

DEVIANT: Depth EquiVarIAnt NeTwork for Monocular 3D Object Detection Supplementary Material

A1 Supportive Explanations

We now add some explanations which we could not put in the main paper because of the space constraints.

A1.1 Equivariance vs Augmentation

Equivariance adds suitable inductive bias to the backbone [17, 19] and is not learnt. Augmentation adds transformations to the input data during training or inference.

Equivariance and data augmentation have their own pros and cons. Equivariance models the physics better, is mathematically principled and is so more agnostic to data distribution shift compared to the data augmentation. A downside of equivariance compared to the augmentation is equivariance requires mathematical modelling, may not always exist [8], is not so intuitive and generally requires more flops for inference. On the other hand, data augmentation is simple, intuitive and fast, but is not mathematically principled. The choice between equivariance and data augmentation is a withstanding question in machine learning [25].

A1.2 Why do 2D CNN detectors generalize?

We now try to understand why 2D CNN detectors generalize well. Consider an image $h(u, v)$ and Φ be the CNN. Let $\mathcal{T}_{\mathbf{t}}$ denote the translation in the (u, v) space. The 2D translation equivariance [6, 7, 61] of the CNN means that

$$\begin{aligned} \Phi(\mathcal{T}_{\mathbf{t}}h(u, v)) &= \mathcal{T}_{\mathbf{t}}\Phi(h(u, v)) \\ \implies \Phi(h(u + t_u, v + t_v)) &= \Phi(h(u, v)) + (t_u, t_v) \end{aligned} \quad (5)$$

where (t_u, t_v) is the translation in the (u, v) space.

Assume the CNN predicts the object position in the image as (u', v') . Then, we write

$$\Phi(h(u, v)) = (\hat{u}, \hat{v}) \quad (6)$$

Now, we want the CNN to predict the output the position of the same object translated by (t_u, t_v) . The new image is thus $h(u + t_u, v + t_v)$. The CNN easily predicts the translated position of the object because all CNN is to do is to invoke its 2D translation equivariance of Eq. (5), and translate the previous prediction by the same amount. In other words,

$$\Phi(h(u + t_u, v + t_v)) = \Phi(h(u, v)) + (t_u, t_v)$$

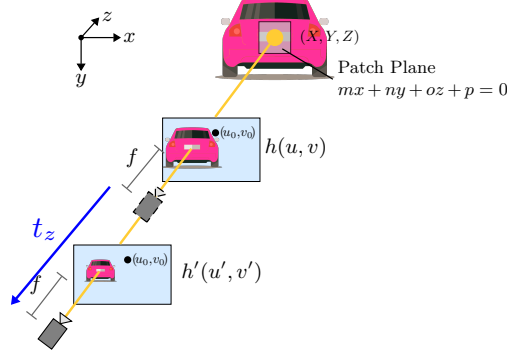


Fig. 5: **Equivariance exists** for the patch plane when there is depth translation of the ego camera. Downscaling converts image h to image h' .

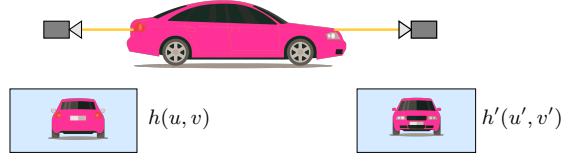


Fig. 6: **Example of non-existence of equivariance** [8] when there is 180° rotation of the ego camera. No transformation can convert image h to image h' .

$$\begin{aligned}
 &= (\hat{u}, \hat{v}) + (t_u, t_v) \\
 &= (\hat{u} + t_u, \hat{v} + t_v)
 \end{aligned}$$

Intuitively, equivariance is a disentanglement method. The 2D translation equivariance disentangles the 2D translations (t_u, t_v) from the original image h and therefore, the network generalizes to unseen 2D translations.

A1.3 Existence and Non-existence of Equivariance

The result from [8] says that generic projective equivariance does not exist in particular with rotation transformations. We now show an example of when the equivariance exists and does not exist in the projective manifold in Figs. 5 and 6 respectively.

A1.4 Why do not Monocular 3D CNN detectors generalize?

