

3D Bounding Box Estimation Using Deep Learning and Geometry

Arsalan Mousavian*
 George Mason University
 amousavi@gmu.edu

Dragomir Anguelov
 Zoox, Inc.
 drago@zoox.com

John Flynn
 Zoox, Inc.
 john.flynn@zoox.com

Jana Košecká
 George Mason University
 kosecka@gmu.edu

Abstract

We present a method for 3D object detection and pose estimation from a single image. In contrast to current techniques that only regress the 3D orientation of an object, our method first regresses relatively stable 3D object properties using a deep convolutional neural network and then combines these estimates with geometric constraints provided by a 2D object bounding box to produce a complete 3D bounding box. The first network output estimates the 3D object orientation using a novel hybrid discrete-continuous loss, which significantly outperforms the L2 loss. The second output regresses the 3D object dimensions, which have relatively little variance compared to alternatives and can often be predicted for many object types. These estimates, combined with the geometric constraints on translation imposed by the 2D bounding box, enable us to recover a stable and accurate 3D object pose. We evaluate our method on the challenging KITTI object detection benchmark [2] both on the official metric of 3D orientation estimation and also on the accuracy of the obtained 3D bounding boxes. Although conceptually simple, our method outperforms more complex and computationally expensive approaches that leverage semantic segmentation, instance level segmentation and flat ground priors [4] and sub-category detection [23][24]. Our discrete-continuous loss also produces state of the art results for 3D viewpoint estimation on the Pascal 3D+ dataset[26].

1. Introduction

The problem of 3D object detection is of particular importance in robotic applications that require decision making or interactions with objects in the real world. 3D object detection recovers both the 6 DoF pose and the dimen-

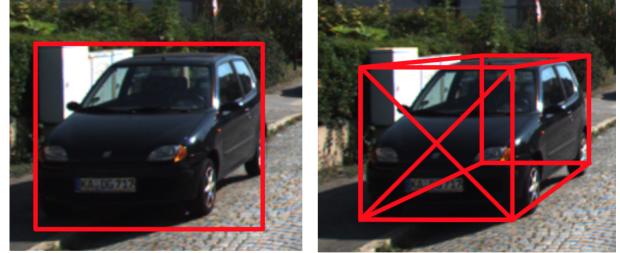


Figure 1. Our method takes the 2D detection bounding box and estimates a 3D bounding box.

sions of an object from an image. While recently developed 2D detection algorithms are capable of handling large variations in viewpoint and clutter, accurate 3D object detection largely remains an open problem despite some promising recent work. The existing efforts to integrate pose estimation with state-of-the-art object detectors focus mostly on viewpoint estimation. They exploit the observation that the appearance of objects changes as a function of viewpoint and that discretization of viewpoints (parametrized by azimuth and elevation) gives rise to sub-categories which can be trained discriminatively [23]. In more restrictive driving scenarios alternatives to full 3D pose estimation explore exhaustive sampling and scoring of all hypotheses [4] using a variety of contextual and semantic cues.

In this work, we propose a method that estimates the pose $(R, T) \in SE(3)$ and the dimensions of an object's 3D bounding box from a 2D bounding box and the surrounding image pixels. Our simple and efficient method is suitable for many real world applications including self-driving vehicles. The main contribution of our approach is in the choice of the regression parameters and the associated objective functions for the problem. We first regress the orientation and object dimensions before combining these estimates with geometric constraints to produce a final 3D pose. This is in contrast to previous techniques that attempt

*Work done as an intern at Zoox, Inc.

to directly regress to pose.

A state of the art 2D object detector [3] is extended by training a deep convolutional neural network (CNN) to regress the orientation of the object’s 3D bounding box and its dimensions. Given estimated orientation and dimensions and the constraint that the projection of the 3D bounding box fits tightly into the 2D detection window, we recover the translation and the object’s 3D bounding box. Although conceptually simple, our method is based on several important insights. We show that a novel *MultiBin* discrete-continuous formulation of the orientation regression significantly outperforms a more traditional L2 loss. Further constraining the 3D box by regressing to vehicle dimensions proves especially effective, since they are relatively low-variance and result in stable final 3D box estimates.

We evaluate our method on the KITTI [2] and Pascal 3D+[26] datasets. On the KITTI dataset, we perform an in-depth comparison of our estimated 3D boxes to the results of other state-of-the-art 3D object detection algorithms [24, 4]. The official KITTI benchmark for 3D bounding box estimation only evaluates the 3D box orientation estimate. We introduce three additional performance metrics measuring the 3D box accuracy: distance to center of box, distance to the center of the closest bounding box face, and the overall bounding box overlap with the ground truth box, measured using 3D Intersection over Union (3D IoU) score. We demonstrate that given sufficient training data, our method is superior to the state of the art on all the above 3D metrics. Since the Pascal 3D+ dataset does not have the physical dimensions annotated and the intrinsic camera parameters are approximate, we only evaluate viewpoint estimation accuracy showing that our *MultiBin* module achieves state-of-the-art results there as well.

In summary, the main contributions of our paper include: 1) A method to estimate an object’s full 3D pose and dimensions from a 2D bounding box using the constraints provided by projective geometry and estimates of the object’s orientation and size regressed using a deep CNN. In contrast to other methods, our approach does not require any preprocessing stages or 3D object models. 2) A novel discrete-continuous CNN architecture called *MultiBin* regression for estimation of the object’s orientation. 3) Three new metrics for evaluating 3D boxes beyond their orientation accuracy for the KITTI dataset. 4) An experimental evaluation demonstrating the effectiveness of our approach for KITTI cars, which also illustrates the importance of the specific choice of regression parameters within our 3D pose estimation framework. 5) Viewpoint evaluation on the Pascal 3D+ dataset.

