# Distance-Normalized Unified Representation for Monocular 3D Object Detection

Xuepeng Shi[1], Zhixiang Chen[1], and Tae-Kyun Kim[1,2]

[1] Imperial College London
[2] Korea Advanced Institute of Science and Technology
x.shi19, zhixiang.chen, tk.kim@imperial.ac.uk

**Abstract.** Monocular 3D object detection plays an important role in autonomous driving and still remains challenging. To achieve fast and accurate monocular 3D object detection, we introduce a single-stage and multi-scale framework to learn a unified representation for objects within different distance ranges, termed as UR3D. UR3D formulates different tasks of detection by exploiting the scale information, to reduce model capacity requirement and achieve accurate monocular 3D object detection. Besides, distance estimation is enhanced by a distance-guided NMS, which automatically selects candidate boxes with better distance estimates. In addition, an efficient fully convolutional cascaded point regression method is proposed to infer accurate locations of the projected 2D corners and centers of 3D boxes, which can be used to recover object physical size and orientation by a projection-consistency loss. Experimental results on the challenging KITTI autonomous driving dataset show that UR3D achieves accurate monocular 3D object detection with a compact architecture.

**Keywords:** Monocular 3D Object Detection, Unified Representation Across Different Distance Ranges, Distance-Guided NMS, Fully Convolutional Cascaded Point Regression.

## 1 Introduction

Object detection is a fundamental and challenging problem in computer vision [25]. In the past years, with the emergence of deep learning [18, 11] and the availability of large-scale annotated datasets [6, 24], the state of the art in 2D object detection has improved significantly [10, 34, 27, 23, 4, 40]. Object detection in the 2D image plane, however, is not sufficient for autonomous driving, which often requires accurate 3D localization of targets in the scene. Currently, the foremost methods [45, 36, 49, 17] on 3D object detection heavily rely on expensive LiDAR sensors to provide accurate depth information as input. Monocular 3D object detection [3, 2, 31, 44, 19, 26, 16, 30] is a promising low-cost solution, but it is much harder due to the ill-posed nature, i.e., lack of depth cues. The performance gap between LiDAR-based approaches and monocular methods is still substantial.

One key challenge for monocular 3D object detection is in handling large distance variations so that the detector can estimate 3D locations accurately. Learning the distance-specific feature requires specific sophisticated designs [33, 42, 29, 1], while simply learning the feature covering all possible locations is difficult and costs much capacity of the model, resulting in a heavy and slow model for good accuracy. In this work, we solve the learning efficiency problem by introducing a single-stage and multi-scale framework that learns a unified representation of objects in different scales and distance ranges, termed as UR3D. The deep model is relieved from learning different representations for objects within different scale and distance ranges, which significantly reduces the cost of network capacity. Besides, the unified object representation reduces the number of learnable parameters and thus prevents overfitting. Consequently, we achieve accurate monocular 3D object detection with a lightweight network.

An important step for monocular 3D object detection is Non-Maximum Suppression (NMS), which is usually based on the confidence from the classification branch [33, 1]. This may cause omissions of candidate boxes with high-quality 3D information prediction, because the higher classification confidence doesn't always interpret as the better 3D information prediction. To solve the mismatch, we propose a distance-guided NMS, which automatically selects candidate boxes with better distance estimations. With the distance-guided NMS, UR3D achieves better distance estimation and 3D detection accuracy.

Another challenge for monocular 3D object detection is recovering object physical sizes. Such physical parameters are abstract 3D quantities not directly linked to how objects appear in images [14]. It is thus hard to directly predict the physical sizes of 3D bounding boxes by CNNs. Besides, estimating orientations of 3D boxes is shown imprecise by direct regression [14, 31, 1]. To tackle this problem, we propose a fully convolutional cascaded point regression to estimate the projected 2D center points and corner points of 3D boxes accurately and efficiently. Then the predicted keypoints are used to post-optimize the physical sizes and orientations by minimizing a projection-consistency loss [14], which improves the estimates. The contributions of the proposed UR3D are summarised below:

1. UR3D is a single stage and multi-scale framework that can learn a unified representation of objects within different distance ranges for monocular 3D object detection, which leads to a compact and robust network.
2. A distance-guided NMS is proposed, which selects the candidate boxes with better distance estimations.
3. A fully convolutional cascaded point regression is proposed to estimate the projected 2D center points and corner points precisely and efficiently. The predicted keypoints are used to post-optimize the estimated physical sizes and orientations by minimizing a projection-consistency loss.
4. Experimental results on the KITTI [9] autonomous driving dataset show that our method achieves accurate monocular 3D object detection with a compact architecture.

## 2 Related Work

### 2.1 2D Object Detection

**Scale-Aware Designs.** Large scale variation is one of the key challenges for 2D object detection. Image pyramid [41, 20, 48, 37, 39] is a classical solution, but not efficient enough. Faster RCNN [34] utilizes multi-scale anchor boxes to achieve multi-scale object detection. SSD [27] further uses multi-scale features to approximate the image pyramid. Recent works [22, 23, 40, 21] not only adopt multi-scale features, but also share the convolutional weights of detection heads on different layers to get better object representation. However, learning the unified object representation across different scales and distance ranges for monocular 3D object detection is not a trivial problem. The reason is that the quantities for 3D boxes are much more complicated, especially the distance is highly nonlinear. Our UR3D learns robust and compact distance-normalized unified object representation via proposed designs.

**Score Mismatch in NMS.** [13, 12] find that probabilities for class labels naturally reflect classification confidence instead of localization confidence, thus they predict the score or uncertainty of bounding box regression, which can be used to guide the NMS procedure to preserve accurately localized bounding boxes. We reveal the severe score mismatch problem in the NMS of monocular 3D object detection and propose distance-guided NMS to tackle it.