Monocular 3D CNN detectors do not generalize well because they are not equivariant to arbitrary 3D translations in the projective manifold. To show this, let $H(x, y, z)$ denote a 3D point cloud. The monocular detection network Φ operates on the projection $h(u, v)$ of this point cloud H to output the position $(\hat{x}, \hat{y}, \hat{z})$ as

$$\Phi(\mathbf{K}H(x, y, z)) = (\hat{x}, \hat{y}, \hat{z})$$

$$\implies \Phi(h(u, v)) = (\hat{x}, \hat{y}, \hat{z}),$$

where $\mathbf{K}$ denotes the projection operator. We translate this point cloud by an arbitrary 3D translation of (t_x, t_y, t_z) to obtain the new point cloud $H(x+t_x, y+t_y, z+t_z)$. Then, we again ask the monocular detector Φ to do prediction over the translated point cloud. However, we find that

$$\begin{aligned} \Phi(\mathbf{K}H(x+t_x, y+t_y, z+t_z)) &\neq \Phi(h(u + \mathbf{K}(t_x, t_y, t_z), v + \mathbf{K}(t_x, t_y, t_z))) \\ &= \Phi(h(u, v)) + \mathbf{K}(t_x, t_y, t_z) \\ \implies \Phi(\mathbf{K}H(x+t_x, y+t_y, z+t_z)) &\neq \Phi(\mathbf{K}H(x, y, z)) + \mathbf{K}(t_x, t_y, t_z) \end{aligned}$$

In other words, the projection operator $\mathbf{K}$ does not distribute over the point cloud H and arbitrary 3D translation of (t_x, t_y, t_z) . Hence, if the network Φ is a vanilla CNN (existing monocular backbone), it can no longer invoke its 2D translation equivariance of Eq. (5) to get the new 3D coordinates $(\hat{x} + t_x, \hat{y} + t_y, \hat{z} + t_z)$.

Note that the LiDAR based 3D detectors with 3D convolutions do not suffer from this problem because they do not involve any projection operator $\mathbf{K}$. Thus, this problem exists only in monocular 3D detection. This makes monocular 3D detection different from 2D and LiDAR based 3D object detection.

A1.5 Overview of Theorem 1

We now pictorially provide the overview of Theorem 1 (Example 13.2 from [30]), which links the planarity and projective transformations in the continuous world in Fig. 7.

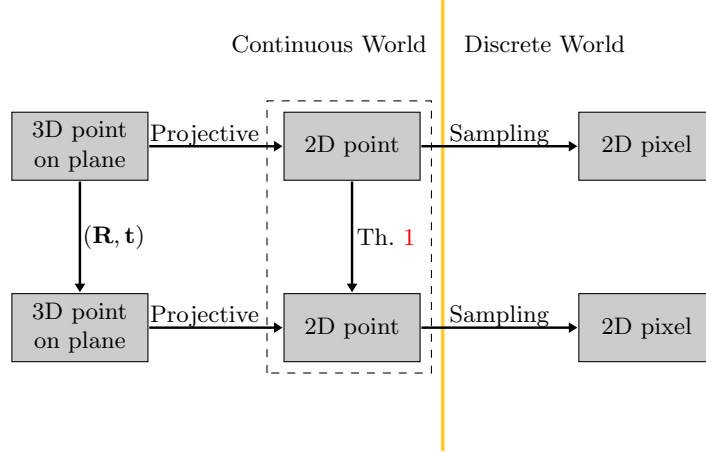


Fig. 7: **Overview** of Theorem 1 (Example 13.2 from [30]), which links the planarity and projective transformations in the continuous world.

A1.6 Approximation of Corollary 1

We now give the approximation under which Corollary 1 is valid. We assume that the ego camera does not undergo any rotation. Hence, we substitute $\mathbf{R} = \mathbf{I}$ in Eq. (1) to get

$$h(u - u_0, v - v_0) = h' \left(\frac{f \left(\frac{1+t_x \frac{m}{p}}{t_z \frac{m}{p}} (u - u_0) + t_x \frac{n}{p} (v - v_0) + t_x \frac{o}{p} f \right)}{t_z \frac{m}{p} (u - u_0) + t_z \frac{n}{p} (v - v_0) + \left(1 + t_z \frac{o}{p} \right) f}, \frac{f \frac{t_y \frac{m}{p}}{t_z \frac{m}{p}} (u - u_0) + \left(1 + t_y \frac{n}{p} \right) (v - v_0) + t_y \frac{o}{p} f}{t_z \frac{m}{p} (u - u_0) + t_z \frac{n}{p} (v - v_0) + \left(1 + t_z \frac{o}{p} \right) f} \right). \quad (7)$$

Next, we use the assumption that the ego vehicle moves in the z -direction as in [5], *i.e.*, substitute $t_x = t_y = 0$ to get

$$h(u - u_0, v - v_0) = h' \left(\frac{u - u_0}{\frac{t_z \frac{m}{p}}{f} (u - u_0) + \frac{t_z \frac{n}{p}}{f} (v - v_0) + \left(1 + t_z \frac{o}{p} \right)}, \frac{v - v_0}{\frac{t_z \frac{m}{p}}{f} (u - u_0) + \frac{t_z \frac{n}{p}}{f} (v - v_0) + \left(1 + t_z \frac{o}{p} \right)} \right). \quad (8)$$

The patch plane is $mx + ny + oz + p = 0$. We consider the planes in the front of camera. Without loss of generality, consider $p < 0$ and $o > 0$.