2. Related Work

The classical problem of 6 DoF pose estimation of an object instance from a single 2D image has been consid-

ered previously as a purely geometric problem known as the *perspective n-point problem (PnP)*. Several closed form and iterative solutions assuming correspondences between 2D keypoints in the image and a 3D model of the object can be found in [10] and references therein. Other methods focus on constructing 3D models of the object instances and then finding the 3D pose in the image that best matches the model [19, 6].

With the introduction of new challenging datasets [2, 26, 25, 12], 3D pose estimation has been extended to object categories, which requires handling both the appearance variations due to pose changes and the appearance variations within the category [9, 15]. In [16, 26] the object detection framework of discriminative part based models (DPMs) is used to tackle the problem of pose estimation formulated jointly as a structured prediction problem, where each mixture component represents a different azimuth section. However, such approaches predict only an Euler angle subset with respect to the canonical object frame, while object dimensions and position are not estimated.

An alternative direction is to exploit the availability of 3D shape models and use those for 3D hypothesis sampling and refinement. For example, Mottaghi *et al.* [13] sample the object viewpoint, position and size and then measure the similarity between rendered 3D CAD models of the object and the detection window using HOG features. A similar method for estimating the pose using the projection of CAD model object instances has been explored by [29] in a robotics table-top setting where the detection problem is less challenging. Given the coarse pose estimate obtained from a DPM-based detector, the continuous 6 DoF pose is refined by estimating the correspondences between the projected 3D model and the image contours. The evaluation was carried out on PASCAL3D+ or simple table top settings with limited clutter or scale variations. An extension of these methods to more challenging scenarios with significant occlusion has been explored in [22], which uses dictionaries of 3D voxel patterns learned from 3D CAD models that characterize both the object’s shape and commonly encountered occlusion patterns.

Recently, deep convolutional neural networks (CNN) have dramatically improved the performance of 2D object detection and several extensions have been proposed to include 3D pose estimation. In [21] R-CNN [7] is used to detect objects and the resulting detected regions are passed as input to a pose estimation network. The pose network is initialized with VGG [20] and fine-tuned for pose estimation using ground truth annotations from Pascal 3D+. This approach is similar to [8], with the distinction of using separate pose weights for each category and a large number of synthetic images with pose annotation ground truth for training. In [17], Poirson *et al.* discretize the object viewpoint and train a deep convolutional network to jointly

perform viewpoint estimation and 2D detection. The network shares the pose parameter weights across all classes. In [21], Tulsiani *et al.* explore the relationship between coarse viewpoint estimation, followed by keypoint detection, localization and pose estimation. Pavlakos et al [14], used CNN to localize the keypoints and they used the keypoints and their 3D coordinates from meshes to recover the pose. However, their approach required training data with annotated keypoints.

Several recent methods have explored 3D bounding box detection for driving scenarios and are most closely related to our method. Xiang *et al.* [23, 24] cluster the set of possible object poses into viewpoint-dependent subcategories. These subcategories are obtained by clustering 3D voxel patterns introduced previously [22]; 3D CAD models are required to learn the pattern dictionaries. The subcategories capture both shape, viewpoint and occlusion patterns and are subsequently classified discriminatively [24] using deep CNNs. Another related approach by Chen *et al.* [4] addresses the problem by sampling 3D boxes in the physical world assuming the flat ground plane constraint. The boxes are scored using high level contextual, shape and category specific features. All of the above approaches require complicated preprocessing including high level features such as segmentation or 3D shape repositories and may not be suitable for robots with limited computational resources.

3. 3D Bounding Box Estimation

In order to leverage the success of existing work on 2D object detection for 3D bounding box estimation, we use the fact that the perspective projection of a 3D bounding box should fit tightly within its 2D detection window. We assume that the 2D object detector has been trained to produce boxes that correspond to the bounding box of the projected 3D box. The 3D bounding box is described by its center $T = [t_x, t_y, t_z]^T$, dimensions $D = [d_x, d_y, d_z]$, and orientation $R(\theta, \phi, \alpha)$, here parameterized by the azimuth, elevation and roll angles. Given the pose of the object in the camera coordinate frame $(R, T) \in SE(3)$ and the camera intrinsics matrix K , the projection of a 3D point $\mathbf{X}_o = [X, Y, Z, 1]^T$ in the object's coordinate frame into the image $\mathbf{x} = [x, y, 1]^T$ is:

$$\mathbf{x} = K [R \ T] \mathbf{X}_o \quad (1)$$

Assuming that the origin of the object coordinate frame is at the center of the 3D bounding box and the object dimensions D are known, the coordinates of the 3D bounding box vertices can be described simply by $\mathbf{X}_1 = [d_x/2, d_y/2, d_z/2]^T$, $\mathbf{X}_2 = [-d_x/2, d_y/2, d_z/2]^T$, ..., $\mathbf{X}_8 = [-d_x/2, -d_y/2, -d_z/2]^T$. The constraint that the 3D bounding box fits tightly into 2D detection window requires that each side of the 2D bounding box to be touched

by the projection of at least one of the 3D box corners. For example, consider the projection of one 3D corner $\mathbf{X}_0 = [d_x/2, -d_y/2, d_z/2]^T$ that touches the left side of the 2D bounding box with coordinate x_{min} . This point-to-side correspondence constraint results in the equation:

$$x_{min} = \left(K [R \ T] \begin{bmatrix} d_x/2 \\ -d_y/2 \\ d_z/2 \\ 1 \end{bmatrix} \right)_x \quad (2)$$

where $(.)_x$ refers to the x coordinate from the perspective projection. Similar equations can be derived for the remaining 2D box side parameters $x_{max}, y_{min}, y_{max}$. In total the sides of the 2D bounding box provide four constraints on the 3D bounding box. This is not enough to constrain the nine degrees of freedom (DoF) (three for translation, three for rotation, and three for box dimensions). There are several different geometric properties we could estimate from the visual appearance of the box to further constrain the 3D box. The main criteria is that they should be tied strongly to the visual appearance and further constrain the final 3D box.

