Weak Cube R-CNN: Weakly Supervised 3D Detection using only 2D Bounding Boxes

Andreas Lau Hansen[1] Lukas Wanzeck[1] Dim P. Papadopoulos[1,2]
[1] Technical University of Denmark [2] Pioneer Center for AI
https://weakcubercnn.compute.dtu.dk/

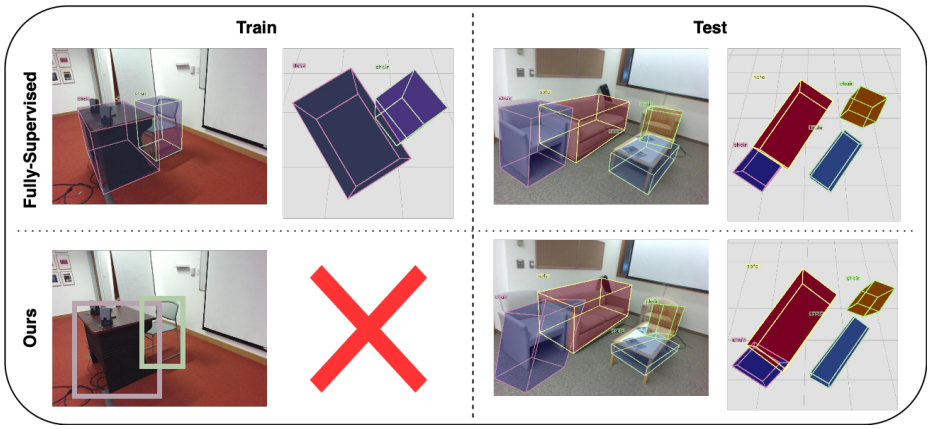

Fig. 1. Weak Cube R-CNN. In contrast to standard 3D object detectors that require 3D ground truths, our proposed method is trained using only 2D bounding boxes but can predict 3D cubes at test time. Weak Cube R-CNN significantly reduces the annotation time since 3D ground-truths require 11× more time than annotating 2D boxes. More importantly, it does not require access to LiDAR or multi-camera setups.

Abstract. Monocular 3D object detection is an essential task in computer vision, and it has several applications in robotics and virtual reality. However, 3D object detectors are typically trained in a fully supervised way, relying extensively on 3D labeled data, which is labor-intensive and costly to annotate. This work focuses on weakly-supervised 3D detection to reduce data needs using a monocular method that leverages a single-camera system over expensive LiDAR sensors or multi-camera setups. We propose a general model *Weak Cube R-CNN*, which can predict objects in 3D at inference time, requiring only 2D box annotations for training by exploiting the relationship between 2D projections of 3D cubes. Our proposed method utilizes pre-trained frozen foundation 2D models to estimate depth and orientation information on a training set. We use these estimated values as pseudo-ground truths during training. We design loss functions that avoid 3D labels by incorporating information from the external models into the loss. In this way, we aim to implicitly transfer knowledge from these large foundation 2D models without having access to 3D bounding box annotations. Experimental results on

the SUN RGB-D dataset show increased performance in accuracy compared to an annotation time equalized Cube R-CNN [3] baseline. While not precise for centimetre-level measurements, this method provides a strong foundation for further research.

Keywords: Weak Supervision · 3D Object Detection · Monocular Object Detection.

1 Introduction

The ability to tell physical scale, distance between, and depth of objects is a very natural ability for humans and animals with binocular vision. It is, therefore, a manageable task to place objects in the three-dimensional space. However, most digital photos are taken with monocular cameras, particularly smartphones, and thus cannot benefit from the same stereoscopic effects. Still, many depth and size cues are present in an image.

3D object detectors [3,15,23,26,32,44] can pick up on many depth cues nicely. The ability to locate objects in physical space with accurate dimensions provides new use cases within scene understanding, augmented reality, room mapping, and more. However, collecting 3D annotated datasets is challenging partly due to the requirement of special physical sensors; LiDAR, depth scanners, or other alternatives and partly because of the annotation time and complexity. As an example, to annotate the 3D data set SUN-RGBD [37] 2051 hours were spent, which is considered to be a tiny data set with its 10,335 images. That equates to roughly 12 minutes per image compared to 65s for 2D annotation ($\approx 9\%$). This is not counting the extra data collection time. On the other hand, an abundance of 2D annotated datasets can be leveraged. This motivates the development of a 3D detector that relies solely on 2D annotations.

Other work has tackled this problem by using additional sensors during inference, primarily LiDAR [7, 24], pseudo LiDAR [29, 40], and stereo cameras [6, 17, 28], as they provide an accurate representation of 3D space. Most of these systems are employed in simplified driving scenarios [5, 8] where visual cues are stable, for example is it easy to tell the orientation of the world as there is a large, unobstructed view of the ground and it is only required to estimate rotation about one axis. However, in a more generalized setting that we are interested in, these cues cannot be expected to be present. Thus, something more generalizable is required.