We first write the denominator D of RHS term in Eq. (8) as

$$\begin{aligned} D &= \frac{t_z}{f} \frac{m}{p} (u - u_0) + \frac{t_z}{f} \frac{n}{p} (v - v_0) + \left(1 + t_z \frac{o}{p} \right) \\ &= 1 + \frac{t_z}{p} \left(\frac{m}{f} (u - u_0) + \frac{n}{f} (v - v_0) + o \right) \end{aligned}$$

Because we considered patch planes in front of the camera, $p < 0$. Also consider $t_z < 0$, which implies $t_z/p > 0$. Now, we bound the term in the parantheses of the above equation as

$$\begin{aligned} D &\leq 1 + \frac{t_z}{p} \left| \frac{m}{f} (u - u_0) + \frac{n}{f} (v - v_0) + o \right| \\ &\leq 1 + \frac{t_z}{p} \left(\left| \frac{m}{f} (u - u_0) \right| + \left| \frac{n}{f} (v - v_0) \right| + |o| \right) \quad \text{by Triangle inequality} \\ &\leq 1 + \frac{t_z}{p} \left(\frac{|m|}{f} \frac{W}{2} + \frac{|n|}{f} \frac{H}{2} + o \right), (u - u_0) \leq \frac{W}{2}, (v - v_0) \leq \frac{H}{2}, |o| = o \\ &\leq 1 + \frac{t_z}{p} \left(\frac{|m|}{f} \frac{W}{2} + \frac{|n|}{f} \frac{W}{2} + o \right), H \leq W \\ &\leq 1 + \frac{t_z}{p} \left(\frac{(|m| + |n|)W}{2f} + o \right), \end{aligned}$$

If the coefficients of the patch plane m, n, o , its width W and focal length f follow the relationship $\frac{(|m|+|n|)W}{2f} \ll o$, the patch plane is “approximately” parallel to the image plane. Then, a few quantities can be ignored in the denominator D to get

$$D \approx 1 + t_z \frac{o}{p} \quad (9)$$

Therefore, the RHS of Eq. (8) gets simplified and we obtain

$$\mathcal{T}_s : h(u - u_0, v - v_0) \approx h' \left(\frac{u - u_0}{1 + t_z \frac{o}{p}}, \frac{v - v_0}{1 + t_z \frac{o}{p}} \right) \quad (10)$$

An immediate benefit of using the approximation is Eq. (2) does not depend on the distance of the patch plane from the camera. This is different from wide-angle camera assumption, where the ego camera is assumed to be far from the patch plane. Moreover, patch planes need not be perfectly aligned with the image plane for Eq. (2). Even small enough perturbed patch planes work. We next show the approximation in the Fig. 8 with θ denoting the deviation from the perfect parallel plane. The deviation θ is about 3 degrees for the KITTI dataset while it is 6 degrees for the Waymo dataset.

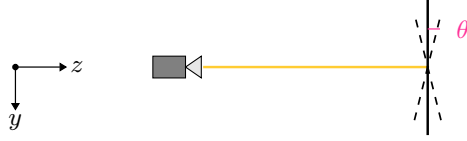


Fig. 8: **Approximation of Corollary 1**. Bold shows the patch plane parallel to the image plane. The dotted line shows the approximated patch plane.

e.g. The following are valid patch planes for KITTI images whose focal length $f = 707$ and width $W = 1242$.

$$\begin{aligned} -0.05x + 0.05y + z &= 30 \\ 0.05x - 0.05y + z &= 30 \end{aligned} \quad (11)$$

The following are valid patch planes for Waymo images whose focal length $f = 2059$ and width $W = 1920$.

$$\begin{aligned} -0.1x + 0.1y + z &= 30 \\ 0.1x - 0.1y + z &= 30 \end{aligned} \quad (12)$$

Although the assumption is slightly restrictive, we believe our method shows improvements on both KITTI and Waymo datasets because the car patches are approximately parallel to image planes and also because the depth remains the hardest parameter to estimate [53].

A1.7 Scale Equivariance of SES Convolution for Images

[74] derive the scale equivariance of SES convolution for a 1D signal. We simply follow on their footsteps to get the scale equivariance of SES convolution for a 2D image $h(u, v)$ for the sake of completeness. Let the scaling of the image h be s . Let $*$ denote the standard vanilla convolution and Ψ denote the convolution filter. Then, the convolution of the downsampled image $\mathcal{T}_s(h)$ with the filter Ψ is given by