3.1. Choice of Regression Parameters

The first set of parameters that have a strong effect on the 3D bounding box is the orientation around each axis (θ, ϕ, α) . Apart from them, we choose to regress the box dimensions D rather than translation T because the variance of the dimension estimate is typically smaller (e.g. cars tend to be roughly the same size) and does not vary as the object orientation changes: a desirable property if we are also regressing orientation parameters. Furthermore, the dimension estimate is strongly tied to the appearance of a particular object subcategory and is likely to be accurately recovered if we can classify that subcategory. In Sec. 5.4 we carried out experiments on regressing alternative parameters related to translation and found that choice of parameters matters: we obtained less accurate 3D box reconstructions using that parametrization. The CNN architecture and the associated loss functions for this regression problem are discussed in Sec. 4.

3.2. Correspondence Constraints

Using the regressed dimensions and orientations of the 3D box by CNN and 2D detection box we can solve for the translation T that minimizes the reprojection error with respect to the initial 2D detection box constraints in Equation 2. Details of how to solve for translation are included in the supplementary material [1]. Each side of the 2D detection box can correspond to any of the eight corners of the 3D box which results in $8^4 = 4096$ configurations. Each different configuration involves solving an over-constrained system of linear equations which is computationally fast and

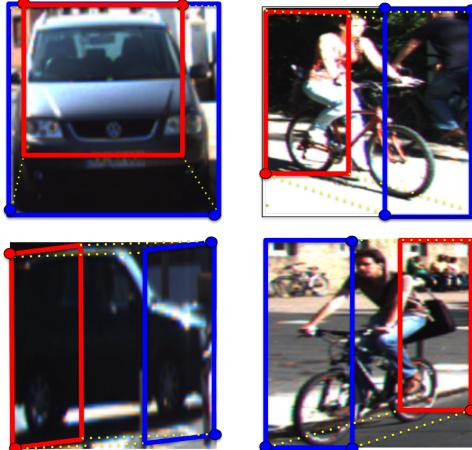


Figure 2. Correspondence between the 3D box and 2D bounding box: Each figure shows a 3D bbox that surrounds an object. The front face is shown in blue and the rear face is in red. The 3D points that are active constraints in each of the images are shown with a circle (best viewed in color).

can be done in parallel. In many scenarios the objects can be assumed to be always upright. In this case, the 2D box top and bottom correspond only to the projection of vertices from the top and bottom of the 3D box, respectively, which reduces the number of correspondences to 1024. Furthermore, when the relative object roll is close to zero, the vertical 2D box side coordinates x_{min} and x_{max} can only correspond to projections of points from vertical 3D box sides. Similarly, y_{min} and y_{max} can only correspond to point projections from the horizontal 3D box sides. Consequently, each vertical side of the 2D detection box can correspond to $[\pm d_x/2, \dots, \pm d_z/2]$ and each horizontal side of the 2D bounding box corresponds to $[., \pm d_y/2, \pm d_z/2]$, yielding $4^4 = 256$ possible configurations. In the KITTI dataset, object pitch and roll angles are both zero, which further reduces the number of configurations to 64. Fig. 2 visualizes some of the possible correspondences between 2D box sides and 3D box points that can occur.

4. CNN Regression of 3D Box Parameters

In this section, we describe our approach for regressing the 3D bounding box orientation and dimensions.

4.1. MultiBin Orientation Estimation

Estimating the global object orientation $R \in SO(3)$ in the camera reference frame from only the contents of the detection window crop is not possible, as the location of the crop within the image plane is also required. Consider the rotation $R(\theta)$ parametrized only by azimuth θ (yaw). Fig. 4 shows an example of a car moving in a straight line. Although the global orientation $R(\theta)$ of the car (its 3D bounding box) does not change, its local orientation θ_l with re-

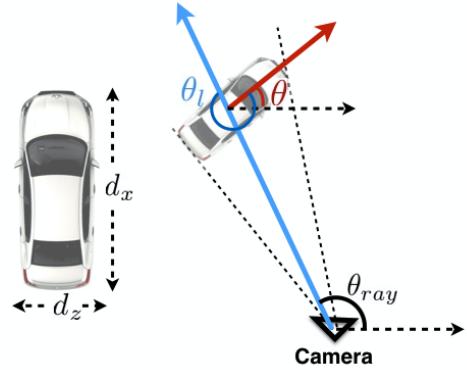


Figure 3. Left: Car dimensions, the height of the car equals d_y . Right: Illustration of local orientation θ_l , and global orientation of a car θ . The local orientation is computed with respect to the ray that goes through the center of the crop. The center ray of the crop is indicated by the blue arrow. Note that the center of crop may not go through the actual center of the object. Orientation of the car θ is equal to $\theta_{ray} + \theta_l$. The network is trained to estimate the local orientation θ_l .