We propose a CNN-based model which only uses 2D annotated data during training to perform 3D object detection. It predicts objects' location in the image plane and then learns to place the objects in 3D by using "weak losses" using only 2D ground truths. A crucial component in this step is to use foundation models for estimating depth and the ground plane. The output of these models is used as pseudo ground truths for the weak losses.

We use a Faster R-CNN [33] type architecture to first predict 2D bounding boxes of objects. The boxes' location is then used as candidate regions for 3D cubes by pooling from backbone feature maps used in a 3D head to predict 3D

cubes. Besides the 3D head, information from a metric depth estimation model is used to estimate the depth and the ground plane. The parameters of the 3D cubes are optimized, such that the 2D and 3D attributes are consistent. For image plane localization, the Generalised IoU loss [34] is employed. The pseudo ground truth depth is sampled from a depth map inferred from the image. Additionally, object size priors are incorporated through a relaxed loss, which ensures that object sizes within a particular class roughly match average-sized objects of the same class. To estimate the rotation, Pose Alignment Loss operates on the objects internally within a scene and uses the assumption that objects are typically aligned in one or more axes. A normal vector loss ensures rotational consistency with the ground, which provides a world frame of reference. Experimental results on SUN RGB-D [37] show increased performance in accuracy compared to an annotation time equalized Cube R-CNN [3] baseline.

2 Related work

Monocular 3D object detection. Monocular 3D Object Detection is the task of predicting 3D bounding boxes of target objects within a single 2D RGB image. This task relies solely on RGB data without additional information such as depth, sensor data, or multiple images. The most prominent uses of 3D object detection are self-driving cars [4,22,25,32,36,39], and indoor spatial room modeling. Cube R-CNN [3] is a simple extension of the established 2D object detection methods. The method is at its core Faster R-CNN with a cube head attached to it, such that it can predict a cube for each 2D box. The idea of leveraging an existing 2D object detector and extrapolating cubes from 2D boxes is used by more methods [25] (YOLO3D). Their key assumption is the fact that a cube fits tightly into a 2D box. Their idea of proposing cubes is similar to RPNs [33], consisting of some simplifications which constrain the number of 3D proposal boxes inside each 2D box. Other methods [16, 18, 41, 45] focus on modeling objects' depth. MonoDETR is among transformer based methods [11, 44], but still uses a CNN as both the feature and depth encoder. The transformer blocks fuse the image and depth features. 3D Datasets vary in distance to objects and field of vision, from indoor [1, 35, 37] to outdoor scenes, where the camera is mounted on a car [5, 8]. Effort has been made to homogenise datasets [3].

Weakly supervised 3D object detection. The type and level of weak supervision used in other work varies substantially. Examples are: click- [24], point cloud-, direction supervision [38], and 2D box supervision [12]. Other methods [9, 38] use multiple frames obtained through a video to mimic a stereo view camera. Many weakly supervised monocular 3D object detection methods rely on point cloud data obtained with LiDAR scanners [20, 24, 25, 27, 30]. Since point clouds are very accurate they can effectively be used to estimate where objects are located in 3D space by considering the density of points. Additionally, [27] only requires the LiDAR data during training, which expands the use cases of the model. Pseudo LiDAR methods [26, 40] are hybrid methods where dense data is transformed into sparse LiDAR data, where LiDAR methods work directly.