### 2.2 Monocular 3D Object Detection

**Distance-Aware Designs.** Handling large distance variations in monocular 3D object detection is challenging, which requires distance-specific representation. MonoDIS [38] uses a two-stage architecture for monocular 3D object detection, in which the 2D module first detects objects then all the detected objects are fed into a 3D detection head to predict 3D parameters. MonoDIS further disentangles dependencies of different parameters by introducing a loss enabling to handle groups of parameters separately. MonoGRNet [33] is a multi-stage method consisting of four specialized modules for different tasks: 2D detection, instance depth estimation, 3D location estimation and local corner regression. MonoGRNet first predicts objects' 3D locations progressively and then estimates the corner coordinates locally.

MonoPSR [16] uses a network to jointly compute 3D bounding boxes from 2D ones and estimate instance point clouds to help recover shape and scale information. Pseudo-Lidar [42] and AM3D [29] convert the estimated depth image into 3D point clouds to utilize the geometry information, then LiDAR-based 3D object detection methods are employed.

To help the spatial feature learning, OFTNet [35] proposes an orthographic feature transform to map image-level feature into a 3D voxel map, which is then reduced to 2D bird's eye view representation. M3D-RPN [1] is a single-stage framework that exploits 3D anchor boxes to utilize 3D location priors and

proposes depth-aware convolution to generate distance-specific feature, which eases the difficulty of learning the distance-information in the full possible range.

To learn the spatial location information, previous works utilize careful multi-stage designs [38, 33], point cloud feature [16, 42, 29], or feature transformation [35, 1]. Prior methods directly learn object representation covering all possible distance locations, without considering the feature reuse between different distance ranges. UR3D solves the learning efficiency problem by learning a unified representation for objects within different distance ranges.

**3D Box Fitting via Projection-Consistency.** Deep3DBox [31] and M3D-RPN [1] fit better 3D boxes by constraining the consistency between the projected 2D boxes from camera coordinate to image coordinate and the network-predicted 2D boxes. SS3D [14] improves the accuracy of 3D box estimation in the similar way. SS3D further optimizes the 3D location, physical size and orientation together. As a comparison, our UR3D solves the projection-consistency loss of corner points and center points as a post-optimization, but only optimizes physical size and orientation prediction.

### 2.3 Cascaded Point Regression

Cascaded point regression is a classical mechanism for keypoint regression [5, 47, 28]. [47, 28] predict facial keypoints by a multi-stage cascaded structure, i.e., a global stage to predict coarse shapes and local stages using shape-indexed feature as input to predict fine shapes. Previous works mainly focus on cascaded point regression with a single object input, which are inefficient when predicting keypoints for thousands of candidates simultaneously. In contrast, our proposed fully convolutional cascaded point regression makes dense prediction efficient.

## 3 Proposed UR3D

We first detail the overall framework, then present the three key components, i.e., distance-normalized unified representation, followed by the distance-guided NMS, and finally the fully convolutional cascaded point regression and projection-consistency based post-optimization. We term our method as UR3D and the main architecture is illustrated in Fig. 1.

### 3.1 Basic Framework

We address the problem of monocular 3D object detection, which predicts the 3D bounding boxes of targets in camera coordinate from a RGB image. As commonly assumed [9], we only consider yaw angles, and set roll and pitch angles as zero. We also assume that per-image calibration parameters are available both at training and testing phase [9]. For a given RGB image $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$, UR3D reports all objects of concerned categories, and the output for each object is the

1. class label *cls* and confidence *score*,

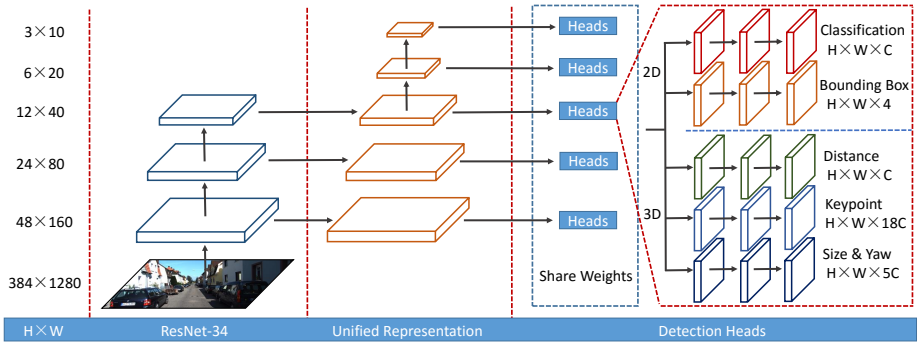



**Fig. 1.** Framework of our UR3D. UR3D learns a compact and robust unified representation for objects within different distance ranges, which relieves the model from learning the complicated distance-specific representation covering all possible locations.

2. 2D bounding box represented by its top-left and bottom-right corners $\mathbf{b} = (a_1, b_1, a_2, b_2)$,
3. 2D projected center point and eight corner points in image coordinate of 3D box in camera coordinate, encoded as $\mathbf{p} = (x_0, x_1, .., x_8, y_0, y_1, .., y_8)$,
4. distance of center point of the 3D bounding box, in image coordinate, encoded as $z_0$,
5. 3D bounding box parameters encoded as $\mathbf{m} = (w, h, l, \sin(\theta), \cos(\theta))$, where $w, h, l$ are the physical dimensions, and $\theta$ is the allocentric pose of the 3D box. UR3D predicts $\sin(\theta)$ and $\cos(\theta)$, then converts them to $\theta$.

UR3D predicts the center point $(x_0, y_0, z_0)$ in image coordinate and converts it to camera coordinate using the calibration parameters during the testing phase.

UR3D is a single-stage and multi-scale architecture (Fig. 1). During the training stage, we assign targets onto five different layers based on their scales. With the rules, we make the scale range of objects assigned on a layer is larger than that of objects assigned on the previous layer. Since the distance is related to scale, objects within different distance ranges are also assigned to different layers. Detailed assignment rules can be found in Section 3.5.

