

AutoShape: Real-Time Shape-Aware Monocular 3D Object Detection

Zongdai Liu, Dingfu Zhou*, Feixiang Lu, Jin Fang and Liangjun Zhang

Robotics and Autonomous Driving Laboratory, Baidu Research,

National Engineering Laboratory of Deep Learning Technology and Application, China

{liuzongdai, zhoudingfu, lufeixiang, fangjin, liangjunzhang}@baidu.com

Abstract

Existing deep learning-based approaches for monocular 3D object detection in autonomous driving often model the object as a rotated 3D cuboid while the object's geometric shape has been ignored. In this work, we propose an approach for incorporating the shape-aware 2D/3D constraints into the 3D detection framework. Specifically, we employ the deep neural network to learn distinguished 2D keypoints in the 2D image domain and regress their corresponding 3D coordinates in the local 3D object coordinate first. Then the 2D/3D geometric constraints are built by these correspondences for each object to boost the detection performance. For generating the ground truth of 2D/3D keypoints, an automatic model-fitting approach has been proposed by fitting the deformed 3D object model and the object mask in the 2D image. The proposed framework has been verified on the public KITTI dataset and the experimental results demonstrate that by using additional geometrical constraints the detection performance has been significantly improved as compared to the baseline method. More importantly, the proposed framework achieves state-of-the-art performance with real time. Data and code will be available at <https://github.com/zongdai/AutoShape>

1. Introduction

Perceiving 3D shapes and poses of surrounding obstacles is an essential task in autonomous driving (AD) perception systems. The accuracy and speed performance of 3D objection detection is important for the following motion planning and control modules in AD. Many 3D object detectors [50, 14] have been proposed, mainly for depth sensors such as LiDAR [35, 45] or stereo cameras [46, 18], which can provide the distance information of the environments directly. However, LiDAR sensors are expensive and stereo rigs suffer from on-line calibration issues. There-

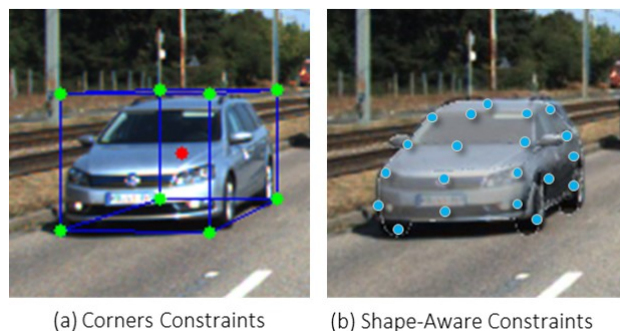


Figure 1: (a): 3D Bbox corners and center are commonly used for monocular 3D object detection. However, the rich structure information from 3D vehicle shapes and their projection on 2D images are not employed. (b) shows our shape-aware constraints constructed from an aligned 3D model. Such 2D-3D keypoints carry more semantic and geometric information and enable to construct stronger geometric constraints for monocular 3D detection.

fore, monocular camera based 3D object detection becomes a promising direction.

The main challenge for monocular-based approaches is to obtain accurate depth information. In general, depth estimation from a single image without any prior information is a challenging problem and recent many deep learning-based approaches achieve good results [7]. With the estimated depth map, pseudo LiDAR point cloud can be reconstructed via pre-calibrated intrinsic camera parameters and 3D detectors designed for LiDAR point cloud can be applied directly on pseudo LiDAR point cloud [33] [37]. Furthermore, [33] integrates the depth estimation and 3D object detection network together following an end-to-end manner. However, heavy computation burden is one main bottleneck of such two-stage approaches.

To improve the efficiency, many direct regression-based approaches have been proposed (e.g., SMOKE [24], RTM3D [19]) and achieved promising results. By representing the object as one center point, the object detection task is formulated as keypoints detection and its corresponding attributes (e.g., size, offsets, orientation, depth,

*Corresponding author

etc.) regression. With this compact representation, the computation speed of this kind of approach can reach 20~30 fps (frame per second). However, the drawback is also obvious. One center point [48, 24] representation ignores the detailed shape of the object and results in location ambiguity if its projected center point is on another object’s surface due to occlusion [47]. To alleviate this ambiguity, other geometrical constraints have been used to improve the performance. RTM3D [19] adds 8 more keypoints as additional constraints which are defined as the projected 2d location of the 3D bounding box’s corners. However, these keypoints have non-real context meanings and their 2D locations vary differently with the changing of the camera view-point, even the object’s orientation. As shown in the left of Fig. 1, some keypoints are on the ground and some are on the sky or trees. This makes the keypoints detection network extremely difficult to distinguish the keypoints or other image pixels.

In this paper, we propose a novel approach to learn the meaningful keypoints on the object surface and then use them as additional geometrical constraints for 3D object detection. Specifically, we design an automatic deformable model-fitting pipeline first to generate the 2D/3D correspondences for each object. Then, the center point plus several distinguished keypoints are learned from the deep neural network. Based on these keypoints and other regressed objects’ attributes (*e.g.*, orientation angle, object dimension *etc.*), the object’s 3D bounding box can be solved with linear equations. The proposed framework can be trained in an end-to-end manner. Our contributions include:

1. We propose a shape-aware 3d object detection framework, which employs keypoints geometry constraints for 2D/3D regression to boost the detection performance.
2. We present a method for automatically fitting the 3D shape to the visual observations and then generating ground-truth annotations of 2D/3D keypoints pairs for the training network. Our source code and dataset will be made public for the community.
3. The effectiveness of our approach has been verified on the public KITTI dataset and achieved SOTA performance. More importantly, the proposed framework achieves real-time (25 fps), which can be integrated into the AD perception module.

2. Related Work

2.1. Monocular-based 3D Detection

Image-based 3D object detection becomes popular due to the cheap price of the camera sensors. Stereo-based approaches usually suffer from calibration issues between two camera rigs. Therefore, many 3D object detection

approaches have proposed to use a single image frame. Generally, these approaches can be categorized into three types: depth-map-based, direct regression-based, and CAD model-based methods.

Depth-map-based methods [39] usually need to estimate the depth map first. In [40] and [37], the estimated depth map is transformed into point clouds, and then point-cloud-based 3D object detectors are employed for achieving the detection results. Rather than transforming the depth map into point clouds, many approaches propose using the depth estimation map directly in the framework to enhance the 3D object detection. In M3D-RPN [1] and [5], the pre-estimated depth map has been used to guide the 2D convolution, which is called as “Depth-Aware Convolution”. Direct regression-based methods are proposed to estimate the objects’ 3D information via image domain directly, such as [16, 24, 48, 47]. Direct-based methods are much more efficient than depth-map-based methods because the depth-map computation procedure is not necessary.