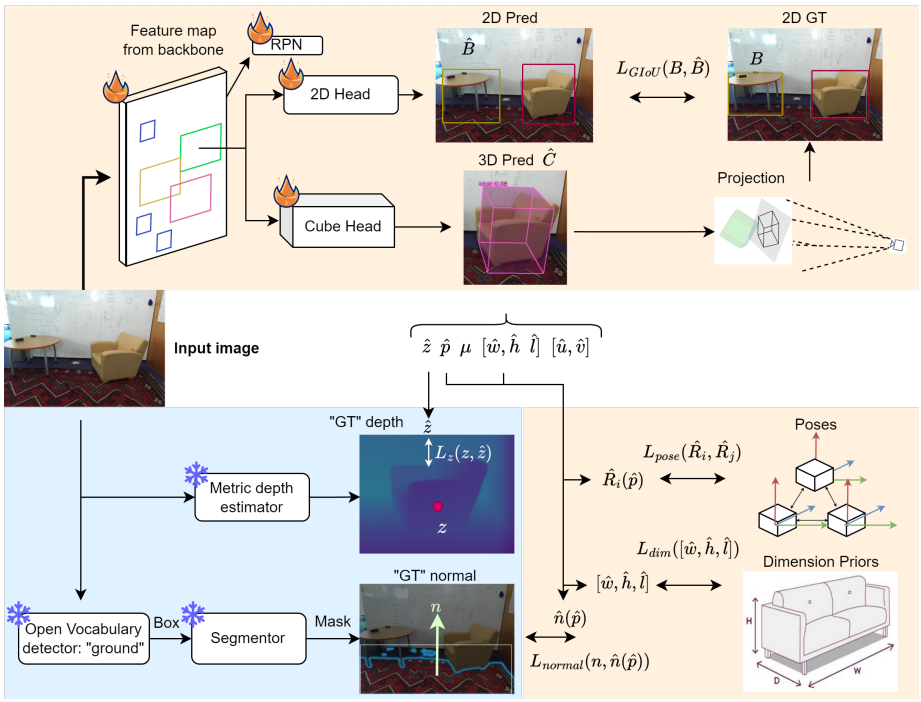

Fig. 2. Overview of *Weak Cube R-CNN*. The model extracts features from an image and predicts objects in 2D and their cubes in 3D. We split the cube into each of its attributes and optimise each attribute with regards to a pseudo ground truth information. During training, instead of the simple 3D ground truth provided in the fully supervised setting, we must use many different sources of information provided by frozen models to emulate the same ground truth annotation.

Incorporation of foundation models. Depth estimation models [2, 42] are ideal for use in downstream tasks. Instead of incorporating depth estimation directly into the main model, only the output is used, namely as pseudo ground truths [40]. [12] does exactly this, where the pseudo ground truth depth is used as a target for a depth estimation branch of the model. Furthermore, they use both 2D and 3D sub-networks and an uncertainty distillation loss to only learn from good predictions.

3 Method

In this section, we describe the modeling approach, structure, and optimization objective of the model. Overall, the model is a derivative of Cube R-CNN [3] and borrows the structure but without any direct 3D ground truths. Thus we name our method *Weak Cube R-CNN*. Fig. 2 provides an overview of the training procedure.

3.1 Overview

We use Cube R-CNN as our basis and therefore inherit all of the modeling techniques used, including virtual depth, IoUness, and allocentric rotation. Our method predicts 2D bounding boxes (bbox) with a Faster R-CNN-like architecture, based on a DLA-34 [43] FPN [19] backbone pre-trained on ImageNet. The 2D bboxes are used to pool features, with ROIAlignV2 [10]. The pooled features are used in a 3D head consisting of 2 linear layers, which predict a 3D cube expressed as 13 parameters corresponding to the image plane coordinates $[\hat{u}, \hat{v}]$, the depth $\hat{z}$, the dimensions $[\hat{w}, \hat{h}, \hat{l}]$, the 6D allocentric rotation [46] $\hat{p}$, and an uncertainty $\hat{\mu}$. A cube is predicted for each 2D box.

Due to a lack of access to the 3D ground truths, we carefully choose the appropriate sources that mimic a real 3D ground truth. When the depth is known, the physical size of an object can relatively easily be estimated based on the 2D detection and known camera intrinsics by using geometry. Finding the ground provides useful information on the rotation in a scene as it provides a frame of reference for all objects and constrains one axis of rotation. We use priors on the dimensions of objects to eliminate unrealistic cubes, though this constraint is relaxed to allow predictions to deviate from their prior sizes.

3.2 Obtaining Pseudo 3D Ground Truths

We use Depth-Anything V2 [42] fine-tuned for metric depth estimation. The model provides a depth value for each pixel in an image. A depth map offers a lot of structure in the image and we therefore use it for two downstream tasks: 1) Estimation of the ground plane in conjunction with a RANSAC algorithm and 2) The value on the depth map at the center point of the detected 2D bbox is the pseudo ground truth depth for 3D detections.

We use GroundingDINO [21] for ground detection by prompting it with the phrase "ground". This provides 2D bboxes that are passed into Segment Anything (SAM-HQ) [13,14] to get a segmentation of the ground. However, in some cases the ground is not visible or GroundingDINO fails for other reasons.

The depth map is interpreted as a point cloud by applying a simple transformation. Given the camera matrix K, we extract the focal length f and the center of the image, the principal point (c_x, c_y). Let (u, v, z) be image coordinates in pixels with depth z in meters. The conversion from the image to a set of points goes as follows: The offset of a point to the principal point is

$$\Delta u = u - c_x, \;\; \Delta v = v - c_y. \tag{1}$$

Let x, y, z be real-world coordinates. Each coordinate is calculated as

$$x = \frac{\Delta u \cdot z}{f}, \quad y = \frac{\Delta v \cdot z}{f}, \quad z = z. \tag{2}$$

This means that the point cloud P_{3D} is a set of all points

$$P_{3D} = (\mathbf{x}, \mathbf{y}, \mathbf{z}) \tag{3}$$

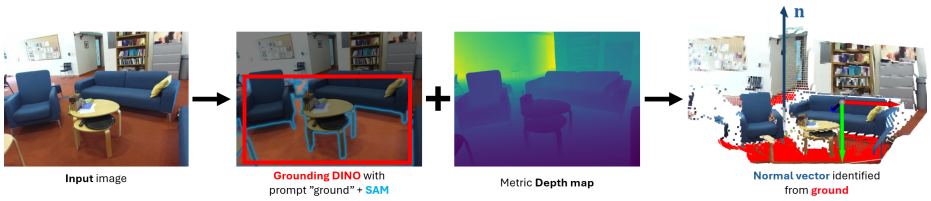

Fig. 3. Ground estimation pipeline showing the point cloud obtained through the depth map. The 2nd step selects the region in the depth map corresponding to the ground in the color image. The depth map is interpreted as a point cloud where plane-RANSAC obtains a normal vector to the ground.