Figure 4. Left: cropped image of a car passing by. Right: Image of whole scene. As it is shown the car in the cropped images rotates while the car direction is constant among all different rows.

spect to the ray through the crop center does, and generates changes in the appearance of the cropped image.

We thus regress to this local orientation θ_l . Fig. 4 shows an example, where the local orientation angle θ_l and the ray angle change in such a way that their combined effect is a constant global orientation of the car. Given intrinsic camera parameters, the ray direction at a particular pixel is trivial to compute. At inference time we combine this ray direction at the crop center with the estimated local orientation in order to compute the global orientation of the object.

It is known that using the L2 loss is not a good fit for many complex multi-modal regression problems. The L2 loss encourages the network to minimize to average loss

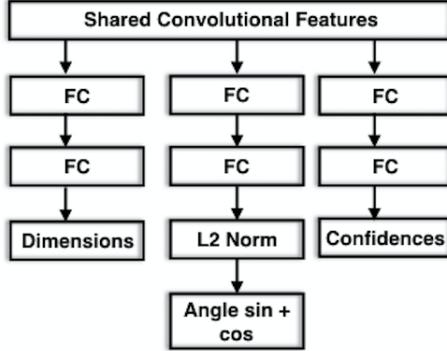


Figure 5. Proposed architecture for MultiBin estimation for orientation and dimension estimation. It consists of three branches. The left branch is for estimation of dimensions of the object of interest. The other branches are for computing the confidence for each bin and also compute the $\cos(\Delta\theta)$ and $\sin(\Delta\theta)$ of each bin

across all modes, which results in an estimate that may be poor for any single mode. This has been observed in the context of the image colorization problem, where the L2 norm produces unrealistic average colors for items like clothing [27]. Similarly, object detectors such as Faster R-CNN [18] and SSD [11] do not regress the bounding boxes directly: instead they divide the space of the bounding boxes into several discrete modes called *anchor boxes* and then estimate the continuous offsets that need to be applied to each anchor box.

We use a similar idea in our proposed *MultiBin* architecture for orientation estimation. We first discretize the orientation angle and divide it into n overlapping bins. For each bin, the CNN network estimates both a confidence probability c_i that the output angle lies inside the i^{th} bin and the residual rotation correction that needs to be applied to the orientation of the center ray of that bin in order to obtain the output angle. The residual rotation is represented by two numbers, for the sine and the cosine of the angle. This results in 3 outputs for each bin i : $(c_i, \cos(\Delta\theta_i), \sin(\Delta\theta_i))$. Valid cosine and sine values are obtained by applying an L2 normalization layer on top of a 2-dimensional input. The total loss for the MultiBin orientation is thus:

$$L_\theta = L_{conf} + w \times L_{loc} \quad (3)$$

The confidence loss L_{conf} is equal to the softmax loss of the confidences of each bin. L_{loc} is the loss that tries to minimize the difference between the estimated angle and the ground truth angle in each of the bins that covers the ground truth angle, with adjacent bins having overlapping coverage. In the localization loss L_{loc} , all the bins that cover the ground truth angle are forced to estimate the correct angle. The localization loss tries to minimize the difference between the ground truth and all the bins that cover that value which is equivalent of maximizing cosine distance as

it is shown in supplementary material [1]. Localization loss L_{loc} is computed as following:

$$L_{loc} = -\frac{1}{n_{\theta^*}} \sum \cos(\theta^* - c_i - \Delta\theta_i) \quad (4)$$

where n_{θ^*} is the number of bins that cover ground truth angle θ^* , c_i is the angle of the center of bin i and $\Delta\theta_i$ is the change that needs to be applied to the center of bin i .

During inference, the bin with maximum confidence is selected and the final output is computed by applying the estimated $\Delta\theta$ of that bin to the center of that bin. The *MultiBin* module has 2 branches. One for computing the confidences c_i and the other for computing the cosine and sine of $\Delta\theta$. As a result, $3n$ parameters need to be estimated for n bins.

In the KITTI dataset cars, vans, trucks, and buses are all different categories and the distribution of the object dimensions for category instances is low-variance and unimodal. For example, the dimension variance for cars and cyclists is on the order of several centimeters. Therefore, rather than using a discrete-continuous loss like the *MultiBin* loss above, we use directly the L2 loss. As is standard, for each dimension we estimate the residual relative to the mean parameter value computed over the training dataset. The loss for dimension estimation L_{dims} is computed as follows:

$$L_{dims} = \frac{1}{n} \sum (D^* - \bar{D} - \delta)^2, \quad (5)$$

where D^* are the ground truth dimensions of the box, \bar{D} are the mean dimensions for objects of a certain category and δ is the estimated residual with respect to the mean that the network predicts.

The CNN architecture of our parameter estimation module is shown in Figure 5. There are three branches: two branches for orientation estimation and one branch for dimension estimation. All of the branches are derived from the same shared convolutional features and the total loss is the weighted combination of $L = \alpha \times L_{dims} + L_\theta$.

5. Experiments and Discussions

5.1. Implementation Details

We performed our experiments on the KITTI [2] and Pascal 3D+[26] datasets.