In order to well benefit the prior knowledge, the shape information has been integrated into the CAD-based approaches. Deep MANTA [2] and ApolloCar3D [36] are two keypoints based methods, in which the 3D keypoints are pre-defined on the CAD model and their corresponding 2D points on the image plane are computed by the deep neural network. Then the 3D pose can be solved with a standard 2D/3D pose solver [15] with these 2D/3D correspondences. Besides keypoints-based methods, dense-matching-based approaches are proposed in [12, 10, 29]. In [12], Rendering-and-Compare loss is designed for optimizing the 3d pose estimation. While in [10] and [29], the 3D pose estimation and reconstruction of each object are generated simultaneously with the deep neural network.

2.2. Data Labeling for 3D Object Detection

For easy representation, objects are usually described as 3D cuboids in deep learning frameworks while the shape information has been totally ignored. Manually label the object shape via only the image observation is extremely difficult and the annotation quality also can not be guaranteed. Many CAD model guided annotation approaches have been proposed to obtain the dense shape annotations. In [30], both the stereo image and the sparse LiDAR point cloud has been employed for generating the dense scene flow for both foreground and background pixels. For dynamic objects, 16 vehicle models are chosen as basic templates and then the dense annotation is achieved by finding an optimal 3D similarity transformation (*e.g.*, the pose and scale of the 3D model) with three types of observations such as LiDAR points, dense disparity computed by SGM [8] and labeled 2D/3D correspondences.

In [36], 66 keypoints are defined on the 3D CAD models and annotators label their corresponding 2D keypoints

on the image. Based on the 2D/3D correspondences, the object poses can be obtained via a PnP solver. In [42], the authors apply a differentiable shape renderer to signed distance fields (SDF), leveraged together with normalized object coordinate spaces (NOCS) to automatically generate the dense 3D shape without the 3D bounding boxes annotation. Although the whole process is labor-free, the annotation quality is far-from the ground truth. Different from [42], we use the ground truth 3D bounding boxes as strong guidance for our 3D shape annotation generation process.

3. Problem Definition

Before the introduction of our proposed approach, a general description of image-based 3D object detection problem is introduced first.

3.1. Pose Estimation

Given an image, the task of **pose estimation** is to estimate the orientation and translation of objects in 3D. Specifically, 6D pose is represented by a rigid transformation (\mathbf{R}, \mathbf{T}) from the *object coordinate system* to the *camera coordinate system*, where \mathbf{R} represents the 3D rotation and \mathbf{T} represents the 3D translation.

Assuming a 3D object point $\mathbf{P}_o(x_o, y_o, z_o)$ in the object coordinate system, transformed 3d point $\mathbf{P}_c(x_c, y_c, z_c)$ in camera coordinate can be obtained as

$$[x_c, y_c, z_c]^T = \mathbf{R}[x_o, y_o, z_o]^T + \mathbf{T}, \quad (1)$$

where \mathbf{R} is rotation matrix, \mathbf{T} is translation vector. Given the camera intrinsic matrix $\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$, the projected image point $\mathbf{p}(u, v)$ can be obtained as

$$s[u, v, 1]^T = \mathbf{K}[x_c, y_c, z_c]^T. \quad (2)$$

Based on Eq. 1 and Eq. 2, the object pose \mathbf{R} and \mathbf{T} can be theoretically recovered with the geometric constraints between 3D points \mathbf{P}_o on the object and the projected 2D image points \mathbf{p} .

3.2. Learning-based 3D Object Detection

In the era of deep learning, many approaches have been proposed to detect objects and directly regress their poses using neural networks, while geometric 2D/3D constraints have been ignored in the formulation. Image-based 3D object detection is a typical task, which aims at estimating the location, orientation of an object in the camera coordinate. Usually, an object is represented as a rotated 3D BBox as

$$\mathbf{r} = (r_x, r_y, r_z); \mathbf{t} = (t_x, t_y, t_z); \mathbf{d} = (l, w, h), \quad (3)$$

in which \mathbf{r} , \mathbf{t} represent the object's orientation, location in the camera coordinate and \mathbf{d} is the dimension of the object. With the super expression ability of neural networks,

all these parameters are regressed directly without imposing addition constraints. Indeed, both 3D object detection and pose estimation are essentially the same problem and (\mathbf{r}, \mathbf{t}) can be easily transformed from (\mathbf{R}, \mathbf{T}) . Therefore, we explicitly employ geometric constraints in pose estimation formulation to improve the learning-based 3D object detection.

4. Proposed Method

In this section, we propose a general deep learning-based 3D object detection framework, which can employ the 2D/3D geometric constraints. To well explore the prior knowledge, CAD models are employed here. First, we pre-define several distinguished 3D keypoints on CAD models. Then, we propose to build the correlation between these 3D keypoints and their 2D projections on the image resorting to the deep learning network. Finally, the object pose can be easily solved with these geometrical constraints. More importantly, all the processes are implemented into the neural network, which can be trained in an end-to-end manner.

4.1. Point-wise 2D-3D Constraints

Assuming a 3D point $\mathbf{P}_o^i(x_o^i, y_o^i, z_o^i)$ in object local coordinate, then its projection location (u^i, v^i) on the image plane can be obtained based on Eq. 1 and Eq. 2 as

$$s \begin{bmatrix} u^i \\ v^i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{T} \end{bmatrix} \begin{bmatrix} x_o^i \\ y_o^i \\ z_o^i \\ 1 \end{bmatrix}. \quad (4)$$