The ground is detected using a simple plane RANSAC algorithm, which runs on the point cloud generated from the depth map. The algorithm finds the largest plane present in the point cloud. However, this is not always the actual floor, as it could also be a wall or any other random set of points roughly outlining a floor. This is why it is necessary to use GroundingDINO to filter the point cloud as seen in Fig. 3.

3.3 Loss functions

Our loss functions use only 2D labels, as this is the highest level of information available to us. This entails finding a relationship between a 3D cube and its corresponding 2D box. To relate 3D cubes to 2D boxes, we project 3D cubes to the 2D image plane using the camera intrinsics. For simplicity, we project to axis-aligned 2D boxes, so some of the rotation information is lost. Occluded objects are also not well handled because the projection will overlay the object on top of the image. To convert a 3D point to 2D:

$$P_{3D} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{bmatrix} \rightarrow P_{2D} = \frac{f}{\mathbf{z}} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \tag{4}$$

Training objective. The model is designed to handle each aspect of a cube independently. We consider the following aspects: the image plane placement, the depth, the dimensions, and the rotation.

Placement loss. For $[u, v]$ image-plane placement, we adopt the Generalised IoU loss [34] between the 2D projection of the 3D detection and the 2D bbox detection. The loss measures the 2D overlap between the projected 2D box $\hat{B}$ of the predicted 3D cube and the ground truth 2D box B.

$$L_{GIoU}(B, \hat{B}) = 1 - GIoU(B, \hat{B}) \tag{5}$$

This loss ensures that the placement of the 3D cube is correct and should align well with its 2D box.

Depth loss. For depth, we use the depth map as a pseudo ground truth z. $\hat{z}$ is a cubes' predicted depth by the 3D head. To obtain z, we use the center point

of the 2D bbox corresponding to its 3D detection to select a pixel in the depth map that has an associated depth to it. When the center point is outside the frame, the point is clamped to within 10 pixels of the image. We do this because the model severely underestimates the depth near the edges. The loss is the ℓ_1 distance between z and the predicted depth $\hat{z}$, see Fig. 2.

$$L_z(z, \hat{z}) = \|z - \hat{z}\|_1 \tag{6}$$

Prior size loss. To incorporate prior knowledge of the sizes of objects, we measure the z-score, *i.e.* how many standard deviations the object size is from the classes' mean size in each of the dimensions. These parameters are obtained for each class from the dataset or, if unavailable, by asking ChatGPT "average dimension of *object x*". For a given object dimension d we model the function acting as a loss:

$$\mathcal{Z} = \frac{1}{3} \sum_{d \in \{w,h,l\}} \left(\frac{|d - \mu_{prior}|}{\sigma_{prior}} \right) \tag{7}$$

$$L_{dim}(\hat{C}; \mu_{prior}, \sigma_{prior}) = \begin{cases} \mathcal{Z} & \text{if } \mathcal{Z} > 1 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Where μ_{prior} and σ_{prior} are the known mean and standard deviation dimensions of each class. This loss is a sum of z-scores for each dimension direction, a loss of 1 means the prediction is one standard deviation away from the prior of the specific class. The loss is overwritten with 0 when the loss is less than 1. This relaxation is necessary to prevent the model from learning to only predict mean sized objects. However, it makes the loss non-smooth which can make it unstable.

Normal vector loss. To ensure that the cubes are aligned with respect to the ground, we use the aforementioned method in Section 3.2 to identify the ground normal vector. We use the cosine similarity between the normal vector of the ground and the "up direction" (normal vector) of the cubes as a measure of alignment with the ground plane. The cosine similarity between two vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ is given by:

$$\cos_{sim}(\mathbf{n}_1, \mathbf{n}_2) = \frac{\mathbf{n}_1 \mathbf{n}_2}{\max(\|\mathbf{n}_1\|_2 \|\mathbf{n}_2\|_2, \epsilon)} \tag{9}$$

The cosine similarity is a number in the range $[0, 1]$, 1 when the vectors are identical and 0 when they are perpendicular, to turn the measure into a loss we want to reverse the relationship. We convert the 6D allocentric rotation $\hat{p}$ to a "up" normal vector of a cube $\hat{\mathbf{n}}(\hat{p})$. For a ground normal $\mathbf{n}$ and a predicted normal $\hat{\mathbf{n}}$, we define the loss:

$$L_{normal}(\mathbf{n}, \hat{\mathbf{n}}) = (1 - \cos_{sim}(\mathbf{n}, \hat{\mathbf{n}})) \cdot \kappa_{\text{ground}} \tag{10}$$

where κ_{ground} is the confidence in whether the ground was found, inspired by [3], that is 1 when the ground is visible and 0.05 otherwise. The model still

benefits from learning the placement and depth in these images, but does not get negatively affected by a poor rotation estimation.