$$\begin{aligned}
& [\mathcal{T}_s(h) * \Psi](u, v) \\
&= \int \int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \Psi(u' - u, v' - v) du' dv' \\
&= s^2 \int \int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \Psi\left(s\frac{u' - u}{s}, s\frac{v' - v}{s}\right) d\left(\frac{u'}{s}\right) d\left(\frac{v'}{s}\right) \\
&= s^2 \int \int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \mathcal{T}_{s^{-1}}\left[\Psi\left(\frac{u' - u}{s}, \frac{v' - v}{s}\right)\right] d\left(\frac{u'}{s}\right) d\left(\frac{v'}{s}\right) \\
&= s^2 \int \int h\left(\frac{u'}{s}, \frac{v'}{s}\right) \mathcal{T}_{s^{-1}}\left[\Psi\left(\frac{u'}{s} - \frac{u}{s}, \frac{v'}{s} - \frac{v}{s}\right)\right] d\left(\frac{u'}{s}\right) d\left(\frac{v'}{s}\right) \\
&= s^2 [h * \mathcal{T}_{s^{-1}}(\Psi)]\left(\frac{u}{s}, \frac{v}{s}\right) \\
&= s^2 \mathcal{T}_s[h * \mathcal{T}_{s^{-1}}(\Psi)](u, v).
\end{aligned} \tag{13}$$

Next, [74] re-parametrize the SES filters by writing $\Psi_\sigma(u, v) = \frac{1}{\sigma^2} \Psi\left(\frac{u}{\sigma}, \frac{v}{\sigma}\right)$. Substituting in Eq. (13), we get

$$[\mathcal{T}_s(h) * \Psi_\sigma](u, v) = s^2 \mathcal{T}_s[h * \mathcal{T}_{s^{-1}}(\Psi_\sigma)](u, v) \tag{14}$$

Moreover, the re-parametrized filters are separable [74] by construction and so, one can write

$$\Psi_\sigma(u, v) = \Psi_\sigma(u) \Psi_\sigma(v). \tag{15}$$

The re-parametrization and separability leads to the important property that

$$\begin{aligned}
\mathcal{T}_{s^{-1}}(\Psi_\sigma(u, v)) &= \mathcal{T}_{s^{-1}}(\Psi_\sigma(u) \Psi_\sigma(v)) \\
&= \mathcal{T}_{s^{-1}}(\Psi_\sigma(u)) \mathcal{T}_{s^{-1}}(\Psi_\sigma(v)) \\
&= s^{-2} \Psi_{s^{-1}\sigma}(u) \Psi_{s^{-1}\sigma}(v) \\
&= s^{-2} \Psi_{s^{-1}\sigma}(u, v).
\end{aligned} \tag{16}$$

Substituting above in the RHS of Eq. (14), we get

$$\begin{aligned}
& [\mathcal{T}_s(h) * \Psi_\sigma](u, v) = s^2 \mathcal{T}_s[h * s^{-2} \Psi_{s^{-1}\sigma}](u, v) \\
\implies & [\mathcal{T}_s(h) * \Psi_\sigma](u, v) = \mathcal{T}_s[h * \Psi_{s^{-1}\sigma}](u, v),
\end{aligned} \tag{17}$$

which is a cleaner form of Eq. (13). Eq. (17) says that convolving the downsampled image with a filter is same as the downscaling the result of convolving the image with the upscaled filter [74]. This additional constraint regularizes the scale (depth) predictions for the image, leading to better generalization.

Table 13: **Comparison of Methods** on the basis of inputs, convolution kernels, outputs and whether output are scale-constrained.

Method	Input Frame	#Conv Kernel	Output	Output Constrained for Scales?
Vanilla CNN	1	1	4D	×
Depth-Aware [4]	1	> 1	4D	×
Dilated CNN [97]	1	> 1	5D	Integer [92]
DEVIANT	1	> 1	5D	Float
Depth-guided [20]	1 + Depth	1	4D	Integer [92]
Kinematic3D [5]	> 1	1	5D	×

A1.8 Why does DEVIANT generalize better compared to CNN backbone?

DEVIANT models the physics better compared to the CNN backbone. CNN generalizes better for 2D detection because of the 2D translation equivariance in the Euclidean manifold. However, monocular 3D detection does not belong to the Euclidean manifold but is a task of the projective manifold. Modeling translation equivariance in the correct manifold improves generalization. For monocular 3D detection, we take the first step towards the general 3D translation equivariance by embedding equivariance to depth translations. The 3D depth equivariance in DEVIANT uses Eq. (14) and thus imposes an additional constraint on the feature maps. This additional constraint results in consistent depth estimates from the current image and a virtual image (obtained by translating the ego camera), and therefore, better generalization than CNNs. On the other hand, CNNs, by design, do not constrain the depth estimates from the current image and a virtual image (obtained by translating the ego camera), and thus, their depth estimates are entirely data-driven.

A1.9 Why not Fixed Scale Assumption?

We now answer the question of keeping the fixed scale assumption. If we assume fixed scale assumption, then vanilla convolutional layers have the right equivariance. However, we do not keep this assumption because the ego camera translates along the depth in driving scenes and also, because the depth is the hardest parameter to estimate [53] for monocular detection. So, zero depth translation or fixed scale assumption is always violated.