In autonomous driving scenario, the road surface that the object lies on is almost flat locally, therefore the orientation parameters are reduced from three to one by keeping only the yaw angle r_y around the Y-axis. Therefore the rotation

matrix \mathbf{R} becomes as $\begin{bmatrix} \cos(r_y) & 0 & \sin(r_y) \\ 0 & 1 & 0 \\ -\sin(r_y) & 0 & \cos(r_y) \end{bmatrix}$ and Eq. 4 can

be simplified as

$$\begin{bmatrix} -1 & 0 & \tilde{u}^i \\ 0 & -1 & \tilde{v}^i \end{bmatrix} \begin{bmatrix} T_x \\ T_z \end{bmatrix} = \begin{bmatrix} x_o^i \cos(r_y) + z_o^i \sin(r_y) + \tilde{u}^i [x_o^i \sin(r_y) - z_o^i \cos(r_y)] \\ y_o^i + \tilde{v}^i [x_o^i \sin(r_y) - z_o^i \cos(r_y)] \end{bmatrix}, \quad (5)$$

where $\tilde{u}^i = (u^i - c_x)/f_x$, $\tilde{v}^i = (v^i - c_y)/f_y$ and $\mathbf{T} = [T_x, T_y, T_z]^T$. As described in Eq. 5, for each object point, two constraints are given. If n points are provided, $n \times 2$ constraints can be obtained as

$$\mathbf{A}\mathbf{T} = \mathbf{B}, \quad (6)$$

where

$$\mathbf{A} = \begin{bmatrix} -1 & 0 & \tilde{u}_1 \\ 0 & -1 & \tilde{v}_1 \\ \vdots & \vdots & \vdots \\ -1 & 0 & \tilde{u}_n \\ 0 & -1 & \tilde{v}_n \end{bmatrix}_{2n \times 3},$$

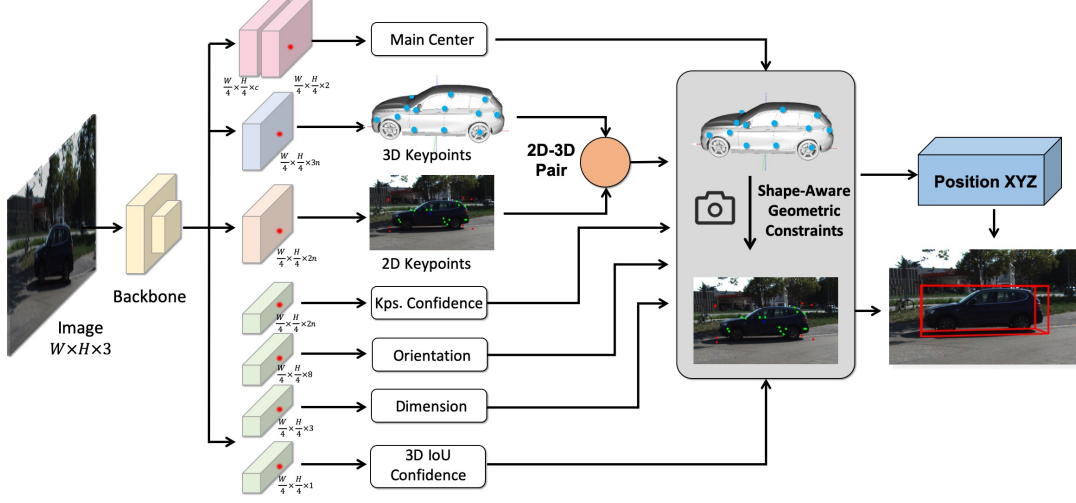


Figure 2: Overview of the proposed keypoints-based 3D detection framework. By passing the backbone network, 8 branch heads are followed for center point classification, center point offset, 2D keypoints, 3D coordinates, keypoints confidence, object orientation, dimension, and 3D detection score regression purpose. Finally, all the regressed information has been employed for recovering the object’s 3D Bbox in the camera coordinate.

and

$$\mathbf{B} = \begin{bmatrix} x_o^1 \cos(r_y) + z_o^1 \sin(r_y) + \tilde{u}^1(x_o^1 \sin(r_y) - z_o^1 \cos(r_y)) \\ y_o^1 + \tilde{v}^1(x_o^1 \sin(r_y) - z_o^1 \cos(r_y)) \\ \vdots \\ x_o^n \cos(r_y) + z_o^n \sin(r_y) + \tilde{u}^n(x_o^n \sin(r_y) - z_o^n \cos(r_y)) \\ y_o^n + \tilde{v}^n(x_o^n \sin(r_y) - z_o^n \cos(r_y)) \end{bmatrix}_{2n \times 1}$$

However, in the real AD scenario, not all the keypoints can be seen from a certain camera viewpoint. For these keypoints which have been seriously occluded, the 2D/3D keypoints regression can not be well guaranteed. To well handle this kind of uncertainty, we propose to output an additional score to measure the confidence of each keypoint. And this score can be used as a weight during the pose calculation process. Specifically, for the $2n$ constrains in Eq. 6, an additional weights $c = \{c_1, c_2, \dots, c_{2n}\}$ have been added to determine their importance during the pose calculation procedure. Therefore, Eq. 6 can be reformulated as

$$\text{diag}(c)\mathbf{A}\mathbf{T} = \text{diag}(c)\mathbf{B} \quad (7)$$

In this linear system, \mathbf{T} represents the object location in the camera coordinate system, which can be solved by providing 2D/3D correspondences and the rotation angle r_y . Here, the 3D keypoints are defined in the local object’s coordinate varying in a relatively small range and the 2D keypoints are defined in the image domain. Both of them are easy for networks to learn. However, manual labeling of the ground truth for 2D and 3D keypoints is very costly and tedious. Therefore, we develop an auto-labeling pipeline by optimizing the 2D and 3D reprojection errors. The detailed annotation pipeline will be introduced in Section 5.

4.2. Network

An overview of our proposed framework is illustrated in Fig. 2. Here, we follow one-stage-based 3D object detection framework such as CenterNet [48] for its inference efficiency. Our proposed framework is backbone independent and here we employ DLA-34 [41] in our implementation. Given an image \mathbf{I} with width W and height H , the output feature map will be 4 times smaller than \mathbf{I} after passing through the backbone network. To well utilize the geometric constraints, the following information is required to be learned from the deep neural network.

Object Center: in anchor-free based object detection frameworks, the object center is essential information, which serves two functions: one is whether there is an object and the other is that if there exists an object, where is the center. Usually, these two functions are realized by a classification branch by distinguishing a pixel whether is an object center or not. The output of this branch will be $\frac{W}{4} \times \frac{H}{4} \times C$, where C is the number of classes.

Besides the classification, an additional “offset” regression branch is required to compensate for the quantization error during the down-sampling process. The output of this branch is $\frac{W}{4} \times \frac{H}{4} \times 2$ to represent the offset in x and y direction respectively.

Object Dimension: a separate branch is used to regress the object dimension h, w, l with the output size of $\frac{W}{4} \times \frac{H}{4} \times 3$. Similar to other approaches, we don’t regress the absolute object’s size directly and regress a relative scale compared to the mean object size of each class. Details operation can be found in [48].