Pose Alignment Loss. This loss ensures rotation consistency within all objects in a scene and relies on the assumption that objects typically are aligned with each other. To compute the alignment between two objects, we consider their rotation matrices. The trace of a rotation matrix R has the following relationship to the rotation angle θ:

$$Tr(R) = 1 + 2\cos(\theta) \tag{11}$$

We derive rotation matrices from the 6D representation $R(\hat{p})$, then calculate the relative angle between two rotation matrices R_1 and R_2, with the formula:

$$\cos(\theta) = \left| \frac{1}{2} \left(Tr(R_1 R_2^\top) - 1 \right) \right|. \tag{12}$$

The formula is an extension of Rodrigues' rotation formula. Since it is only possible to compare rotation matrices pairwise, we need all the unique combinations of all n instances in the image. This means we have to compute the angle for all $\frac{n(n-1)}{2}$ combinations. The number of instances in images varies greatly but is usually < 30. To not have images with many instances dominate a batch, we weight the loss inversely according to the number of objects in an image. To make a loss we flip Eq. (12):

$$L_{pose}(\theta) = 1 - \cos(\theta). \tag{13}$$

The loss is undefined for images with one instance because it doesn't make sense to align one object with itself.

Total 3D loss. Using the superscript to denote the component the loss concerns, the final 3D loss thus becomes

$$\begin{aligned} L_{3D} = \lambda_{GIoU} L_{GIoU}^{(u,v)} + \lambda_z L_z^{(z)} + \lambda_{dim} L_{dim}^{(\hat{w},\hat{h},\hat{l})} \\ + \lambda_{normal} L_{normal}^{(\hat{p})} + \lambda_{pose} L_{pose}^{(\hat{p})} \end{aligned} \tag{14}$$

The final training objective is

$$L = L_{RPN} + L_{2D} + \sqrt{2} e^{(-\mu)} \cdot L_{3D} + \mu \tag{15}$$

All terms, except L_{3D} are the losses from [3] and L_{3D} is the collective 3D loss as described in the sections above. μ is the uncertainty predicted by the model.

4 Experimental results

In this section, we present our experimental results, where we put special emphasis on achieving correct rotation.

Table 1. Weak Cube R-CNN ablations on SUN-RGBD. We report $AP_{3D}^{com.}$ on 10 "common categories" and on all 38 classes. Overall, not having L_z and L_{dim} significantly decreases performance.

Method	table	bed	sofa	bathtub	sink	shelves	cabinet	fridge	chair	tv	$AP_{3D}^{com.}$	AP_{3D}^{all}
w/ all (*Weak Cube R-CNN*)	12.3	26.4	20.5	20.5	10.4	1.2	6.8	5.3	18.8	5.4	**12.7**	5.4
w/o L_{GIoU}	12.3	13.1	19.8	10.9	7.7	1.4	2.6	4.2	19.6	2.9	9.5	4.4
w/o L_z	5.4	22.3	22.2	8.9	8.8	0.0	0.2	14.4	0.0	1.5	8.4	4.0
w/o L_{dim}	2.9	2.1	6.3	8.4	9.2	1.0	1.1	1.8	8.4	3.1	4.4	2.6
w/o L_{normal}	9.6	20.6	22.7	22.9	15.2	1.5	4.5	2.6	21.4	4.4	12.5	**5.7**
w/o L_{pose}	8.4	24.6	19.5	13.7	5.6	0.6	8.9	6.0	21.7	4.2	11.3	5.0

Table 2. Comparison on equal annotation time models. We report $AP_{3D}^{com.}$ on 10 "common categories". We train Cube R-CNN on SUN-RGBD based on their code. *Weak Cube R-CNN* outperforms *Cube R-CNN time eq.* overall and on all classes except table and chair, which are the 2 classes with the highest frequency in the dataset.

Method	table	bed	sofa	bathtub	sink	shelves	cabinet	fridge	chair	tv	$AP_{3D}^{com.}$	AP_{3D}^{all}
Cube R-CNN	39.8	64.4	60.0	38.3	27.4	3.1	14.1	21.6	53.5	3.8	32.6	15.1
Cube R-CNN time eq.	**13.4**	13.4	16.1	0.1	3.9	0.3	1.5	0.7	**24.2**	0.5	7.4	3.3
Weak Cube R-CNN	12.3	26.4	20.5	20.5	10.4	1.2	6.8	5.3	18.8	5.4	**12.7**	**5.4**

4.1 Setup

Datasets. We evaluate on the indoor SUN RGB-D 3D object detection dataset, which contains 10335 images, 5285 train and 5050 test images. Additionally, we use the outdoor KITTI 3D object detection dataset, with a total of 7481 images, 3712 for train and 3769 for test. We follow [3, 8] and remove objects with high occlusion ($> 66\%$), truncation ($> 33\%$) and with small projections ($< 6.25\%$ of image height). We do not use the full Omni3D dataset [3] due to computational constraints. We create SUN RGB-D mini and KITTI mini with 433 and 333 randomly selected images respectively, these datasets have equal annotation time to our method, *i.e.* 9% the size of the original data set.