KITTI dataset: The KITTI dataset has a total of 7481 training images. We train the MS-CNN [3] object detector to produce 2D boxes and then estimate 3D boxes from 2D detection boxes whose scores exceed a threshold. For regressing 3D parameters, we use a pretrained VGG network [20] without its FC layers and add our 3D box module, which is shown in Fig. 5. In the module, the first FC layers in each of the orientation branches have 256 dimensions,

while the first FC layer for dimension regression has a dimension of 512. During training, each ground truth crop is resized to 224x224. In order to make the network more robust to viewpoint changes and occlusions, the ground truth boxes are jittered and the ground truth θ_l is changed to account for the movement of the center ray of the crop. In addition, we added color distortions and also applied mirroring to images at random. The network is trained with SGD using a fixed learning rate of 0.0001. The training is run for 20K iterations with a batch size of 8 and the best model is chosen by cross validation. Fig. 6 shows the qualitative visualization of estimated 3D boxes for cars and cyclists on our KITTI validation set. We used two different training/test splits for our experiments. The first split was used to report results on the official KITTI test set and uses the majority of the available training images. The second split is identical to the one used by SubCNN [24], in order to enable fair comparisons. They use half of the available data for validation.

Pascal3D+ dataset: The dataset consists of images from Pascal VOC and Imagenet for 12 different categories that are annotated with 6 DoF pose. Images from the Pascal training set and Imagenet are used for training and the evaluation is done on the Pascal validation set. Unlike KITTI, the intrinsic parameters are approximate and therefore it is not possible to recover the true physical object dimensions. Therefore we only evaluate on 3 DoF viewpoint estimation to show the effectiveness of our *MultiBin* loss. We used $C \times 3$ *MultiBin* modules to predict 3 angles for each of the C classes. For a fair comparison with [21], we kept the *fc6* and *fc7* layers of VGG and eliminated the separate convolution branches of our *MultiBin* modules. All the necessary inputs are generated using a single fully connected layer that takes *fc7* as input. We also reused the hyperparameters chosen in [21] for training our model.

5.2. 3D Bounding Box Evaluation

KITTI orientation accuracy. The official 3D metric of the KITTI dataset is Average Orientation Similarity (AOS), which is defined in [2] and multiplies the average precision (AP) of the 2D detector with the average cosine distance similarity for azimuth orientation. Hence, AP is by definition the upper bound of AOS. At the time of publication, we are first among all methods in terms of AOS for easy car examples and first among all non-anonymous methods for moderate car examples on the KITTI leaderboard. Our results are summarized in Table 1, which shows that we outperform all the recently published methods on orientation estimation for cars. For moderate cars we outperform SubCNN [24] despite having similar AP, while for hard examples we outperform 3DOP [5] despite much lower AP. The ratio of AOS over AP for each method is representative of how each method performs only on orientation estima-

tion, while factoring out the 2D detector performance. We refer to this score as Orientation Score (OS), which represents the error $(1 + \cos(\Delta\theta))/2$ averaged across all examples. OS can be converted back to angle error by the $\text{acos}(2 * OS - 1)$ formula, resulting in 3° error for easy, 6° for moderate, and 8° on hard cars for our *MultiBin* model on the official KITTI test set. Our method is the only one that does not rely on computing additional features such as stereo, semantic segmentation, instance segmentation and does not need preprocessing as in [24] and [23].

Pascal3D+ viewpoint accuracy. Two metrics are used for viewpoint accuracy: Median Error *MedErr* and the percentage of the estimations that are within $\frac{\pi}{6}$ of the groundtruth viewpoint *Acc $\frac{\pi}{6}$* . The distance between rotations is computed as $\Delta(R_1, R_2) = \frac{\|\log(R_1^T R_2)\|_F}{\sqrt{2}}$. The evaluation is done using the groundtruth bounding boxes. Table 3 shows that *MultiBin* modules are more effective than discretized classification [21] and also keypoint based method of [14] which is based on localizing keypoints and solving a sophisticated optimization to recover the pose.

MultiBin loss analysis. Table 4 shows the effect of choosing a different number of bins for the Multibox loss on both KITTI and Pascal3D+. In both datasets, using more than one bin consistently outperforms the single-bin variant, which is equivalent to the L2 loss. On KITTI, the best performance is achieved with 2 bins while 8 bins works the best for Pascal3D+. This is due to the fact that the viewpoint distribution in the Pascal3D+ dataset is more diverse. As Table 4 shows, over-binning eventually decreases the effectiveness of the method, as it decreases the training data amount for each bin. We also experimented with different widths of the fully connected layers (see Table 5) and found that increasing the width of the FC layers further yielded some limited gains even beyond width 256.

3D bounding box metrics and comparison. The orientation estimation loss evaluates only a subset of 3D bounding box parameters. To evaluate the accuracy of the rest, we introduce 3 metrics, on which we compare our method against SubCNN [24] for KITTI cars. The first metric is the average error in estimating the 3D coordinate of the center of the objects. The second metric is the average error in estimating the closest point of the 3D box to the camera. This metric is important for driving scenarios where the system needs to avoid hitting obstacles. The last metric is the 3D intersection over union (3D IoU) which is the ultimate metric utilizing all parameters of the estimated 3D bounding boxes. In order to factor away the 2D detector performance for a side-by-side comparison, we kept only the detections from both methods where the detected 2D boxes have $\text{IoU} \geq 0.7$. As Fig. 8 shows, our method outperforms the SubCNN method [24], the current state of the art, across the board in all 3 metrics. Despite this, the 3D IoU numbers are significantly smaller than those that 2D detec-



Figure 6. Qualitative illustration of the 2D detection boxes and the estimated 3D projections, in red for cars and green for cyclists.