### 3.2 Distance-Normalized Unified Representation

At this part we detail the distance-normalized unified representation. As shown in Fig. 1, there are five different detection heads on each detection layer, corresponding to five tasks, i.e., classification, bounding box regression, distance estimation, keypoint regression and physical size and yaw angle prediction. To learn a unified representation for objects assigned on different detection layers, we first share the learnable weights of the detection heads on different layers, then we normalize each task's training targets on different layers to a same range according to their relationships with scale, details as follows:

**Scale-Invariant Task.** Object category, physical size and orientation are attributes not related to the apparent scale, so the classification and physical size and yaw angle prediction are scale-invariant tasks. Thus the learnable weights of the classification head and size and yaw head on different layers can naturally be shared to form a unified representation between different layers.

**Scale-Linear Task.** The numerical ranges of 2D bounding box and keypoint are linearly dependent on the apparent scale, so the bounding box regression and keypoint regression are two tasks linear to scale. We normalize the targets of these two tasks by introducing learnable parameters $\alpha_i$ and $\beta_i$, and the loss functions of an object are defined as:

$$L_{bbox} = loss(\hat{\mathbf{b}}_i, \mathbf{b}_i) = loss(\hat{\mathbf{b}}_i, \alpha_i \mathbf{b}_i{}'), \tag{1}$$

$$L_{point} = loss(\hat{\mathbf{p}}_i, \mathbf{p}_i) = loss(\hat{\mathbf{p}}_i, \beta_i \mathbf{p}_i{}'), \tag{2}$$

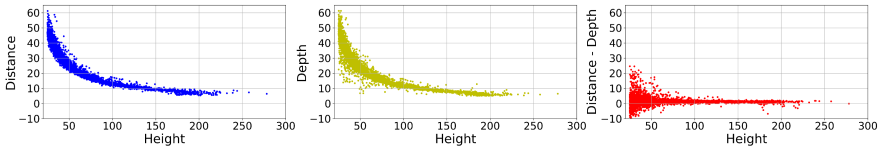
where $i = 0, 1, 2, 3, 4$ denotes the index of the object-assigned detection layer, $\hat{\mathbf{b}}_i$ and $\hat{\mathbf{p}}_i$ are groundtruths of bounding box regression and keypoint regression respectively, $\mathbf{b}_i{}'$ and $\mathbf{p}_i{}'$ are network-predicted bounding box regression result and keypoint regression result respectively, $0 < \alpha_0 < \alpha_1 < \alpha_2 < \alpha_3 < \alpha_4$ and $0 < \beta_0 < \beta_1 < \beta_2 < \beta_3 < \beta_4$. During the training phase, the network learns the best normalization parameters $\alpha_i$ and $\beta_i$ automatically. During the testing phase, we use $\mathbf{b}_i = \alpha_i \mathbf{b}_i{}'$ and $\mathbf{p}_i = \beta_i \mathbf{p}_i{}'$ as outputs for the bounding box regression and keypoint regression respectively.

**Scale-Nonlinear Task.** To investigate the relationship between distance values and apparent scales, we show some statistics of the car category in KITTI training set [9] in Fig. 2(a). The left figure shows the relationship of distance vs. height, the middle figure shows the curve of depth value of the center point vs. height, and the right figure shows their difference vs. height. The depth images are generated by a monocular depth estimation model [8] as in [42, 29]. Apparently the relationships of distance vs. height and depth vs. height are highly nonlinear but in the similar trends (left figure and middle figure), i.e., subtracting the depth can reduce the degree of nonlinearity of distance (right figure).

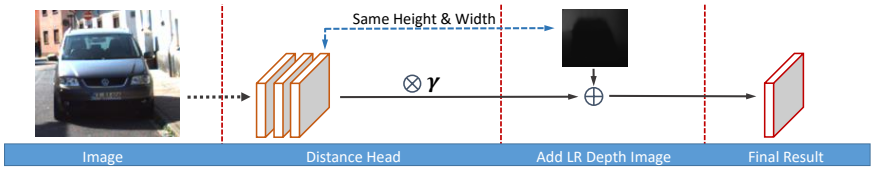
To get accurate distance estimation, we first introduce learnable parameters $\gamma_i$ multiplied with the output of $i_{th}$ distance head to use a piece-wise linear curve to fit the nonlinear distance curve. However, the capacity of our piece-wise linear distance estimation model consisting of only five parts is limited, and we still cannot fit the highly nonlinear distance precisely. We further subtract the depth value of a low resolution depth image with the same size of the distance head (Fig. 2(b)), to reduce the degree of nonlinearity of distance, which significantly eases the distance learning. The distance loss of an object is defined as:

$$L_{dist} = loss(\hat{z_{0}}_i, z_{0i}) = loss(\hat{z_{0}}_i, \gamma_i z_{0i}{}' + depth), \tag{3}$$

where $i = 0, 1, 2, 3, 4$ denotes the index of the object-assigned detection layer, $\hat{z_{0}}_i$ is the groundtruth distance, $z_{0i}{}'$ is the network-predicted distance result,




(a) Statistics of car category in KITTI [9] training set. The curves of distance vs. height and depth vs. height are highly nonlinear but in the similar trends. Distance and depth are in image coordinate, height is in pixels. The curves are general conclusions not only limited to KITTI [9], under the assumption of autonomous driving application.




(b) Illustration of the distance head. We first introduce the learnable parameter $\gamma$ multiplied with the output to use a piece-wise linear curve to fit the nonlinear distance curve, then we further add the depth value from a estimated depth image to reduce the nonlinearity degree of distance. Such designs ease the distance learning significantly.