Evaluation metric. The most common metric for both 2D and 3D object detection is the average precision (AP). We follow the Omni3D benchmark and use *mean* AP_{3D} as our evaluation metric. It averages over all classes for different levels of IoU3D at thresholds $\tau \in [0.05, 0.10, ..., 0.5]$.

Implementation details. Unless stated otherwise, we use the following setting for all models in this paper. We use Detectron2 and PyTorch3D [31] to implement *Weak-Cube R-CNN*. We train all models for 34 epochs with a batch size of 12 images on an A100 GPU. We use SGD with a learning rate of 0.007, which decays after 12 and 29 epochs by a factor of 10. Following [3], we use random data augmentation by horizontal flipping ($p = 0.5$) and scaling $\in [0.50, 1.25]$ during training. When indoors, we use Depth Anything V2 [42] fine-tuned for metric depth estimation on indoor scenes, with a max distance of $20m$, outdoors we use the outdoor model with a max of $80m$. For all our Weak Cube R-CNN models we use these loss weights: $\lambda_{GIoU} = 4$, $\lambda_z = 1$, $\lambda_{dim} = 0.1$, $\lambda_{normal} = 70$,

Fig. 4. Qualitative examples of *Weak Cube R-CNN* predictions on SUN-RGBD test set. Images are selected to showcase behaviour in various scenarios. Only the last row is shown with ground truths in red to avoid clutter. In the last row ground truths are shown in red with predictions in green. Each image is shown side-by-side with its corresponding top-view image, where each square is 1x1 m.

$\lambda_{pose} = 7$, which were found by trial and error mostly to get the magnitude of each term roughly equal. Especially of note is the low weight λ_{dim}.

Training Scheme. We only require the depth and ground estimator during training. Since we only use the output of frozen models, we opt to preprocess the dataset offline. We train our model in two stages. First, the model is fine-tuned in 2D mode only. This is done because we assume the 2D head is capable of finding the objects in the image. When the 2D head nearly converges, we switch on the 3D losses and train altogether. Our reasoning for this is that the 3D head synergises with the 2D head and improves the 2D IoU, compared to freezing the 2D head.

Baselines. We compare our model to the fully supervised Cube R-CNN [3]. We also implement a version of Cube R-CNN trained on the mini datasets, which is comparative in terms of annotation time to *Weak Cube R-CNN*. Based on 12 min per 3D annotated- and 65s per 2D annotated image.

4.2 Results

We run ablation experiments to study the impact of each loss term on the model. We show that the model performs well compared to the corresponding annotation time equalised model.

Ablations. Table 1 ablates the loss functions of *Weak Cube R-CNN*. We disclose AP_{3D} on the full dataset (all classes) and at a subset of 10 common classes. We

observe an increase in precision when adding a loss, most noticeably with L_{dim}, which improves AP_{3D} by +3.1%. We believe L_{dim} is important because the model has a tendency to quickly begin predicting strange cubes that it never gets away from, but L_{dim} effectively prevents this. We observed an initial large loss on L_{dim} which decreases quickly to 0. It also has a significant effect on objectively good cubes, as $AP^{\text{all}}_{\text{3D}}$ without it is closest to 0. L_{GIoU} also has an impact on good cubes, as $AP^{\text{com.}}_{\text{3D}}$ decreases by −3.2% without it. L_{pose} has the smallest effect, which intuitively makes sense as this loss enforces matching but not necessarily correct rotation.

It stands out that removing L_z does not have the greatest impact on performance, especially for better cubes. When examining predictions, many look much better in 2D than 3D, but still have low IoU. That is mainly due to incorrect depth. As such, we must conclude that L_z does not work perfectly as a proxy for a true depth loss, yet still has a positive effect on overall predictions.

The ablations in Table 1 contradict our assumption that the ground provides a rotation frame for a scene. To validate our approach, we further test on a subset of classes to see if it really is better to include L_{normal}. Furthermore, we provide more experiments on KITTI, Table 3, which shows that the model with L_{normal} is better in terms of rotation. This is an indication that L_{normal} is indeed having the intended effect. Because we want to focus on ensuring correct rotation we thus include L_{normal} in the final model.

Comparison to other models. Against the fully supervised Cube R-CNN method, *Weak-Cube R-CNN* achieves about 1/3 of the performance as presented in Table 2. AP3D sees a drop from 15.1% to 5.4%, when evaluating on all classes.