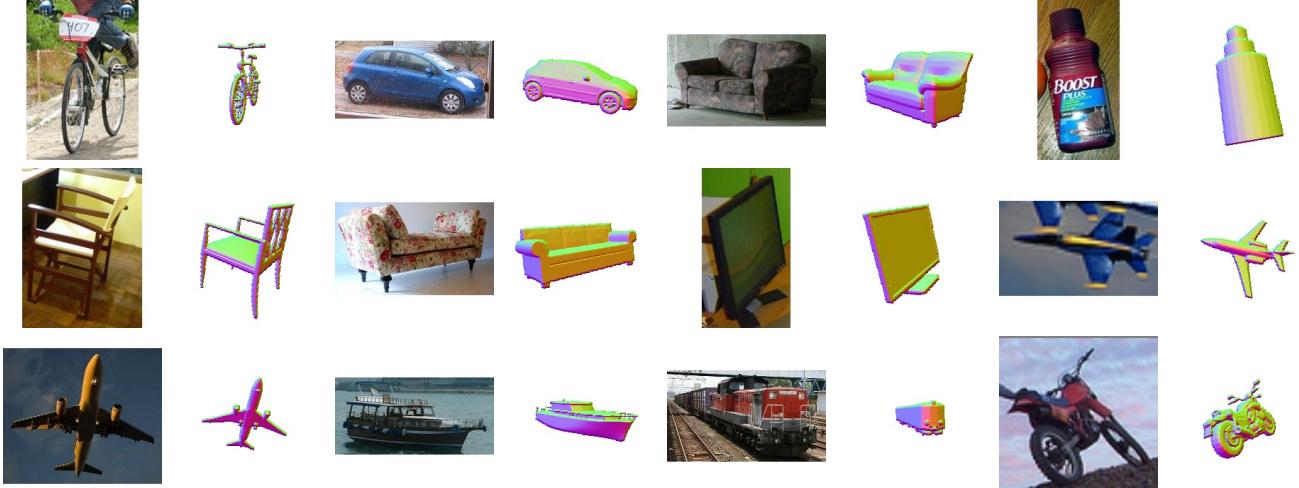


Figure 7. Visualization of Estimated Poses on Pascal3D+ dataset

tors typically obtain on the corresponding 2D metric. This is due to the fact that 3D estimation is a more challenging task, especially as the distance to the object increases. For example, if the car is 50m away from the camera, a translation error of 2m corresponds to about half the car length. Our method handles increasing distance well, as its error for the box center and closest point metrics in Fig. 8 increases approximately linearly with distance, compared to SubCNN’s super-linear degradation. To evaluate the importance of estimating the car dimensions, we evaluated a variant of our method that uses average sizes instead of estimating them. The evaluation shows that regressing the di-

mensions makes a difference in all the 3D metrics. To facilitate comparisons with future work on this problem, we have made the estimated 3D boxes on the split of [22] available at <http://bit.ly/2oaiBgi>.

Training data requirements. One downside of our method is that it needs to learn the parameters for the fully connected layers; it requires more training data than methods that use additional information. To verify this hypothesis, we repeated the experiments for cars but limited the number of training instances to 1100. The same method that achieves 0.9808 in Table 4 with 10828 instances can only achieve 0.9026 on the same test set. Moreover, our re-

| Method | Easy | | | Moderate | | | Hard | | |
|------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| | AOS | AP | OS | AOS | AP | OS | AOS | AP | OS |
| 3DOP[5] | 91.44% | 93.04% | 0.9828 | 86.10% | 88.64% | 0.9713 | 76.52% | 79.10% | 0.9673 |
| Mono3D[4] | 91.01% | 92.33% | 0.9857 | 0.8662% | 88.66% | 0.9769 | 76.84% | 78.96% | 0.9731 |
| SubCNN[24] | 90.67% | 90.81% | 0.9984 | 88.62% | 89.04% | 0.9952 | 78.68% | 79.27% | 0.9925 |
| Our Method | 92.90% | 92.98% | 0.9991 | 88.75% | 89.04% | 0.9967 | 76.76% | 77.17% | 0.9946 |

Table 1. Comparison of the Average Orientation Estimation (AOS), Average Precision (AP) and Orientation Score (OS) on official KITTI dataset for cars. Orientation score is the ratio between AOS and AP.

| Method | Easy | | | Moderate | | | Hard | | |
|------------|--------|--------|--------|----------|--------|--------|--------|--------|--------|
| | AOS | AP | OS | AOS | AP | OS | AOS | AP | OS |
| 3DOP[5] | 70.13% | 78.39% | 0.8946 | 58.68% | 68.94% | 0.8511 | 52.32% | 61.37% | 0.8523 |
| Mono3D[4] | 65.56% | 76.04% | 0.8621 | 54.97% | 66.36% | 0.8283 | 48.77% | 58.87% | 0.8284 |
| SubCNN[24] | 72.00% | 79.48% | 0.9058 | 63.65% | 71.06% | 0.8957 | 56.32% | 62.68% | 0.8985 |
| Our Method | 69.16% | 83.94% | 0.8239 | 59.87% | 74.16% | 0.8037 | 52.50% | 64.84% | 0.8096 |

Table 2. AOS comparison on the official KITTI dataset for cyclists. Our purely data-driven model is not able to match the performance of methods that use additional features and assumptions with just 1100 training examples.

sults on the official KITTI set is significantly better than the split of [22] (see Table 1) because yet more training data is used for training. A similar phenomenon is happening for the KITTI cyclist task. The number of cyclist instances are much less than the number of car instances (1144 labeled cyclists vs 18470 labeled cars). As a result, there is not enough training data for learning the parameters of the fully connected layer well. Although our purely data-driven method achieves competitive results on the cyclists (see Table 5.2), it cannot outperform other methods that use additional features and assumptions.