**Fig. 2.** Illustration of distance estimation method.

$\gamma_0 > \gamma_1 > \gamma_2 > \gamma_3 > \gamma_4 > 0$, and *depth* is the depth value from the corresponding position of the low resolution depth image. During the training phase, the network can learn the best slope parameters $\gamma_i$ automatically. During the testing phase, we use $z_{0i} = \gamma_i z_{0i}{}' + depth$ as output for distance estimation. For both train and test, we run the depth estimation model [8] once and downsample the depth map five times to feed into each distance head, and the maximum size of depth maps we need is only one eighth of the size of depth maps required by [42, 29].

### 3.3 Distance-Guided NMS

In this part, we detail the distance-guided NMS. Firstly, to get the score of distance estimation, we extend an uncertainty-aware regression loss [15] for distance estimation, as follows:

$$L_{dist}(\hat{z}_0, z_0) = \lambda_{dist} \frac{loss(\hat{z}_0, z_0)}{\sigma^2} + \lambda_{uncertain} log(\sigma^2), \tag{4}$$

where $\hat{z}_0$ and $z_0$ are the groundtruth and estimated distance respectively, $loss(\hat{z}_0, z_0)$ is a normal regression loss, $\lambda_{dist}$ and $\lambda_{uncertain}$ are positive parameters to balance the two parts. $\sigma^2$ is a positive learnable parameter and $\frac{1}{\sigma^2}$ can be regarded as the score of distance estimation.

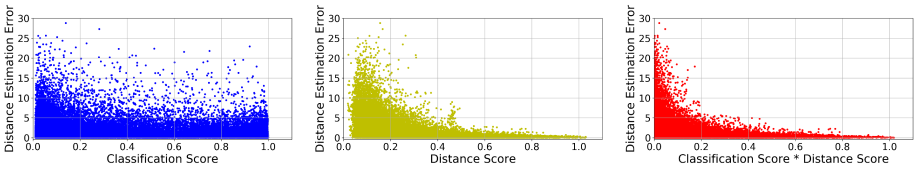



**Fig. 3.** Illustration for distance estimation error vs. different scores of candidate boxes. Classification score × distance score can best push boxes with inaccurate estimates to the left side. Statistics are based on a UR3D model trained with KITTI [9] car class.

**Algorithm 1** Distance-Guided NMS.
$\mathcal{B}$: $N \times 27$ matrix of initial 2D/3D boxes,
$\mathcal{S}$: corresponding classification scores,
$\mathcal{C}$: classification scores normalized by distance estimation variances,
$\mathcal{D}$: final detection set, $\Omega_{\text{nms}}$: NMS threshold,
$top(k)$: function finding the top $k$ largest elements,
$Ave(z_1, .., z_K, w_1, .., w_K)$: function returning the average of $z_1, .., z_K$ weighted by $w_1, .., w_K$,
$K$: number of boxes participating the average.
The lines in blue and in red are traditional NMS and Distance-Guided NMS respectively.

**Input:** $\mathcal{B} = \{o_1, o_2, .., o_N\}$,
$\mathcal{S} = \{score_1, score_2, .., score_N\}$,
$\mathcal{C} = \{\frac{score_1}{\sigma_1^2}, \frac{score_2}{\sigma_2^2}, .., \frac{score_N}{\sigma_N^2}\}$, $K$, $\Omega_{\text{nms}}$,
**Output:** $\mathcal{D} \leftarrow \{\}$
**while** $\mathcal{B} \neq$ empty **do**
$m \leftarrow \arg\max \mathcal{S}$
$\mathcal{D} \leftarrow \mathcal{D} \bigcup o_m$
$\mathcal{B} \leftarrow \mathcal{B} - o_m$
$m_1, m_2, .., m_K \leftarrow \arg \text{top}(K)\mathcal{C}$
$temp \leftarrow o_{m_1}$
$temp.z_0 =$
$\text{Ave}(o_{m_1}.z_0, .., o_{m_K}.z_0, \frac{score_{m_1}}{\sigma_{m_1}^2}, .., \frac{score_{m_K}}{\sigma_{m_K}^2})$
$\mathcal{D} \leftarrow \mathcal{D} \bigcup temp$
$\mathcal{B} \leftarrow \mathcal{B} - o_{m_1}$
$\mathcal{T} \leftarrow \mathcal{B}$
**for** $o_i \in \mathcal{B}$ **do**
**if** $\text{IoU}(o_m, o_i) > \Omega_{\text{nms}}$ **then**
$\mathcal{T} \leftarrow \mathcal{T} - o_i$
**end if**
**end for**
$\mathcal{B} \leftarrow \mathcal{T}$
**end while**
**return** $\mathcal{D}$

In Fig. 3, we show the correlations between the distance estimation error of predicted 3D bounding boxes and corresponding *score*, $\frac{1}{\sigma^2}$, $\frac{score}{\sigma^2}$. As can be seen, $\frac{score}{\sigma^2}$ best pushes candidates with inaccurate distance estimates to the left side. Traditional NMS does not select the candidate boxes with better distance estimates, we propose Distance-Guided NMS (Algorithm 1) to solve the problem.

### 3.4 Fully Convolutional Cascaded Point Regression

The proposed efficient fully convolutional cascaded point regression (Fig. 4) is adapted from [4] and consists of two stages. In the first stage, we directly regress the positions of center point and eight corner points, and the results of position $q$ are encoded as:

$$\mathbf{p}_0 = \{p_0, p_1, \ldots, p_8\} = \{(x_0, y_0), (x_1, y_1), \ldots, (x_8, y_8)\},$$

In the second stage, we extract the shape-indexed feature guided by $\mathbf{p}_0$, and predict the residual values of keypoints. The extraction of shape-indexed feature can be formulated as an efficient convolutional layer as in [4], instead of traditional time-consuming multi-patch extraction [47, 28]. Let the nine positions of a $3 \times 3$ convolutional kernels correspond to the nine keypoints. The convolutional layer for the extraction consists of two steps: 1) sampling using $\mathbf{p}_0$ as the kernel point positions over the input feature map $\mathbf{f}_{in}$; 2) summation of sampled values weighted by kernel weights $\mathbf{w}$ to get the output feature map $\mathbf{f}_{out}$, i.e.,

$$\mathbf{f}_{out}(q) = \sum_{i=0}^{8} \mathbf{w}(i) \cdot \mathbf{f}_{in}(p_i). \tag{5}$$