Comparison with equal annotation time. The goal of *Weak-Cube R-CNN* is to cut down on annotation time. We show results with our method compared to Cube R-CNN when using fully annotated 3D data but trained on SUN RGB-D mini. Furthermore, we present results on a narrow selection of classes. Table 2 demonstrates that our method exceeds baseline performance in many categories and achieves +5.3% mean AP compared to the time equalised Cube R-CNN on the reduced set of classes. It is clear that Cube R-CNN time eq. is greatly held back by very few samples of certain classes like "bathtub" where it does not learn to detect anything meaningful. Unsurprisingly, Cube R-CNN outperforms both models in all categories. However, considering the annotation time is about 11x more it does not achieve 11x the performance vs. *Weak-Cube R-CNN*.

Qualitative results. When looking at the qualitative results in Fig. 4 we find that for indoor scenes the predictions are generally clearer for simple scenes. In the scenes with many objects and occluded objects it generally struggles. Overall, depth seems to be quite accurately predicted which we can see in the last row that is shown with ground truths. It seems that pose alignment does improve detection accuracy overall but makes the harder cases harder. Objects, like cabinets, that are not rooted in the ground are nearly all detected poorly.

Comparison on KITTI. The advantage that KITTI provides compared to indoor data sets is that the ground is much more consistently visible, and thus, we expect rotation to be easier to determine. For outdoor scenes, the model

Table 3. Ablation results on KITTI (with all classes), we report the mean AP_{3D}, and at thresholds $0.15, 0.25$, and 0.5. We observe that using L_{normal} improves the precision considerably when the ground is clearly visible on outdoor images.

Method	AP_{3D}	AP^{15}_{3D}	AP^{25}_{3D}	AP^{50}_{3D}
w/ L_{normal}	**8.2**	**12.1**	**8.3**	**2.0**
w/o L_{normal}	6.3	10.9	6.8	0.4

Table 4. Performance with f = fully supervised methods on KITTI.

Method	f	AP^{KITTI}_{3D}
Cube R-CNN [3]	✓	36.0
SMOKE [22]	✓	25.4
ImVoxelNet [36]	✓	23.5
M3D-RPN [4]	✓	10.4
Cube R-CNN time-eq.	✓	16.4
Weak Cube R-CNN	✗	8.2

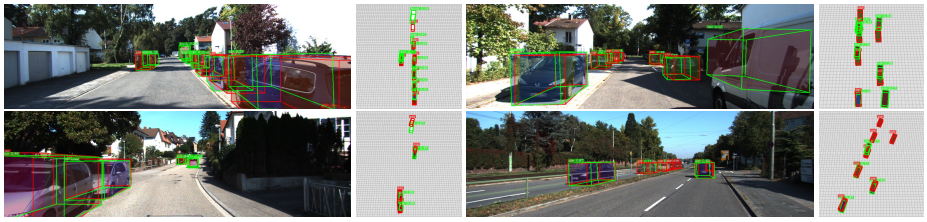

Fig. 5. Qualitative examples of *Weak Cube R-CNN* predictions on KITTI test set. KITTI predictions are shown in green and ground truth in red. Each image is shown with its corresponding top-view image, where each square is 1x1 m.

generally provides excellent predictions in the front view, as shown in Fig. 5. However, in the top view, it is seen that depth is often wrong. Table 4 Shows that *Weak Cube R-CNN* comes close to achieving the same precision as the older fully supervised method M3D-RPN, but even the time equalized Cube R-CNN beats it by 2x. The primary difference between KITTI and SUN RGB-D is the depth, which appears to be what *Weak Cube R-CNN* struggles with the most.

5 Conclusion

We have proposed *Weak Cube R-CNN*, a novel approach for 3D object detection that relies solely on single-view images and 2D image annotations. Our method overcomes the most prominent limitation of 3D object detection, which is the annotation availability of datasets, by leveraging weaker supervision while still achieving competitive performance. *Weak Cube R-CNN* demonstrates strong detection capabilities for objects with high visibility and simple geometric structures. Notably, on the SUN-RGBD dataset, given the same annotation time, it achieves better performance than a fully supervised Cube R-CNN model trained with 3D annotation bounding box annotations. Further work could involve exploring other weak signals, as the method is slow to converge due to the weak signals carried through the loss functions.

Acknowledgements. D. Papadopoulos was supported by the DFF Sapere Aude Starting Grant "ACHILLES".