5.3. Implicit Emergent Attention

In this section, we visualize the parts of cars and bicycles that the network uses in order to estimate the object orientation accurately. Similar to [28], a small gray patch is slid around the image and for each location we record the difference between the estimated and the ground truth orientation. If occluding a specific part of the image by the patch causes a significantly different output, it means that the network attends to that part. Fig. 9 shows such heatmaps of the output differences due to grayed out locations for several car detections. It appears that the network attends to distinct object parts such as tires, lights and side mirror for cars. Our method seems to learn local features similar to keypoints used by other methods, without ever having seen explicitly labeled keypoint ground truth. Another advantage is that our network learns task-specific local features, while human-labeled keypoints are not necessarily the best ones for the task.

5.4. Alternative Representation

In this section we demonstrate the importance of choosing suitable regression parameters within our estimation framework. Here instead of object dimensions, we regress

the location of the 3D box center projection in the image. This allows us to recover the camera ray towards the 3D box center. Any point on that ray can be described by a single parameter λ which is the distance from the camera center. Given the projection of the center of the 3D box and the box orientation, our goal is to estimate λ and the object dimensions: four unknowns for which we have four constraints between 2D box sides and 3D box corners. While the number of parameters to be regressed in this representation is less than those of the proposed method, this representation is more sensitive to regression errors. When there is no constraint on the physical dimension of the box, the optimization tries to satisfy the 2D detection box constraints even if the final dimensions are not plausible for the category of the object.

In order to evaluate the robustness of this representation, we take the ground truth 3D boxes and add realistic noise either to the orientation or to the location of the center of the 3D bounding box while keeping the enclosing 2D bounding box intact. The reason that we added noise was to simulate the parameter estimation errors. 3D boxes reconstructed using this formation satisfy the 2D-3D correspondences but have large box dimension errors as result of small errors in the orientation and box center estimates, as shown in Fig. 10. This investigation supports our choice of 3D regression parameters.

6. Conclusions and Future Directions

In this work, we show how to recover the 3D bounding boxes for known object categories from a single view. Using a novel MultiBin loss for orientation prediction and an effective choice of box dimensions as regression parameters, our method estimates stable and accurate posed 3D bounding boxes without additional 3D shape models, or sampling strategies with complex pre-processing pipelines.

| | aero | bike | boat | bottle | bus | car | chair | table | mbike | sofa | train | tv | mean |
|-----------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|---------------|
| <i>MedErr</i> ([21]) | 13.8 | 17.7 | 21.3 | 12.9 | 5.8 | 9.1 | 14.8 | 15.2 | 14.7 | 13.7 | 8.7 | 15.4 | 13.6 |
| <i>MedErr</i> ([14]) | 8.0 | 13.4 | 40.7 | 11.7 | 2.0 | 5.5 | 10.4 | N/A | N/A | 9.6 | 8.3 | 32.9 | N/A |
| <i>MedErr</i> (Ours) | 13.6 | 12.5 | 22.8 | 8.3 | 3.1 | 5.8 | 11.9 | 12.5 | 12.3 | 12.8 | 6.3 | 11.9 | 11.1 |
| <i>Acc</i> $\frac{\pi}{6}$ ([21]) | 0.81 | 0.77 | 0.59 | 0.93 | 0.98 | 0.89 | 0.80 | 0.62 | 0.88 | 0.82 | 0.80 | 0.80 | 0.8075 |
| <i>Acc</i> $\frac{\pi}{6}$ (Ours) | 0.78 | 0.83 | 0.57 | 0.93 | 0.94 | 0.90 | 0.80 | 0.68 | 0.86 | 0.82 | 0.82 | 0.85 | 0.8103 |

Table 3. Viewpoint Estimation with Ground Truth box on Pascal3D+

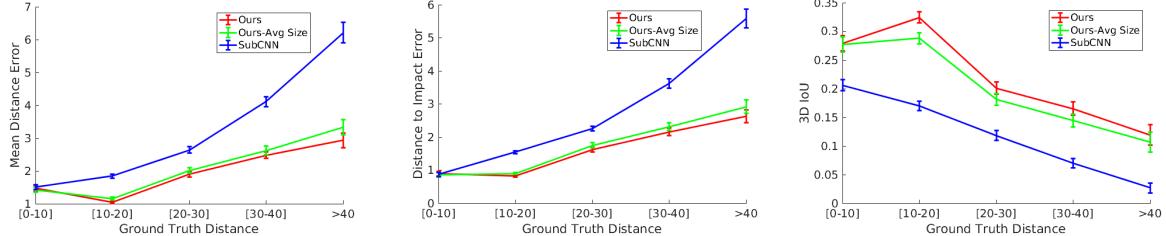


Figure 8. 3D box metrics for KITTI cars. Left: Mean distance error for box center, in meters. Middle: Error in estimating the closest distance from the 3D box to the camera, which is proportional to time-to-impact for driving scenarios. Right: 3D IoU between the predicted and ground truth 3D bounding boxes.

| dataset | # of Bins | 1 | 2 | 4 | 8 | 16 |
|-----------|----------------------------|------|-------------|------|-------------|------|
| KITTI | OS | 0.89 | 0.98 | 0.97 | 0.97 | 0.96 |
| Pascal3D+ | <i>Acc</i> $\frac{\pi}{6}$ | 0.65 | 0.72 | 0.78 | 0.81 | 0.77 |