The sampling is on the irregular locations. As the location $p_i$ is typically fractional, Eq. (5) $\mathbf{f}_{in}(p_i)$ is obtained by bilinear interpolation. The detailed implementation is similar to [4]. Note during the training, the gradients will not be backpropagated to $p_i$ through Eq. (5), because $p_i$ has its own supervised loss. The keypoint losses for two stages are:

$$L_{point_0} = loss(\hat{\mathbf{p}}, \mathbf{p}_0), \tag{6}$$

$$L_{point_1} = loss(\hat{\mathbf{p}}, \mathbf{p}) = loss(\hat{\mathbf{p}}, \mathbf{p}_0 + \mathbf{p}_1), \tag{7}$$

where $\hat{\mathbf{p}}$ is the groundtruth of keypoint regression, $\mathbf{p}_0$ and $\mathbf{p}_1$ are the outputs of the first and second stage respectively, and $\mathbf{p} = \mathbf{p}_0 + \mathbf{p}_1$ is the final output of keypoint regression.

Fully convolutional cascaded point regression achieves accurate prediction of thousands of candidates simultaneously. Then we use the estimated keypoints to post-optimize the physical size and yaw angle prediction. Given a set of center point $(x_0, y_0, z_0)$, physical size $w, h, l$, and yaw angle $\theta$, we calculate the center and corner points of corresponding 3D bounding box in camera coordinate with

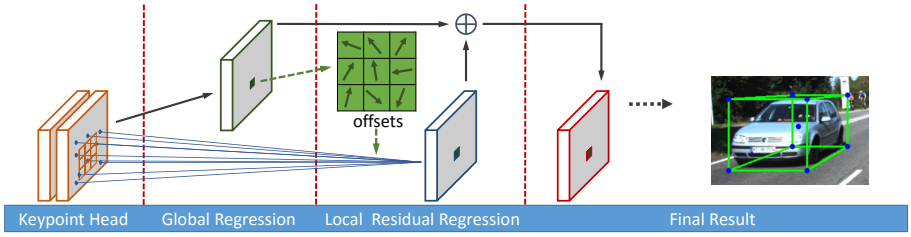



**Fig. 4.** Illustration of fully convolutional cascaded point regression, which formulates dense cascaded point regression as an efficient convolutional layer.

calibration parameters. Denote the calculation function as $\mathbf{F}(x_0, y_0, z_0, w, h, l, \theta)$. We try to find a set of $w', h', l', \theta'$ to minimize the objective function:

$$\begin{aligned} \arg \min_{w',h',l',\theta'} & \lambda_{post} \left\| \mathbf{F}(x_0, y_0, z_0, w', h', l', \theta') - \mathbf{p} \right\|_2^2 \\ & + \left[ (w' - w)^2 + (h' - h)^2 + (l' - l) \right]^2, \end{aligned} \tag{8}$$

where $x_0, y_0, z_0, w, h, l, \theta$ are the network-predicted results, $w', h', l', \theta'$ are the post-optimized results. This is a standard nonlinear optimization problem, which can be solved by an optimization toolbox.

### 3.5 Implementation Details

**Object Assignment Rule.** During the training stage, we assign a position $q$ on a detection layer $\mathbf{f}_i$ $(i = 0, 1, 2, 3, 4)$ to an object, if 1) $q$ falls in the object, 2) the maximum distance from $q$ to the boundaries of the object is within a given range $\mathbf{r}_i$, and 3) the distance from $q$ to the center of the object is less than a given value $\mathbf{d}_i$. $\mathbf{r}_i$ denotes the scale range of objects assigned on each detection layer [40], and $\mathbf{d}_i$ defines the radius of positive samples on each detection layer. $\mathbf{r}_i$ is $[0, 64], [64, 128], [128, 256], [256, 512], [512, 1024]$ for the five layers, and $\mathbf{d}_i$ is $12, 24, 48, 96, 192$ respectively, all in pixels. Positions without assigning to any object will be regarded as negative samples, except that the positions adjacent with the positive samples are treated as ignored samples.

**Network Architecture.** The backbone of UR3D is ResNet-34 [11]. All the head depth of the detection heads is two. Images are scaled to a fixed height of 384 pixels for both training and testing.

**Loss.** We use the focal loss [23] for classification task, IoU loss [46] for bounding box regression, smooth $L_1$ loss [10] for keypoint regression, and Wing loss [7] for distance, size and orientation estimation. The loss weights are $1, 1, 0.003, 0.1, 0.05, 0.1, 0.001$ for the classification, bounding box regression, keypoint regression, distance estimation, distance variance estimation, size and orientation estimation, and post-optimization, respectively.

**Optimization.** We adopt the step strategy to adjust a learning rate. At first the learning rate is fixed to 0.01 and reduced by 50 times every $3 \times 10^4$ iterations. The total iteration number is $9 \times 10^4$ with batch size 5. The only augmentation we perform is random mirroring. We implement our framework using Python and PyTorch [32]. All the experiments run on a server with 2.6GHz CPU and GTX Titan X.

## 4 Experiments

We evaluate our method on KITTI [9] dataset with the car class under the two 3D localization tasks: Bird's Eye View (BEV) and 3D Object Detection. The method is comprehensively tested on two validation splits [3, 43] and the official test dataset. We further present analyses on the impacts of individual components of the proposed UR3D. Finally we visualize qualitative examples of UR3D on KITTI (Fig. 5).