2D Keypoints: rather than directly detect these key-

points from the image, we regress n ordered 2D offset coordinates for each object center. The benefit is that the number and order of keypoints for each object can be well guaranteed. In addition, the regression for the offset is easier by removing the object center. The output size of this branch is $\frac{W}{4} \times \frac{H}{4} \times 2n$.

3D Keypoints: similar to the 2D keypoints, we regress the 3d keypoints in the local object coordinate. In addition, all 3D keypoints values are normalized by object dimension (l, w, h) in x, y, z -direction respectively. By using this format, the 3D keypoints values are in a relatively small range, which will benefit the whole regression process. The output size of this branch is $\frac{W}{4} \times \frac{H}{4} \times 3n$.

Object Orientation: similarly, we regress local orientation angle with respect to the ray through the perspective point of 3D center following Multi-Bin based method [31]. Here, 8 bins are used with the output size of $\frac{W}{4} \times \frac{W}{4} \times 8$.

Keypoints Confidence Scores: for each keypoint, a couple of additional confidence scores have been regressed for measuring its contribution in the linear system for solving the object pose. For $2n$ constrains in Eq. 6, a feature map with size of $\frac{W}{4} \times \frac{W}{4} \times 2n$ will be outputted.

3D IoU Confidence Score: rather using the classification score directly as the object detection confidence, we add one branch to regress the 3D IoU score in purpose. This score is supervised by the IoU between estimated Bbox and ground truth Bbox. Finally, the product of this score and the output classification score is assigned as the final 3D detection confidence score.

4.3. Loss Function

The overall loss contains the following items: a center point classification loss l_m and center point offset regression loss l_{off} , a 2D keypoints regression loss l_{2D} , a 3D keypoints points regression loss l_{3D} , an orientation multi-bin loss l_r , a dimension regression loss l_D , a 3D IoU confidence loss l_c and a 3D bounding box IoU loss l_{IoU} . Specifically, the multi-task loss is defined as

$$\mathbf{L} = w_m l_m + w_{off} l_{off} + w_{2D} l_{2D} + w_{3D} l_{3D} + w_r l_r + w_D l_D + w_c l_c + w_{IoU} l_{IoU} \quad (8)$$

where l_m is the focal loss as used in [48], l_{2D} is a depth-guided $l-1$ loss as used in [17], l_D and l_{3D} are L1 loss with respect to the ground truth. Orientation loss l_r is the Multi-Bin loss. 3D IoU confidence l_c is a binary cross-entropy loss supervised by the IoU between the predicted 3D BBox and ground truth. l_{IoU} is the IoU loss between the predicted 3D BBox and ground truth [44].

5. 3D Shape Auto-Labeling

In this section, we will introduce how to automatically fit the 3D shape to the visual observations and then automatically generate ground-truth annotations of 2D keypoints and

3D locations in the local object coordinate for training the network. The main process is illustrated in 3. Different from existing methods which use only a few CAD models for 3D labeling (*e.g.*, 11 in [42] and 16 in [30]), we adopt a 3D deformable vehicle template [36, 25] that can represent arbitrary vehicle shape by adjusting the parameters. Therefore, the 3D shape labeling process can be formulated as an optimization problem that aims at computing the optimal parameter combination to fit the visual observations (*i.e.* 2D instance mask, 3D bounding box, and 3D LiDAR points).

5.1. Deformable Vehicle Template

In the real-world traffic scenarios, there are many different vehicle types (*e.g.*, coupe, hatchback, notchback, SUV, MPV, *etc.*) and their geometric shapes vary significantly. To perform 3D shape fitting, a straight-forward solution is to build a 3D shape dataset and the fitting process can be regarded as model retrieval. However, dataset construction is labor-intensive, inefficient, and costly. Instead, we use a deformable 3D model for vehicle representation [25]. Specifically, this template is composed of a set of PCA (Principal Components Analysis) basis r . Any new 3D vehicle $\mathcal{M}(s)$ can be represented as a mean shape model \mathcal{M}_0 plus a linear combination of r principal components with coefficient $s = [s_1, s_2, \dots, s_r]$ as

$$\mathcal{M}(s) = \mathcal{M}_0 + \sum_{k=1}^r s_k \delta_k p_k, \quad (9)$$

where p_k and δ_k are the principal component direction and corresponding standard deviation and s_k is the coefficient of the k_{th} principal component. Based on the vehicle template, we can automatically fit an optimal 3D shape to the visual observations (details in Subsec. 5.2).

5.2. 3D Shape Optimization

For each vehicle, our goal is to assign a proper 3D shape to fit the visual observations, including the 2D instance mask, the 3D bounding box, and the 3D LiDAR points. Specifically, the annotations of 2D instance mask I_{ins} and the 3D bounding box B_{box} are provided by the KINS dataset [32] and the KITTI dataset [6], respectively. The annotation of 3D LiDAR points for each vehicle is much more complex. According to the labeled 3D bounding boxes, we first segment out the individual 3D points from the entire raw point cloud. Then we remove the ground points using the ground-plane estimation method (*i.e.* RANSAC-based plane fitting). Finally, we obtain the “clean” 3D points for each vehicle, which is represented as $\mathbf{p} = \{p_0, \dots, p_k\}$.

The 3D shape annotation is to compute the best PCA coefficient \hat{s} and the object 6-DoF pose $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$. Existing 3D object detection benchmarks (*e.g.*, KITTI, Waymo) only label the yaw angle because they assume vehicles are on the

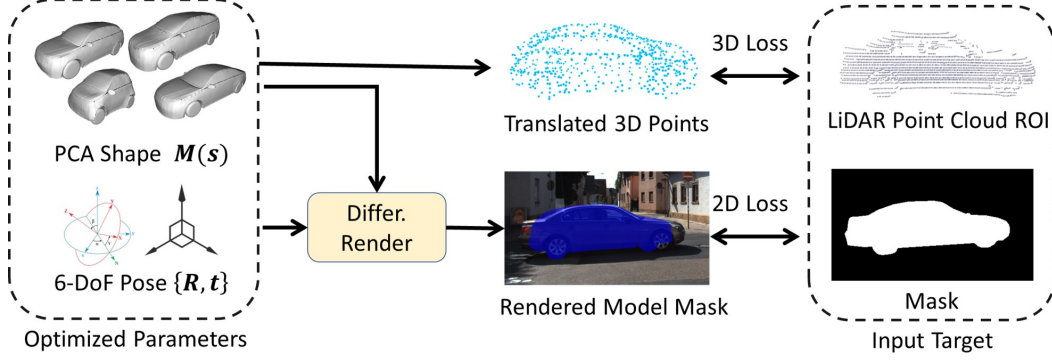


Figure 3: The pipeline of the proposed 3D shape-aware auto-labeling framework. By providing different kinds of vehicle CAD models, a mean shape template and r principal basic can be obtained. By giving a 3D object sample in the KITTI dataset, the optimal principal components with coefficient s and 6-DoF pose can be iteratively optimized by minimizing the 2D and 3D losses which are defined on the scanned sparse point cloud and the instance mask.