A1.10 Comparisons with Other Methods

We now list out the differences between different convolutions and monocular detection methods in Tab. 13. Kinematic3D [5] does not constrain the output at feature map level, but at system level using Kalman Filters. The closest to our method is the Dilated CNN (DCNN) [97]. We show in Tab. 9 that DEVIANT outperforms Dilated CNN.

A1.11 Why is Depth the hardest among all parameters?

Images are the 2D projections of the 3D scene, and therefore, the depth is lost during projection. Recovering this depth is the most difficult to estimate, as shown in Tab. 1 of [53]. Monocular detection task involves estimating 3D center, 3D dimensions and the yaw angle. The right half of Tab. 1 in [53] shows that if the ground truth 3D center is replaced with the predicted center, the detection reaches a minimum. Hence, 3D center is the most difficult to estimate among center, dimensions and pose. Most monocular 3D detectors further decompose the 3D center into projected (2D) center and depth. Out of projected center and depth, Tab. 1 of [53] shows that replacing ground truth depth with the predicted depth leads to inferior detection compared to replacing ground truth projected center with the predicted projected center. Hence, we conclude that depth is the hardest parameter to estimate.

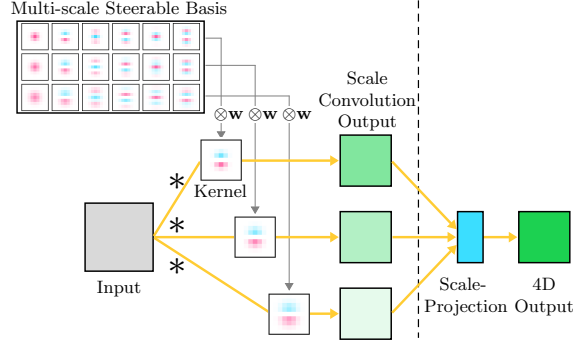


Fig. 9: **(a) SES convolution** [29, 74] The non-trainable basis functions multiply with learnable weights $\mathbf{w}$ to get kernels. The input then convolves with these kernels to get multi-scale 5D output. **(b) Scale-Projection** [74] takes max over the scale dimension of the 5D output and converts it to 4D. [Key: * = Vanilla convolution.]

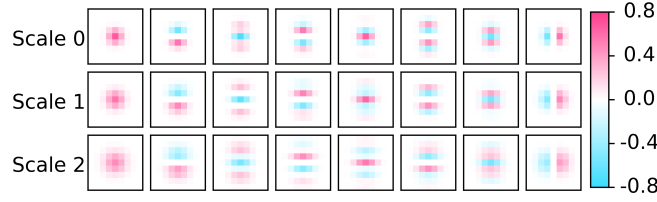


Fig. 10: **Steerable Basis** [74] for 7×7 SES convolution filters. (Showing only 8 of the 49 members for each scale).

A2 Implementation Details

We now provide some additional implementation details for facilitating reproduction of this work.

A2.1 Steerable Filters of SES Convolution

We use the scale equivariant steerable blocks proposed by [73] for our DEVIANT backbone. We now share the implementation details of these steerable filters.

Basis. Although steerable filters can use any linearly independent functions as their basis, we stick with the Hermite polynomials as the basis [73]. Let $(0, 0)$ denote the center of the function and (u, v) denote the pixel coordinates. Then, the filter coefficients $\psi_{\sigma nm}$ [73] are

$$\psi_{\sigma nm} = \frac{A}{\sigma^2} H_n \left(\frac{u}{\sigma} \right) H_m \left(\frac{v}{\sigma} \right) e^{-\frac{u^2 + v^2}{\sigma^2}} \quad (18)$$

H_n denotes the Probabilist's Hermite polynomial of the n th order, and A is the normalization constant. The first six Probabilist's Hermite polynomials are

$$H_0(x) = 1 \quad (19)$$

$$H_1(x) = x \quad (20)$$

$$H_2(x) = x^2 - 1 \quad (21)$$

$$H_3(x) = x^3 - 3x \quad (22)$$

$$H_4(x) = x^4 - 6x^2 + 3 \quad (23)$$

Fig. 10 visualizes some of the SES filters and shows that the basis is indeed at different scales.

A2.2 Monocular 3D Detection

Architecture. We use the DLA-34 [98] configuration, with the standard Feature Pyramid Network (FPN) [44], binning and ensemble of uncertainties. FPN is a bottom-up feed-forward CNN that computes feature maps with a downscaling factor of 2, and a top-down network that brings them back to the high-resolution ones. There are total six feature maps levels in this FPN.

We use DLA-34 as the backbone for our baseline GUP Net [49], while we use SES-DLA-34 as the backbone for DEVIANT. We also replace the 2D pools by 3D pools with pool along the scale dimensions as 1 for DEVIANT.