### 4.1 KITTI

The KITTI [9] dataset provides multiple widely used benchmarks for computer vision problems in autonomous driving. The Bird's Eye View (BEV) and 3D Object Detection tasks are used to evaluate 3D localization performance. These two tasks are characterized by 7481 training and 7518 test images with 2D and 3D annotations for cars, pedestrians, cyclists, etc. Each object is assigned with a difficulty level, i.e., easy, moderate or hard, based on its occlusion level and truncation degree.

We conduct experiments on three common data splits including val1 [3], val2 [43], and the official test split [9]. Each split contains images from non-overlapping sequences such that no data from an evaluated frame, or its neighbors, are used for training. We report the $\text{AP}|_{R_{11}}$ and $\text{AP}|_{R_{40}}$ on val1 and val2, and $\text{AP}|_{R_{40}}$ on test subset. We use the car class, the most representative, and the official IoU criteria for cars, i.e., 0.7.

**Val Set Results.** We evaluate UR3D on val1 and val2 as detailed in Tab. 1 and Tab. 2. Using the same monocular depth estimator [8] as in AM3D [29] and Pseudo-LiDAR [42], UR3D can compete with them on the two splits. The time cost of depth map generation of our UR3D can be much smaller than that of [29, 42], since the size of depth maps we need is only one eighth of the size of depth maps required by them. We use depth priors to normalize the learning targets of distance instead of converting to point clouds as in [29, 42], leading to a more compact and efficient architecture.

**Test Set Results.** We evaluate the results on test set in Tab. 3. Compared with FQNet [26], ROI-10D [30], GS3D [19], and MonoGRNet [33], UR3D outperforms

| Method | Time(ms) | $AP\|_{R_{11}}$ [val1 / val2 ] | | | $AP\|_{R_{40}}$ [val1 / val2 ] | | |
|---|---|---|---|---|---|---|---|
| | | Easy | Mod | Hard | Easy | Mod | Hard |
| ROI-10D [30] | 200 | 14.76 / – | 9.55 / – | 7.57 / – | – / – | – / – | – / – |
| MonoPSR [16] | 200 | 20.63 / 21.52 | 18.67 / 18.90 | 14.45 / 14.94 | – / – | – / – | – / – |
| MonoGRNet [33] | 60 | 24.97 / – | 19.44 / – | 16.30 / – | 19.72 / – | 12.81 / – | 10.15 / – |
| M3D-RPN [1] | 160 | 25.94 / 26.86 | 21.18 / 21.15 | 17.90 / 17.14 | 20.85 / 21.36 | 15.62 / 15.22 | 11.88 / 11.28 |
| Pseudo-LiDAR [42] | – | 40.60 / – | 26.30 / – | 22.90 / – | – / – | – / – | – / – |
| AM3D [29] | 400 | 43.75 / – | 28.39 / – | 23.87 / – | – / – | – / – | – / – |
| UR3D (Ours) | 120 | 37.35 / 36.15 | 26.01 / 25.25 | 20.84 / 20.12 | 33.07 / 32.35 | 20.84 / 20.05 | 15.25 / 14.4 |

**Table 1. Bird's Eye View**. Comparisons on the Bird's Eye View task ($AP_{BEV}$) on val1 [3] and val2 [43] of KITTI [9].

| Method | Time(ms) | $AP\|_{R_{11}}$ [val1 / val2 ] | | | $AP\|_{R_{40}}$ [val1 / val2 ] | | |
|---|---|---|---|---|---|---|---|
| | | Easy | Mod | Hard | Easy | Mod | Hard |
| ROI-10D [30] | 200 | 10.25 / – | 6.39 / – | 6.18 / – | – / – | – / – | – / – |
| MonoPSR [16] | 200 | 12.75 / 13.94 | 11.48 / 12.24 | 8.59 / 10.77 | – / – | – / – | – / – |
| MonoGRNet [33] | 60 | 13.88 / – | 10.19 / – | 7.62 / – | 11.90 / – | 7.56 / – | 5.76 / – |
| M3D-RPN [1] | 160 | 20.27 / 20.40 | 17.06 / 16.48 | 15.21 / 13.34 | 14.53 / 14.57 | 11.07 / 10.07 | 8.65 / 7.51 |
| Pseudo-LiDAR [42] | – | 28.20 / – | 18.50 / – | 16.40 / – | – / – | – / – | – / – |
| AM3D [29] | 400 | 32.23 / – | 21.09 / – | 17.26 / – | – / – | – / – | – / – |
| UR3D (Ours) | 120 | 28.05 / 26.30 | 18.76 / 16.75 | 16.55 / 13.60 | 23.24 / 22.15 | 13.35 / 11.10 | 10.15 / 9.15 |

**Table 2. 3D Detection**. Comparisons on the 3D Detection task ($AP_{3D}$) on val1 [3] and val2 [43] of KITTI [9].

| Method | Reference | Time(ms) | $AP\|_{R_{40}}$ [Easy / Mod / Hard ] | |
|---|---|---|---|---|
| | | | $AP_{3D}$ | $AP_{BEV}$ |
| FQNet [26] | CVPR 2019 | 500 | 2.77 / 1.51/ 1.01 | 5.40 / 3.23/ 2.46 |
| ROI-10D [30] | CVPR 2019 | 200 | 4.32 / 2.02/ 1.46 | 9.78 / 4.91/ 3.74 |
| GS3D [19] | CVPR 2019 | 2000 | 4.47 / 2.90 / 2.47 | 8.41 / 6.08/ 4.94 |
| MonoGRNet [33] | AAAI 2019 | 60 | 9.61 / 5.74/ 4.25 | 18.19 / 11.17/ 8.73 |
| MonoDIS [38] | ICCV 2019 | - | 10.37 / 7.94 / 6.40 | 17.23 / 13.19/ 11.12 |
| M3D-RPN [1] | ICCV 2019 | 160 | 14.76 / 9.71 / 7.42 | 21.02 / 13.67/ 10.23 |
| AM3D [29] | ICCV 2019 | 400 | 16.50 / 10.74 / 9.52 | 25.03 / 17.32/ 14.91 |
| UR3D (Ours) | | 120 | 15.58 / 8.61/ 6.00 | 21.85 / 12.51/ 9.20 |

