 3D Shape Completion under Weak Supervision. In *IJCV*, 2018.
34. Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Multi-view Consistency as Supervisory Signal for Learning Shape and Pose Prediction. In *CVPR*, 2018.
35. Rui Wang, Nan Yang, Joerg Stueckler, and Daniel Cremers. DirectShape: Photometric Alignment of Shape Priors for Visual Vehicle Pose and Shape Estimation. In *ICRA*, 2020.
36. Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR from Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving. In *CVPR*, 2019.
37. Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019.
38. Bin Xu and Zhenzhong Chen. Multi-level Fusion Based 3D Object Detection from Monocular Images. In *CVPR*, 2018.
39. Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR++: Accurate Depth for 3D Object Detection in Autonomous Driving. In *ICLR*, 2020.
40. Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3D Objects with Differentiable Rendering of SDF Shape Priors. In *CVPR*, 2020.
41. Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*, 2018.
42. Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael J. Black. Three-D Safari: Learning to Estimate Zebra Pose, Shape, and Texture from Images "In the Wild". In *ICCV*, 2019.