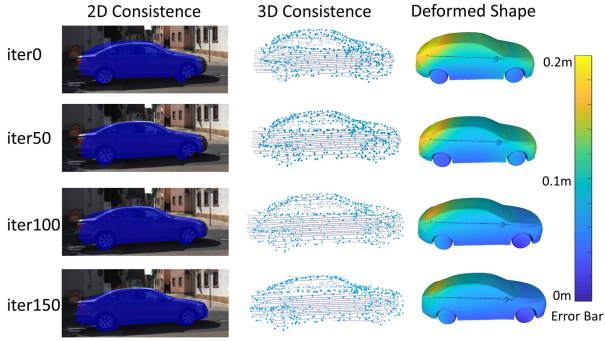


Figure 4: Illustration of the 3D shape optimization process from step 0 to step 200. From the top to bottom, the rendered mask gradually covers the target mask and 3D model vertices align with the point cloud gradually and the 3D deformed shape changes from a mean shape to a ‘notchback’.

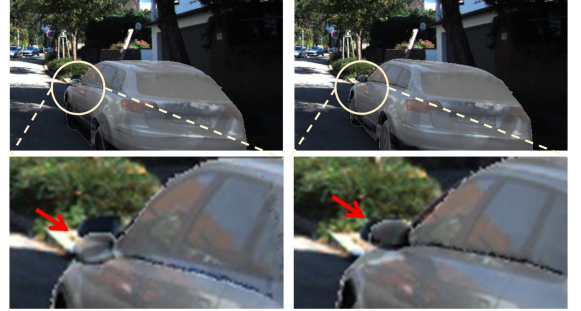
road plane. However, we experimentally (an example has been given in Fig. 5) find that the other two angles (*i.e.* pitch and roll) can significantly improve the 3D shape annotation results. Therefore, the loss function is formulated as

$$\hat{s}, \hat{\mathbf{R}}, \hat{\mathbf{t}} = \arg \min_{s, \mathbf{R}, \mathbf{t}} \{ \alpha \mathbf{L}_{2D} (Pr(\mathcal{M}(s), \mathbf{R}, \mathbf{t}), I_{ins}) + \beta \mathbf{L}_{3D}(v(s), \mathbf{R}, \mathbf{t}, \mathbf{p}) \}, \quad (10)$$

which consists of 3D points loss \mathbf{L}_{3D} and 2D instance loss \mathbf{L}_{2D} . α, β are two hype-parameters to balance these two constraints.

The operation $Pr(\cdot)$ is a differentiable rendering function to produce the binary mask \tilde{I}_{ins} of $\mathcal{M}(s)$ with translation $\{\mathbf{R}, \mathbf{t}\}$. Specifically, the \mathbf{L}_{2D} is defined as the sum distance over each pixel (i, j) in image I

$$\mathbf{L}_{2D} = \sum_{(i,j) \in I} \left\| \tilde{I}_{ins}(i, j) - I_{ins}(i, j) \right\|. \quad (11)$$



(a) Only *yaw* optimized (b) *yaw, roll, pitch* optimized

Figure 5: An example of CAD model-fitting results with one angle (*yaw*) only or three angles (*yaw, roll, and pitch*). When the road is not flat, serious misalignment will happen if only *yaw* angle is applied for optimization.

For each 3D point $p_i \in \mathbf{p}$, we find its nearest neighbor v_i in $v(\tilde{s})$ and compute the distance between them. \mathbf{L}_{3D} is defined as the sum distance over all correspondences pairs

$$\mathbf{L}_{3D} = \sum_{p_i \in \mathbf{p}} \|p_i - v(\tilde{s})_i\|. \quad (12)$$

The function of Eq. 10 can be optimized by the gradient descent strategy. The vehicle’s center position and orientation are used for \mathbf{t} and *yaw* angle initialization, while the pitch, roll angles, and PCA coefficients are initialized as zeros. Then we forward the pipeline and compute \mathbf{L}_{2D} and \mathbf{L}_{3D} and finally back-propagate gradients to update $s, \mathbf{R}, \mathbf{t}$. In Fig. 4, we depict the intermediate result during the optimization process.

6. Experimental Results

We implement the approach and evaluate it on the public KITTI [6] 3D object detection benchmark.

Methods	Modality	AP _{3D} 70 (%)			AP _{BEV} 70 (%)			Time (s)
		Moderate	Easy	Hard	Moderate	Easy	Hard	
M3D-RPN [1]	Mono	9.71	14.76	7.42	13.67	21.02	10.23	0.16
SMOKE [24]	Mono	9.76	14.03	7.84	14.49	20.83	12.75	0.03
MonoPair [4]	Mono	9.99	13.04	8.65	14.83	19.28	12.89	0.06
RTM3D [19]	Mono	10.34	14.41	8.77	14.20	19.17	11.99	0.05
AM3D [27]	Mono*	10.74	16.50	9.52	17.32	25.03	14.91	0.4
PatchNet [26]	Mono*	11.12	15.68	10.17	16.86	22.97	14.97	0.4
RefinedMPL [37]	Mono*	11.14	18.09	8.94	17.60	28.08	13.95	0.15
KM3D [17]	Mono	11.45	16.73	9.92	16.20	23.44	14.47	0.03
D4LCN [5]	Mono*	11.72	16.65	9.51	16.02	22.51	12.55	0.20
JAFA [47]	Mono	12.01	17.81	10.61	17.88	25.88	15.35	0.03
YOLOMono3D [22]	Mono	12.06	18.28	8.42	17.15	26.79	12.56	0.05
Monodet [28]	Mono	12.26	17.23	10.29	18.89	24.79	16.00	0.04
MonoRUn [3]	Mono	12.30	19.65	10.58	17.34	27.94	15.24	0.07
GrooMeD-NMS [11]	Mono	12.32	18.10	9.65	18.27	26.19	14.05	0.12
DDMP-3D [38]	Mono	12.78	19.71	9.80	17.89	28.08	13.44	0.18
Ground-Aware [23]	Mono	13.25	21.65	9.91	17.98	29.81	13.08	0.05
CaDDN [34]	Mono*	13.41	19.17	11.46	18.91	27.94	17.19	0.63
MonoEF [49]	Mono	13.87	21.29	11.71	19.70	29.03	17.26	0.03
MonoFlex [43]	Mono	13.89	19.94	12.07	19.75	28.23	16.89	0.03
Baseline Method [17]	Mono	11.45	16.73	9.92	16.20	23.44	14.47	0.03
AutoShape-16kps	Mono†	13.72	21.75	10.96	19.00	30.43	15.57	0.04
AutoShape-48kps	Mono†	14.17	22.47	11.36	20.08	30.66	15.59	0.05
Improvements	-	+2.72	+5.74	+1.44	+3.88	+7.22	+1.12	-