**Table 3. Test Set Results**. Comparisons of our UR3D to SOTA methods of monocular 3D object detection on the test set of KITTI [9].

them significantly in all indicators. Compared with MonoDIS [38], UR3D outperforms it by a large margin in three indicators, i.e., $AP_{3D}$ of easy subset, $AP_{3D}$ of moderate subset and $AP_{BEV}$ of easy subset. Note MonoDIS [38] is a two-stage method while ours is a more compact single-stage method. Compared with another single-stage method, M3D-RPN [1], UR3D outperforms it on two indicators, i.e., $AP_{3D}$ and $AP_{BEV}$ of easy subset, with a more lightweight backbone. Compared with AM3D [29], UR3D runs with a much faster speed.

| Setting | Time(ms) | $AP\|_{R_{40}}$ [Easy / Mod / Hard] $AP_{3D}$ | $AP_{BEV}$ |
|---|---|---|---|
| Baseline | 90 | 11.26 / 6.52 / 4.25 | 18.26 / 10.25 / 8.55 |
| + LR Depth Image | 90 | 18.57 / 10.65 / 7.90 | 27.65 / 15.25 / 12.52 |
| + Distance Guided NMS ($K = 1$) | 90 | 19.75 / 11.35 / 8.50 | 30.55 / 18.47 / 14.18 |
| + Distance Guided NMS ($K = 2$) | 90 | 20.20 / 11.59 / 8.85 | 31.69 / 19.68 / 14.81 |
| + Post-Optimization | 110 | 22.50 / 12.95 / 9.90 | 32.58 / 20.54 / 15.05 |
| + Cascaded Regression (UR3D) | 120 | 23.24 / 13.35 / 10.15 | 33.07 / 20.84 / 15.25 |

**Table 4. Ablations**. We ablate the effects of key components of UR3D with respect to accuracy and inference time.

**Learned Parameters.** We initialize $\alpha_i$ and $\beta_i$ with $32, 64, 128, 256, 512$, and $16, 8, 4, 2, 1$ for $\gamma_i$. The learned results on val1 split are $5.7, 10.6, 20.7, 41.0, 82.3$ for $\alpha_i$, $5.3, 10.4, 20.6, 41.4, 82.2$ for $\beta_i$, and $2.3, 1.4, 0.8, 0.3, 0.2$ for $\gamma_i$.

### 4.2 Ablation Study

We conduct ablation experiments to examine how each proposed component affects the final performance of UR3D. We evaluate the performance by first setting a simple baseline which doesn't adopt proposed components, then adding the proposed designs one-by-one, as shown in Tab. 4. For all ablations we use the KITTI val1 dataset split and evaluate based on the car class. From the results listed in Tab. 4, some promising conclusions can be summed up as follows:

**Distance-Normalized Unified Representation is crucial.** The results of "+ LR Depth Image" show that adding the low resolution depth image to help normalize the distance improves the $AP_{3D}$ and $AP_{BEV}$ of baseline a lot, which indicates that reducing the nonlinear degree of distance estimation eases the unified object representation learning dramatically.

**Distance-Guided NMS is promising.** The $AP_{3D}$ and $AP_{BEV}$ of "+ Distance-Guided NMS ($K = 1$)" are much better than the results of "+ LR Depth Image". It supports that our distance-guided NMS can select the candidate boxes with better distance estimates automatically and effectively. Increasing the number of candidates participating the average (from $K = 1$ to $K = 2$) also helps, suggesting that the candidate with the best distance estimate may not be the top one but among the top K due to the noise of distance score.

**Fully Convolutional Cascaded Point Regression is effective.** The results of "+ Post-Optimization" illustrate that introducing the projection-consistency based post-optimization improves $AP_{3D}$ and $AP_{BEV}$. The results of "+ Cascaded Regression" show that adding the fully convolutional cascaded point regression

further improves $AP_{3D}$ and $AP_{BEV}$. The fully convolutional cascaded point regression only costs $10ms$ with a non-optimized Python implementation.

**Fig. 5. Qualitative Examples**. We visualize qualitative examples of UR3D. All illustrated images are from the val1 [3] split and not used for training. Bird's eye view results (right) are also provided and the red lines indicate the yaw angles of cars.

## 5 Conclusions

In this work, we present a monocular 3D object detector, i.e., UR3D, which learns a distance-normalized unified object representation, in contrast to prior works which learns to represent objects in full possible range. UR3D is uniquely designed to learn the shared representation across different distance ranges, which is robust and compact. We further propose a distance-guided NMS to select candidate boxes with better distance estimates and a fully convolutional cascaded point regression predicting accurate keypoints to post-optimize the 3D boxes parameters, both of which improve the accuracy. Collectively, our method achieves accurate monocular 3D object detection with a compact architecture.