We initialize the vanilla CNN from ImageNet weights. For DEVIANT, we use the regularized least squares [73] to initialize the trainable weights in all the Hermite scales from the ImageNet [18] weights. Compared to initializing one of the scales as proposed in [73], we observed more stable convergence in initializing all the Hermite scales.

We output three foreground classes for KITTI dataset. We also output three foreground classes for Waymo dataset ignoring the Sign class [62].

Datasets. We use the publicly available KITTI, Waymo and nuScenes datasets for our experiments. KITTI is available at http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d under CC BY-NC-SA 3.0 License. Waymo is available at https://waymo.com/intl/en_us/dataset-download-terms/ under the Apache License, Version 2.0. nuScenes is available at <https://www.nuscenes.org/nuscenes> under CC BY-NC-SA 4.0 International Public License.

Augmentation. Unless otherwise stated, we horizontal flip the training images with probability 0.5, and use scale augmentation as 0.4 as well for all the models [49] in training.

Pre-processing. The only pre-processing step we use is image resizing.

- *KITTI*. We resize the [370, 1242] sized KITTI images, and bring them to the [384, 1280] resolution [49].
- *Waymo*. We resize the [1280, 1920] sized Waymo images, and bring them to the [512, 768] resolution. This resolution preserves their aspect ratio.

Box Filtering. We apply simple hand-crafted rules for filtering out the boxes. We ignore the box if it belongs to a class different from the detection class.

- *KITTI*. We train with boxes which are atleast $2m$ distant from the ego camera, and with visibility > 0.5 [49].
- *Waymo*. We train with boxes which are atleast $2m$ distant from the ego camera. The Waymo dataset does not have any occlusion based labels. However,

Waymo provides the number of LiDAR points inside each 3D box which serves as a proxy for the occlusion. We train the boxes which have more than 100 LiDAR points for the vehicle class and have more than 50 LiDAR points for the cyclist and pedestrian class.

Training. We use the training protocol of GUP Net [49] for all our experiments. Training uses the Adam optimizer [35] and weight-decay 1×10^{-5} . Training dynamically weighs the losses using Hierarchical Task Learning (HTL) [49] strategy keeping K as 5 [49]. Training also uses a linear warmup strategy in the first 5 epochs to stabilize the training. We choose the model saved in the last epoch as our final model for all our experiments.

- *KITTI*. We train with a batch size of 12 on single Nvidia A100 (40GB) GPU for 140 epochs. Training starts with a learning rate 1.25×10^{-3} with a step decay of 0.1 at the 90th and the 120th epoch.
- *Waymo*. We train with a batch size of 40 on single Nvidia A100 (40GB) GPU for 30 epochs because of the large size of the Waymo dataset. Training starts with a learning rate 1.25×10^{-3} with a step decay of 0.1 at the 18th and the 26th epoch.

Losses. We use the GUP Net [49] multi-task losses before the NMS for training. The total loss $\mathcal{L}$ is given by

$$\begin{aligned} \mathcal{L} = & \mathcal{L}_{\text{heatmap}} + \mathcal{L}_{2\text{D},\text{offset}} + \mathcal{L}_{2\text{D},\text{size}} + \mathcal{L}_{3\text{D}2\text{D},\text{offset}} + \mathcal{L}_{3\text{D},\text{angle}} \\ & + \mathcal{L}_{3\text{D},l} + \mathcal{L}_{3\text{D},w} + \mathcal{L}_{3\text{D},h} + \mathcal{L}_{3\text{D},\text{depth}}. \end{aligned} \quad (24)$$

The individual terms are given by

$$\mathcal{L}_{\text{heatmap}} = \text{Focal}(\text{class}^b, \text{class}^g), \quad (25)$$

$$\mathcal{L}_{2\text{D},\text{offset}} = L_1(\delta_{2\text{D}}^b, \delta_{2\text{D}}^g), \quad (26)$$

$$\mathcal{L}_{2\text{D},\text{size}} = L_1(w_{2\text{D}}^b, w_{2\text{D}}^g) + L_1(h_{2\text{D}}^b, h_{2\text{D}}^g), \quad (27)$$

$$\mathcal{L}_{3\text{D}2\text{D},\text{offset}} = L_1(\delta_{3\text{D}2\text{D}}^b, \delta_{3\text{D}2\text{D}}^g) \quad (28)$$

$$\mathcal{L}_{3\text{D},\text{angle}} = \text{CE}(\alpha^b, \alpha^g) \quad (29)$$

$$\mathcal{L}_{3\text{D},l} = L_1(\mu_{l3\text{D}}^b, \delta_{l3\text{D}}^g) \quad (30)$$