Table 4. The effect of the number of bins on viewpoint estimation in KITTI and Pascal3D+ datasets

| FC | 64 | 128 | 256 | 512 | 1024 |
|----|--------|--------|--------|--------|--------|
| OS | 0.9583 | 0.9607 | 0.9808 | 0.9854 | 0.9861 |

Table 5. effect of FC width in orientation accuracy

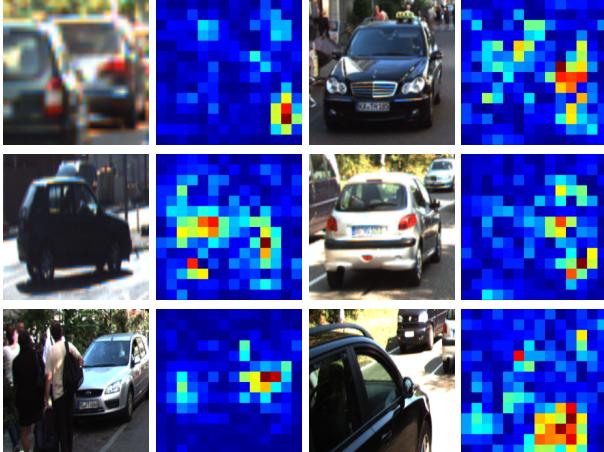


Figure 9. Visualization of the learned attention of the model for orientation estimation. The heatmap shows the image areas that contribute to orientation estimation the most. The network attends to certain meaningful parts of the car such as tires, lights, and side mirrors.

One future direction is to explore the benefits of augmenting the RGB image input in our method with a separate depth channel computed using stereo. Another is to explore 3D box estimation in video, which requires using the tempo-

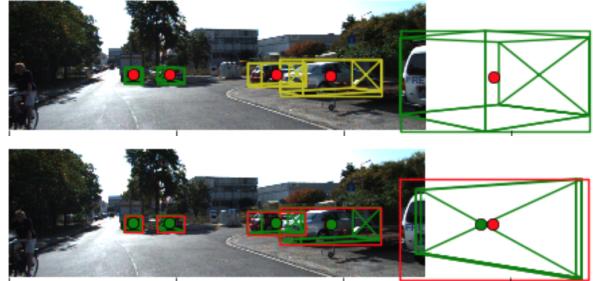


Figure 10. Illustration of the sensitivity of the alternative representation that is estimating both dimensions and translation from geometric constraints. We added small amount of noise to the ground truth angles and tried to recover the ground truth box again. All other parameters are set to the ground truth values.

ral information effectively and can enable the prediction of future object position and velocity.

References

- [1] Supplementary materials. <http://bit.ly/2oYmpuw>. 3, 5
- [2] P. L. A. Geiger and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 1, 2, 5, 6
- [3] Z. Cai, Q. Fan, R. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *ECCV*, 2016. 2, 5
- [4] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *IEEE CVPR*, 2016. 1, 2, 3, 8
- [5] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *NIPS*, 2015. 6, 8

- [6] V. Ferrari, T. Tuytelaars, and L. Gool. Simultaneous object recognition and segmentation from single or multiple model views. *International Journal of Computer Vision (IJCV)*, 62(2):159–188, 2006. 2
- [7] R. Girshick, J. D. T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2015. 2
- [8] S. Hao, Q. Charles, L. Yangyan, and G. Leonidas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 2
- [9] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Category-specific object reconstruction from a single image. In *CVPR*, 2015. 2
- [10] V. Lepetit, F. Moreno-Noguer, and P. Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision (IJCV)*, 2009. 2
- [11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 5
- [12] K. Matzen and N. Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *ICCV*, 2013. 2
- [13] R. Mottaghi, Y. Xiang, and S. Savarese. A coarse-to-fine model for 3d pose estimation and sub-category recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2015. 2
- [14] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017. 3, 6, 9
- [15] B. Pepik, M. Stark, P. Gehler, T. Ritschel, and B. Schiele. 3d object class detection in the wild. In *CVPR*, 2015. 2
- [16] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3d geometry to deformable part models. In *CVPR*, 2012. 2
- [17] P. Poirson, P. Ammirato, A. Berg, and J. Kosecka. Fast single shot detection and pose estimation. In *3DV*, 2016. 2
- [18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 5
- [19] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *IJCV*, 66(3):231–259, 2006. 2
- [20] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 2, 5
- [21] S. Tulsiani and J. Malik. Viewpoints and keypoints. In *CVPR*, 2015. 2, 3, 6, 9
- [22] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *Proceedings of the International Conference on Learning Representation*, 2015. 2, 3, 7, 8
- [23] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2015. 1, 3, 6
- [24] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Subcategory-aware convolutional neural networks for object proposals and detection. In *arXiv:1604.04693*. 2016. 1, 2, 3, 6, 8
- [25] Y. Xiang, W. Kim, W. Chen, J. Ji, C. Choy, H. Su, R. Mottaghi, L. Guibas, and S. Savarese. Objectnet3d: A large scale database for 3d object recognition. In *ECCV*, 2016. 2
- [26] Y. Xiang, R. Mottaghi, and S. Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, 2014. 1, 2, 5
- [27] R. Zhang, P. Isola, and A. Efros. Colorful image colorization. In *ECCV*, 2016. 5
- [28] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *Proceedings of the International Conference on Learning Representation*, 2015. 8
- [29] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detection and pose estimation for grasping. In *IEEE ICRA*, 2013. 2