Table 1: Comparison with other public methods on the KITTI testing server for 3D “Car” detection. For the “direct” methods, we represent the “Modality” with “Mono” only. We use * to indicate that the “depth” has been used by these methods during training and inference procedure. † indicates that ‘CAD models’ have been used in data labeling stage. For easy understanding, we have highlighted the top numbers in red for each column and the second best is shown in blue.

6.1. Dataset and Implementation Details

Dataset: the KITTI dataset is collected from the real traffic environment from the Europe streets. The whole dataset has been divided into training and test two subsets, which consist of 7, 481 and 7, 518 frames, respectively. Since the ground truth for the test set is not available, we divide the training data into a *train* set and a *val* set as in [50], and obtain 3, 712 data samples for training and 3, 769 data samples for validation to refine our model. On the KITTI benchmark, the objects have been categorized into “Easy”, “Moderate”, and “Hard” based on their height in the image and occlusion ratio, *etc.*

Evaluation Metric: we focus on the evaluation on “Car” category because it has been considered most in the previous approaches. For evaluation, the average precision (AP) with Intersection over Union (IoU) is used as the metric for evaluation. Our AutoShape approach is compared with existing methods on the *test* set using $AP_{R_{40}}$ by training our model on the whole 7, 481 images. We evaluate on the *val* set for ablation by training our model on the *train* set using $AP_{R_{40}}$.

Implementation Details: We implement our auto-labeling approach (Sec. 5) using differentiable renderer [9, 20, 21] which is optimized by using the Adam optimizer with a learning rate of 0.002. To speed the optimization and save memory, we downsample the PCA model to 666 vertices and 998 faces. We set α and β to 1.0 and 5.0, respectively. Our shape-aware 3D detection network uses DLA-34 [41] as backbone. We pad the image size to 1280×384 . 3D IoU confidence loss weight w_c and 3D IoU loss weight

w_{iou} are increased from 0 to 1 with exponential RAMP-UP strategy [13]. We use Adam optimizer with a base learning rate of 0.0001 for 200 epochs and reduce by $10\times$ at 100 and 160 epochs. We project the ground truth to corresponding right image and use random scaling (between 0.6 to 1.4), random shifting in the image range, and color jittering for data augmentation. The network is trained on 2 NVIDIA Tesla V100 (16G) GPU cards and the batch size is set to 16. For the KITTI *test* set evaluation, we sample 16/48 key-points from the 3D shape, 8 corner points, and 1 center to train network.

6.2. Data Auto-Labeling Evaluation

Our approach can automatically generate the 2D key-points and their corresponding 3D locations in the local object coordinate which are employed as the supervision signal during the training process. To verify the quality of the labeling results, the 2d instance segmentation mean AP and 3D bounding box mean AP is used here for verification. Specifically, the 2D instance segmentation IoU is calculated using the projected mask by 3D models and the ground truth mask (from KINS [32]). We obtain the labeled 3D bounding box using the dimension of the 3D model with the optimized 6-DoF pose, which is compared to the ground-truth 3D bounding box provided by [6] with the mean IoU score. Tab. 2 shows the detailed comparison results. The proposed method can achieve 0.86 for 2D mean AP and 0.76 for 3D mean AP, which justifies the effectiveness of our auto-labeling approach.

6.3. Evaluation for 3D Object Detection

The evaluation of the proposed approach with other SOTA methods for 3D detection on KITTI [6] *test* set are given in Tab. 1. From the table, we can obviously find that the proposed method with 48 keypoints achieves 4 first places in 6 tasks with the $AP|_{R_{40}}$ metric. We also report our method with 16 keypoints, which has faster inference time and keeps promising accuracy. In addition, most of the existing methods such as [27, 26, 37, 5, 34], need to estimate the depth map, resulting in a heavy computation burden in inference. In contrast, our method obtains the depth information by 3D shape-aware geometric constraints, which is more accurate with faster running speed. We achieve 25 FPS with an NVIDIA V100 GPU card with 16 keypoints configuration. Compared with baseline geometric constraint methods [17, 19] using 8 corners and 1 center point as keypoints for training, our method with 48 keypoints utilizes more shape-aware keypoints to construct stronger geometric constraints, getting +5.74%, +2.72%, 1.44%, +7.22%, +3.88%, +1.12% improvements for AP_{3D} and AP_{BEV} on “Easy”, “Moderate”, and “Hard” categories.

6.4. Qualitative Results

Qualitative results of 3D shape auto-labeling are shown in Fig 6. Each vehicle in the image is overlaid with a rendered 3D model optimized by our method. We can see the consistency of our labeled shape and the real object. We also visualize some representative results of our shape-aware model in Fig 7. Our model can predict object location accurately even for distant and truncated objects.



Figure 6: Qualitative results of our 3D shape auto-labeling.

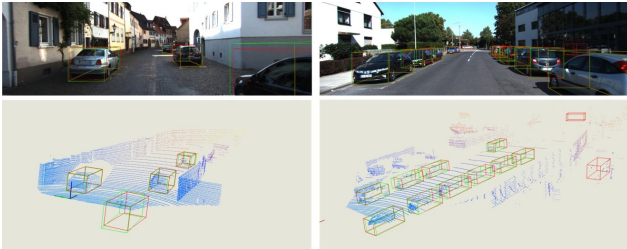


Figure 7: Qualitative 3d detection results on KITTI validation set. Red boxes represent our predictions, and green boxes come from ground truth. LiDAR signals are only used for visualization.