**Acknowledgment** The authors are partly funded by Huawei.

## References


1. Brazil, G., Liu, X.: M3D-RPN: monocular 3d region proposal network for object detection. In: ICCV. pp. 9287–9296 (2019)
2. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: CVPR. pp. 2147–2156 (2016)
3. Chen, X., Kundu, K., Zhu, Y., Berneshawi, A.G., Ma, H., Fidler, S., Urtasun, R.: 3d object proposals for accurate object class detection. In: NeurIPS. pp. 424–432 (2015)
4. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: ICCV. pp. 764–773 (2017)
5. Dollár, P., Welinder, P., Perona, P.: Cascaded pose regression. In: CVPR. pp. 1078–1085 (2010)
6. Everingham, M., Eslami, S.M.A., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision **111**(1), 98–136 (2015)
7. Feng, Z., Kittler, J., Awais, M., Huber, P., Wu, X.: Wing loss for robust facial landmark localisation with convolutional neural networks. In: CVPR. pp. 2235–2245 (2018)
8. Fu, H., Gong, M., Wang, C., Batmanghelich, K., Tao, D.: Deep ordinal regression network for monocular depth estimation. In: CVPR. pp. 2002–2011 (2018)
9. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: CVPR. pp. 3354–3361 (2012)
10. Girshick, R.B.: Fast R-CNN. In: ICCV. pp. 1440–1448 (2015)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
12. He, Y., Zhu, C., Wang, J., Savvides, M., Zhang, X.: Bounding box regression with uncertainty for accurate object detection. In: CVPR. pp. 2888–2897 (2019)
13. Jiang, B., Luo, R., Mao, J., Xiao, T., Jiang, Y.: Acquisition of localization confidence for accurate object detection. In: ECCV. pp. 816–832 (2018)
14. Jörgensen, E., Zach, C., Kahl, F.: Monocular 3d object detection and box fitting trained end-to-end using intersection-over-union loss. CoRR **abs/1906.08070** (2019)
15. Kendall, A., Gal, Y., Cipolla, R.: Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: CVPR. pp. 7482–7491 (2018)
16. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: CVPR. pp. 11867–11876 (2019)
17. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR. pp. 12697–12705 (2019)
18. LeCun, Y., Bengio, Y., Hinton, G.E.: Deep learning. Nature **521**(7553), 436–444 (2015)
19. Li, B., Ouyang, W., Sheng, L., Zeng, X., Wang, X.: GS3D: an efficient 3d object detection framework for autonomous driving. In: CVPR. pp. 1019–1028 (2019)
20. Li, H., Lin, Z., Shen, X., Brandt, J., Hua, G.: A convolutional neural network cascade for face detection. In: CVPR. pp. 5325–5334 (2015)
21. Li, Y., Chen, Y., Wang, N., Zhang, Z.: Scale-aware trident networks for object detection. In: CVPR. pp. 6054–6063 (2019)
22. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. In: CVPR. pp. 936–944 (2017)

23. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2999–3007 (2017)
24. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: ECCV. pp. 740–755 (2014)
25. Liu, L., Ouyang, W., Wang, X., Fieguth, P.W., Chen, J., Liu, X., Pietikäinen, M.: Deep learning for generic object detection: A survey. International Journal of Computer Vision **128**(2), 261–318 (2020)
26. Liu, L., Lu, J., Xu, C., Tian, Q., Zhou, J.: Deep fitting degree scoring network for monocular 3d object detection. In: CVPR. pp. 1057–1066 (2019)
27. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. In: ECCV. pp. 21–37 (2016)
28. Lv, J., Shao, X., Xing, J., Cheng, C., Zhou, X.: A deep regression architecture with two-stage re-initialization for high performance facial landmark detection. In: CVPR. pp. 3691–3700 (2017)
29. Ma, X., Wang, Z., Li, H., Ouyang, W., Zhang, P.: Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In: ICCV. pp. 6851–6860 (2019)
30. Manhardt, F., Kehl, W., Gaidon, A.: ROI-10D: monocular lifting of 2d detection to 6d pose and metric shape. In: CVPR. pp. 2069–2078 (2019)
31. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: CVPR. pp. 5632–5640 (2017)
32. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS. pp. 8024–8035 (2019)
33. Qin, Z., Wang, J., Lu, Y.: Monogrnet: A geometric reasoning network for monocular 3d object localization. In: AAAI. pp. 8851–8858 (2019)
34. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NeurIPS. pp. 91–99 (2015)
35. Roddick, T., Kendall, A., Cipolla, R.: Orthographic feature transform for monocular 3d object detection. In: British Machine Vision Conference (2019)
36. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: CVPR. pp. 770–779 (2019)
37. Shi, X., Shan, S., Kan, M., Wu, S., Chen, X.: Real-time rotation-invariant face detection with progressive calibration networks. In: CVPR. pp. 2295–2303 (2018)
38. Simonelli, A., Bulò, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: ICCV. pp. 1991–1999 (2019)
39. Singh, B., Davis, L.S.: An analysis of scale invariance in object detection SNIP. In: CVPR. pp. 3578–3587 (2018)
40. Tian, Z., Shen, C., Chen, H., He, T.: FCOS: fully convolutional one-stage object detection. In: CVPR. pp. 9627–9636 (2019)
41. Viola, P.A., Jones, M.J.: Robust real-time face detection. International Journal of Computer Vision **57**(2), 137–154 (2004)
42. Wang, Y., Chao, W., Garg, D., Hariharan, B., Campbell, M.E., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: CVPR. pp. 8445–8453 (2019)
43. Xiang, Y., Choi, W., Lin, Y., Savarese, S.: Subcategory-aware convolutional neural networks for object proposals and detection. In: WACV. pp. 924–933 (2017)

44. Xu, B., Chen, Z.: Multi-level fusion based 3d object detection from monocular images. In: CVPR. pp. 2345–2353 (2018)
45. Yang, B., Luo, W., Urtasun, R.: PIXOR: real-time 3d object detection from point clouds. In: CVPR. pp. 7652–7660 (2018)
46. Yu, J., Jiang, Y., Wang, Z., Cao, Z., Huang, T.S.: Unitbox: An advanced object detection network. In: ACM MM. pp. 516–520. ACM (2016)
47. Zhang, J., Shan, S., Kan, M., Chen, X.: Coarse-to-fine auto-encoder networks (CFAN) for real-time face alignment. In: ECCV. pp. 1–16 (2014)
48. Zhang, K., Zhang, Z., Li, Z., Qiao, Y.: Joint face detection and alignment using multitask cascaded convolutional networks. IEEE Signal Process. Lett. **23**(10), 1499–1503 (2016)
49. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: CVPR. pp. 4490–4499 (2018)