References

1. Baruch, G., et al.: Arkitscenes - a dataset for 3d indoor scene understanding using mobile rgb-d data. In: NeurIPS (2021)
2. Bochkovskii, A., Delaunoy, A., Germain, H., Santos, M., Zhou, Y., Richter, S.R., Koltun, V.: Depth pro: Sharp monocular metric depth in less than a second. arXiv:2410.02073 (2024)
3. Brazil, G., Kumar, A., Straub, J., Ravi, N., Johnson, J., Gkioxari, G.: Omni3d: A large benchmark and model for 3d object detection in the wild. In: CVPR (2023)
4. Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: CVPR (2019)
5. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR (2020)
6. Chen, Y., Liu, S., Shen, X., Jia, J.: Dsgn: Deep stereo geometry network for 3d object detection. In: CVPR (2020)
7. Fan, L., Xiong, X., Wang, F., Wang, N., Zhang, Z.: Rangedet: In defense of range view for lidar-based 3d object detection. In: CVPR (2021)
8. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR (2012)
9. He, J., Wang, Y., Chen, Y., Zhang, Z.: Weakly supervised 3d object detection with multi-stage generalization (2024)
10. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)
11. Huang, K.C., Wu, T.H., Su, H.T., Hsu, W.H.: Monodtr. In: CVPR (2022)
12. Jiang, X., Jin, S., Lu, L., Zhang, X., Lu, S.: Weakly supervised monocular 3d detection with a single-view image. In: CVPR (2024)
13. Ke, L., Ye, M., Danelljan, M., Liu, Y., Tai, Y.W., Tang, C.K., Yu, F.: Segment anything in high quality. NeurIPS (2023)
14. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. In: CVPR (2023)
15. Ku, J., Pon, A.D., Waslander, S.L.: Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In: CVPR (2019)
16. Kumar, A., Brazil, G., Corona, E., Parchami, A., Liu, X.: Deviant: Depth equivariant network for monocular 3d object detection. In: ECCV (2022)
17. Li, P., Chen, X., Shen, S.: Stereo r-cnn based 3d object detection for autonomous driving. In: CVPR (2019)
18. Li, Z., Qu, Z., Zhou, Y., Liu, J., Wang, H., Jiang, L.: Diversity matters: Fully exploiting depth clues for reliable monocular 3d object detection. In: CVPR (2022)
19. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR (2017)
20. Liu, H., Ma, H., Wang, Y., Zou, B., Hu, T., Wang, R., Chen, J.: Eliminating spatial ambiguity for weakly supervised 3d object detection without spatial labels. In: ACM (2022)
21. Liu, S., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In: ECCV (2024)
22. Liu, Z., Wu, Z., Tóth, R.: Smoke: Single-stage monocular 3d object detection via keypoint estimation. In: CVPR workshops (2020)
23. Liu, Z., Zhou, D., Lu, F., Fang, J., Zhang, L.: Autoshape: Real-time shape-aware monocular 3d object detection. In: CVPR (2021)

24. Meng, Q., Wang, W., Zhou, T., Shen, J., Van Gool, L., Dai, D.: Weakly supervised 3d object detection from lidar point cloud. In: ECCV (2020)
25. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry (2017)
26. Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: CVPR (2021)
27. Peng, L., Yan, S., Wu, B., Yang, Z., He, X., Cai, D.: Weakm3d: Towards weakly supervised monocular 3d object detection. arXiv preprint arXiv:2203.08332 (2022)
28. Pon, A.D., Ku, J., Li, C., Waslander, S.L.: Object-centric stereo matching for 3d object detection. In: ICRA (2020)
29. Qian, R., Garg, D., Wang, Y., You, Y., Belongie, S., Hariharan, B., Campbell, M., Weinberger, K.Q., Chao, W.L.: End-to-end pseudo-lidar for image-based 3d object detection. In: CVPR (2020)
30. Qin, Z., Wang, J., Lu, Y.: Weakly supervised 3d object detection from point clouds. In: ACM (2020)
31. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Pytorch3d. arXiv:2007.08501 (2020)
32. Reading, C., Harakeh, A., Chae, J., Waslander, S.L.: Categorical depth distribution network for monocular 3d object detection. In: CVPR (2021)
33. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. NeurIPS (2015)
34. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: WACV (2019)
35. Roberts, M., Ramapuram, J., Ranjan, A., Kumar, A., Bautista, M.A., Paczan, N., Webb, R., Susskind, J.M.: Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In: CVPR (2021)
36. Rukhovich, D., Vorontsova, A., Konushin, A.: Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3dod. In: WACV (2022)
37. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: CVPR (2015)
38. Tao, R., Han, W., Qiu, Z., Xu, C.z., Shen, J.: Weakly supervised monocular 3dod using multi-view projection and direction consistency. In: CVPR (2023)
39. Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: CVPR (2021)
40. Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: CVPR (2019)
41. Xia, C., Zhao, W., Han, H., Tao, Z., Ge, B., Gao, X., Li, K.C., Zhang, Y.: Monosaid: Monocular 3d object detection based on scene-level adaptive instance depth estimation. Journal of Intelligent & Robotic Systems (2024)
42. Yang, L., Kang, B., Huang, Z., Zhao, Z., Xu, X., Feng, J., Zhao, H.: Depth anything v2 (2024)
43. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: CVPR (2018)
44. Zhang, R., Qiu, H., Wang, T., Guo, Z., Cui, Z., Qiao, Y., Li, H., Gao, P.: Monodetr. In: CVPR (2023)
45. Zhang, Y., Lu, J., Zhou, J.: Objects are different: Flexible monocular 3d object detection. In: CVPR (2021)
46. Zhou, Y., Barnes, C., Lu, J., Yang, J., Li, H.: On the continuity of rotation representations in neural networks. In: CVPR (2019)