$$\mathcal{L}_{3\text{D},w} = L_1(\mu_{w3\text{D}}^b, \delta_{w3\text{D}}^g) \quad (31)$$

$$\mathcal{L}_{3\text{D},h} = \frac{\sqrt{2}}{\sigma_{h3\text{D}}} L_1(\mu_{h3\text{D}}^b, \delta_{h3\text{D}}^g) + \ln(\sigma_{h3\text{D}}) \quad (32)$$

$$\mathcal{L}_{3\text{D},\text{depth}} = \frac{\sqrt{2}}{\sigma_d} L_1(\mu_d^b, \mu_d^g) + \ln(\sigma_d), \quad (33)$$

where,

$$\mu_d^b = f \frac{\mu_{h3\text{D}}^b}{h_{2\text{D}}^b} + \mu_{d,\text{pred}} \quad (34)$$

$$\sigma_d = \sqrt{\left(f \frac{\sigma_{h3\text{D}}}{h_{2\text{D}}^b}\right)^2 + \sigma_{d,\text{pred}}^2}. \quad (35)$$

The superscripts b and g denote the predicted box and ground truth box respectively. CE and Focal denote the Cross Entropy and Focal loss respectively.

The number of heatmaps depends on the number of output classes. δ_{2D} denotes the deviation of the 2D center from the center of the heatmap. $\delta_{3D2D,offset}$ denotes the deviation of the projected 3D center from the center of the heatmap. The orientation loss is the cross entropy loss between the binned observation angle of the prediction and the ground truth. The observation angle α is split into 12 bins covering 30° range. δ_{l3D} , δ_{w3D} and δ_{h3D} denote the deviation of the 3D length, width and height of the box from the class dependent mean size respectively.

The depth is the hardest parameter to estimate [53]. So, GUP Net uses in-network ensembles to predict the depth. It obtains a Laplacian estimate of depth from the 2D height, while it obtains another estimate of depth from the prediction of depth. It then adds these two depth estimates.

Inference. Our testing resolution is same as the training resolution. We do not use any augmentation for test/validation. We keep the maximum number of objects to 50 in an image, and we multiply the class and predicted confidence to get the box’s overall score in inference as in [36]. We consider output boxes with scores greater than a threshold of 0.2 for KITTI [49] and 0.1 for Waymo [62].

Table 14: **Generalization gap** ($\uparrow$) on KITTI Val cars. Monocular detection has huge generalization gap between training and inference sets. [Key: **Best**]

Method	Scale Eqv	Set	IoU _{3D} ≥ 0.7						IoU _{3D} ≥ 0.5					
			AP _{3D R40} [%]($\uparrow$)			AP _{BEV R40} [%]($\uparrow$)			AP _{3D R40} [%]($\uparrow$)			AP _{BEV R40} [%]($\uparrow$)		
			Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
GUP Net [49]		Train	91.83	74.87	67.43	95.19	80.95	73.55	99.50	93.62	86.22	99.56	93.88	86.46
		Val	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
		Gap	70.73	59.39	54.55	66.61	60.03	55.72	40.55	49.63	48.15	34.96	46.12	43.49
DEVIANT	$\checkmark$	Train	91.09	76.19	67.16	94.76	82.61	75.51	99.37	93.56	88.57	99.50	93.87	88.90
		Val	24.63	16.54	14.52	32.60	23.04	19.99	61.00	46.00	40.18	65.28	49.63	43.50
		Gap	66.46	59.65	52.64	62.16	59.57	55.52	38.37	47.56	48.39	34.22	44.24	45.40

 Table 15: **Comparison on multiple backbones** on KITTI Val cars. [Key: **Best**]

BackBone	Method	IoU _{3D} ≥ 0.7						IoU _{3D} ≥ 0.5					
		AP _{3D R40} [%]($\uparrow$)			AP _{BEV R40} [%]($\uparrow$)			AP _{3D R40} [%]($\uparrow$)			AP _{BEV R40} [%]($\uparrow$)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
ResNet-18	GUP Net [49]	18.86	13.20	11.01	26.05	19.37	16.57	54.90	40.65	34.98	60.54	46.13	40.12
	DEVIANT	20.27	14.21	12.56	28.09	20.32	17.49	55.75	42.41	36.97	60.82	46.43	40.59
DLA-34	GUP Net [49]	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
	DEVIANT	24.63	16.54	14.52	32.60	23.04	19.99	61.00	46.00	40.18	65.28	49.63	43.50

A3 Additional Experiments and Results

We now provide additional details and results of the experiments evaluating our system’s performance.

A3.1 KITTI Val Split

Monocular Detection has Huge Generalization Gap. As mentioned in Sec. 1, we now show that the monocular detection has huge generalization gap between training and inference. We report the object detection performance on the train and validation (val) set for the two models on KITTI Val split in Tab. 14. Tab. 14 shows that the performance of our baseline GUP Net [49] and our DEVIANT is huge on the training set, while it is less than one-fourth of the train performance on the val set.