6.5. Ablation Studies

The Number of Keypoints: our shape-aware 3D detection network benefits from the geometric constraint of 2D-3D keypoints from the 3D shape. To better understand the effect of different numbers of the keypoints, we set it from 0 to 48 with an interval of 8. Note that the 8 corners and 1 center point are always maintained in this experiment and we vary the extra keypoints. As shown in Fig. 8, from 0 to 16, the network performance is significantly improved. From 16 to 48, however, we observe that the network performance is not sensitive to the number of the keypoints. The main reason is that more dense 2D shape points can be overlapped in the $\frac{W}{4} \times \frac{H}{4}$ heatmap during the regression process. Furthermore, with more keypoints, the network consumes more GPU memory for storage and computation, resulting in longer training and inference time. In practice, we set the number of extra keypoints to 16, which is a good compromise of accuracy and efficiency.

2D/3D Loss for Auto-Labeling: our auto-labeling approach (Sec. 5) can generate precise posed 3D shape for each 2D vehicle instance. The key technique is simultaneously optimizing the 2D/3D constraints (loss) for better matching. Here, we conduct an ablation study to justify

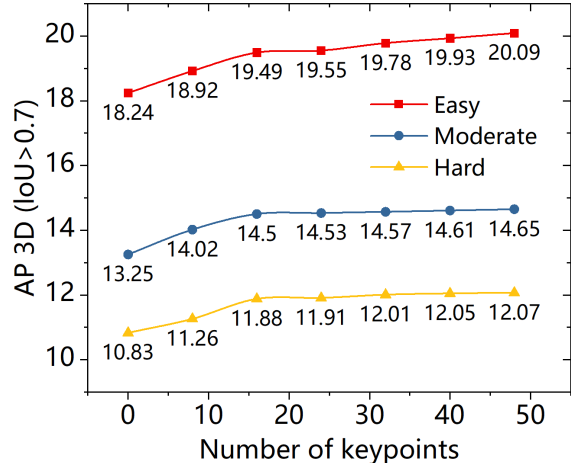


Figure 8: 3D object detection performance with different number of keypoints on KITTI val set using $AP|R_{40}$.

the effectiveness of the 2D/3D loss. We first only use 3D point loss L_{3D} in the objective function. Then we only use the 2D mask loss L_{2D} for optimization. Finally, we take both 2D/3D loss into computation. Tab. 2 shows that using both 2D/3D loss get the best performance in the auto-labeling process. We further observe that the impact of 2D mask loss L_{2D} is more important than 3D point loss. By using both L_{2D} and L_{3D} , the labeling accuracy is improved to 86.35 and 76.92, resulting in better 3D detection performance. This correlation indicates that 3D detection performance can be significantly improved by using high-quality labeling data of 3D shapes.

L_{2D}	L_{3D}	Label Acc.		Car 3D Det.		
		I_{2D}	I_{3D}	Easy	Mod.	Hard
	✓	0.61	0.71	15.49	11.35	9.34
✓		0.82	0.74	18.36	13.88	11.23
✓	✓	0.86	0.76	20.09	14.65	12.07

Table 2: Shape Autolabeling Ablation Experiments on KITTI val set using $AP|R_{40}$.

7. Conclusion

In this paper, we present a framework for real-time monocular 3D object detection by explicitly employing shape-aware geometric constraints between 3D keypoints and their 2D projections on images. Both the 3D keypoints and 2D project points are learned from deep neural networks. We further design an automatic annotation pipeline for labeling object 3D shape, which can automatically generate the shape-aware 2D/3D keypoints correspondences for each object. Experimental results show our approach can achieve state-of-the-art detection accuracy with real-time performance. Our approach is general for other types of vehicles, and in the future, we are interested in validating the performances of our approach on other objects.

References

- [1] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9287–9296, 2019. 2, 7, 11
- [2] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teulière, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2040–2049, 2017. 2
- [3] Hansheng Chen, Yuyao Huang, Wei Tian, Zhong Gao, and Lu Xiong. Monorun: Monocular 3d object detection by reconstruction and uncertainty propagation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10379–10388, 2021. 7
- [4] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang Li. Monopair: Monocular 3d object detection using pairwise spatial relationships. *arXiv preprint arXiv:2003.00504*, 2020. 7, 11
- [5] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1000–1001, 2020. 2, 7
- [6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 5, 6, 7
- [7] Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 3828–3838, 2019. 1
- [8] Heiko Hirschmüller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 807–814. IEEE, 2005. 2
- [9] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018. 7
- [10] Jason Ku, Alex D Pon, and Steven L Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11867–11876, 2019. 2
- [11] Abhinav Kumar, Garrick Brazil, and Xiaoming Liu. Groomed-nms: Grouped mathematically differentiable nms for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8973–8983, 2021. 7
- [12] Abhijit Kundu, Yin Li, and James M Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3559–3568, 2018. 2
- [13] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 7
- [14] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1
- [15] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnnp: An accurate o(n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009. 2
- [16] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1019–1028, 2019. 2
- [17] Peixuan Li. Monocular 3d detection with geometric constraints embedding and semi-supervised training, 2020. 5, 7
- [18] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo r-cnn based 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7644–7652, 2019. 1
- [19] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. *arXiv preprint arXiv:2001.03343*, 2020. 1, 2, 7
- [20] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019. 7
- [21] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. A general differentiable mesh renderer for image-based 3d reasoning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 7
- [22] Yuxuan Liu, Lujia Wang, and Liu Ming. Yolostereo3d: A step back to 2d for efficient stereo 3d detection. In *2021 International Conference on Robotics and Automation (ICRA)*. IEEE, 2021. 7
- [23] Yuxuan Liu, Yuan Yixuan, and Ming Liu. Ground-aware monocular 3d object detection for autonomous driving. *IEEE Robotics and Automation Letters*, 6(2):919–926, 2021. 7
- [24] Zechen Liu, Zizhang Wu, and Roland Tóth. Smoke: single-stage monocular 3d object detection via keypoint estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 996–997, 2020. 1, 2, 7
- [25] Feixiang Lu, Zongdai Liu, Xibin Song, Dingfu Zhou, Wei Li, Hui Miao, Miao Liao, Liangjun Zhang, Bin Zhou, Ruigang Yang, et al. Permo: Perceiving more at once from a single image for autonomous driving. *arXiv e-prints*, pages arXiv–2007, 2020. 5
- [26] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking pseudo-lidar representation. In *European Conference on Computer Vision*, pages 311–327. Springer, 2020. 7

- [27] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, and Xin Fan. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6851–6860, 2019. 7
- [28] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4721–4730, 2021. 7
- [29] Fabian Manhardt, Wadim Kehl, and Adrien Gaidon. Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2069–2078, 2019. 2
- [30] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3061–3070, 2015. 2, 5
- [31] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017. 5
- [32] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2019. 5, 7
- [33] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. *arXiv preprint arXiv:2004.03080*, 2020. 1
- [34] Cody Reading, Ali Harakeh, Julia Chae, and Steven L Waslander. Categorical depth distribution network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8555–8564, 2021. 7
- [35] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointcnn: 3d object proposal generation and detection from point cloud. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 1
- [36] Xibin Song, Peng Wang, Dingfu Zhou, Rui Zhu, Chenye Guan, Yuchao Dai, Hao Su, Hongdong Li, and Ruigang Yang. Apollo3d: A large 3d car instance understanding benchmark for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5452–5462, 2019. 2, 5
- [37] Jean Marie Uwabeza Vianney, Shubhra Aich, and Bingbing Liu. Refinedmpl: Refined monocular pseudolidar for 3d object detection in autonomous driving. *arXiv preprint arXiv:1911.09712*, 2019. 1, 2, 7
- [38] Li Wang, Liang Du, Xiaoqing Ye, Yanwei Fu, Guodong Guo, Xiangyang Xue, Jianfeng Feng, and Li Zhang. Depth-conditioned dynamic message propagation for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 454–463, 2021. 7
- [39] Xinlong Wang, Wei Yin, Tao Kong, Yuning Jiang, Lei Li, and Chunhua Shen. Task-aware monocular depth estimation for 3d object detection. *arXiv preprint arXiv:1909.07701*, 2019. 2
- [40] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 0–0, 2019. 2
- [41] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018. 4, 7
- [42] Sergey Zakharov, Wadim Kehl, Arjun Bhargava, and Adrien Gaidon. Autolabeling 3d objects with differentiable rendering of sdf shape priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12224–12233, 2020. 3, 5
- [43] Yunpeng Zhang, Jiwen Lu, and Jie Zhou. Objects are different: Flexible monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3289–3298, 2021. 7, 11
- [44] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019. 5
- [45] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous driving. In *the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1839–1849, 2020. 1
- [46] Dingfu Zhou, Vincent Frémont, Benjamin Quost, and Bihao Wang. On modeling ego-motion uncertainty for moving object detection from a mobile platform. In *IEEE Intelligent Vehicles Symposium Proceedings*, pages 1332–1338, 2014. 1
- [47] Dingfu Zhou, Xibin Song, Yuchao Dai, Junbo Yin, Feixiang Lu, Miao Liao, Jin Fang, and Liangjun Zhang. Iafa: Instance-aware feature aggregation for 3d object detection from a single image. In *Proceedings of the Asian Conference on Computer Vision*, 2020. 2, 7
- [48] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2, 4, 5
- [49] Yunsong Zhou, Yuan He, Hongzi Zhu, Cheng Wang, Hongyang Li, and Qinhong Jiang. Monocular 3d object detection: An extrinsic parameter free approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7556–7566, 2021. 7
- [50] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 1, 7

8. Supplemental Material

8.1. Ablation Study for Keypoints Confidence Regression

The 2D/3D keypoints regression is a critical component in the proposed framework, however, inaccurate regression of these keypoints is inevitable in the real AD scenario due to many reasons e.g., viewpoint change, occlusion, and labeling noise, etc. Especially, these prediction outliers will greatly affect the results of the linear system described in Eq. 6. In order to handle this problem, we propose to predict a confidence score for each keypoint and employ it as a weight for determining its contribution to the linear system. To verify the effectiveness of the prediction confidence, we set a series of ablations studies on the “Car” category.

We give the results in Tab. 3. From this table, we can see that the 3D object detection performance can be significantly improved by integrating the regressed key points confidences. More importantly, this improvement is independent of the number of keypoints. In addition, for further understanding the actual meaning of this predicted confidence, we have visualized them in Fig. 9. Interestingly, we find that these keypoints with high confidences usually come from the ground point (the intersection point between the tire and the ground) and these distinguished shape border points. These points will give more contribution to the object pose estimation.



Figure 9: Visualization of keypoints confidences. Here, the blue represents score “1” and yellow represents score “0” and the color changing from blue to yellow represents the confidence score decreasing from “1” to “0”. This figure is better to view in color print.

Num. Kps.	Kps. Confi.	Car 3D Det.		
		Easy	Mod.	Hard
16		16.49	12.31	10.54
	✓	19.59	14.50	11.88
48		16.85	12.39	10.04
	✓	20.09	14.65	12.07

Table 3: Keypoints confidence ablation experiments on KITTI *val* set using $AP|_{R40}$ metric.

8.2. Multi-classes Detection

Currently, the designed Autoshape model can’t generate the keypoints annotation for “Pedestrian” and “Cyclist” due to the lack of CAD models. Here, we simply transform the 3D keypoints from the mean “Car” template to the “Pedestrian” and “Cyclist” by normalize them first and re-scale them to the bounding box’s size of other categories. By generating these keypoints, then the object’s pose can easily solve as the “Car” category. We evaluate multi-class 3d detection on the KITTI *test* sever and the performances are shown in Tab. 4. From this table, we can find that the proposed framework performs relatively well even though the keypoints annotation is not very accurate of “Pedestrian” and “Cyclist”. Interestingly, we find that the cyclist gives much better results than the “Pedestrian” and this is because the “Cyclist” can be considered as a rigid object to some extent. On the contrary, the “Pedestrian” is a non-rigid object and the location of these keypoints varies a lot with different object pose.

Methods	3D Det.					
	Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard
M3D-RPN[1]	4.92	3.48	2.94	0.94	0.65	0.47
MonoPair[4]	10.02	6.68	5.53	3.79	2.12	1.83
MonoFlex[43]	9.43	6.31	5.26	4.17	2.35	2.04
Ours	5.46	3.74	3.03	5.99	3.06	2.70

Table 4: Quantitative results for “Pedestrian” and “Cyclist” on KITTI *test* set with $AP|_{R40}$ metric.