We also report the **generalization gap** metric [93] in Tab. 14, which is the difference between training and validation performance. The generalization gap at both the thresholds of 0.7 and 0.5 is huge.

Comparison on Multiple Backbones. A common trend in 2D object detection community is to show improvements on multiple backbones [82]. DD3D [57] follows this trend and also reports their numbers on multiple backbones. Therefore, we follow the same and compare with our baseline on multiple backbones on KITTI Val cars in Tab. 15. Tab. 15 shows that DEVIANT shows consistent improvements over GUP Net [49] in 3D object detection on multiple backbones, proving the effectiveness of our proposal.

Comparison with Bigger CNN Backbones. Since the SES blocks increase the Flop counts significantly compared to the vanilla convolution block, we next compare DEVIANT with bigger CNN backbones with comparable GFLOPs and FPS/ wall-clock time (instead of same configuration) in Tab. 16. We compare

Table 16: **Results with bigger CNNs having similar flops on KITTI Val cars.** [Key: **Best**]

Method	BackBone	Param (M)	Disk Size (MB)	Flops (G)	Infer (ms)	AP _{3D} IoU _{3D} ≥ 0.7 (★)			AP _{3D} IoU _{3D} ≥ 0.5 (★)		
		(M)	(MB)	(G)	(ms)	Easy	Mod	Hard	Easy	Mod	Hard
GUP Net [49]	DLA-34	16	235	30	20	21.10	15.48	12.88	58.95	43.99	38.07
GUP Net [49]	DLA-102	34	583	70	25	20.96	14.64	12.80	57.06	41.78	37.26
GUP Net [49]	DLA-169	54	814	114	30	21.76	15.35	12.72	57.60	43.27	37.32
DEVIANT	SES-DLA-34	16	236	235	40	24.63	16.54	14.52	61.00	46.00	40.18

Table 17: **Results on KITTI Val cyclists and pedestrians (Cyc/Ped) (IoU_{3D} ≥ 0.5).** [Key: **Best**, **Second Best**]

Method	Extra	Cyc AP _{3D} R ₄₀ [%] (★)			Ped AP _{3D} R ₄₀ [%] (★)		
		Easy	Mod	Hard	Easy	Mod	Hard
GrooMeD-NMS [36]	—	0.00	0.00	0.00	3.79	2.71	2.61
MonoDIS [70]	—	1.52	0.73	0.71	3.20	2.28	1.71
MonoDIS-M [68]	—	2.70	1.50	1.30	9.50	7.10	5.70
GUP Net (Retrained) [49]	—	4.41	2.17	2.03	9.37	6.84	5.73
DEVIANT (Ours)	—	4.05	2.20	2.14	9.85	7.18	5.42

DEVIANT with DLA-102 and DLA-169 - two biggest DLA networks with ImageNet weights⁴ on KITTI Val split. We use the fvcare library⁵ to get the parameters and flops. Tab. 16 shows that DEVIANT again outperforms the bigger CNN backbones, especially on nearby objects. We believe this happens because the bigger CNN backbones have more trainable parameters than DEVIANT, which leads to overfitting. Although DEVIANT takes more time compared to the CNN backbones, DEVIANT still keeps the inference almost real-time.

Performance on Cyclists and Pedestrians. Tab. 17 lists out the results of 3D object detection on KITTI Val Cyclist and Pedestrians. The results show that DEVIANT is competitive on challenging Cyclist and achieves SOTA results on Pedestrians on the KITTI Val split.

Cross-Dataset Evaluation Details. For cross-dataset evaluation, we test on all 3,769 images of the KITTI Val split, as well as all frontal 6,019 images of the nuScenes Val split [9], as in [67]. We first convert the nuScenes Val images to the KITTI format using the `export_kitti`⁶ function in the nuscnescs devkit. We keep KITTI Val images in the [384, 1280] resolution, while we keep the nuScenes Val images in the [384, 672] resolution to preserve the aspect ratio. For M3D-RPN [4], we bring the nuScenes Val images in the [512, 910] resolution.

Monocular 3D object detection relies on the camera focal length to back-project the projected centers into the 3D space. Therefore, the 3D centers depends on the focal length of the camera used in the dataset. Hence, one should take the camera focal length into account while doing cross-dataset evaluation. We now calculate the camera focal length of a dataset as follows. We take the camera matrix $\mathbf{K}$ and calculate the normalized focal length $\bar{f} = \frac{2f_y}{H}$, where H denotes the height of the image. The normalized focal length $\bar{f}$ for the KITTI

⁴ Available at <http://dl.yf.io/dla/models/imagenet/>

⁵ <https://github.com/facebookresearch/fvcare>

⁶ https://github.com/nutonomy/nuscenes-devkit/blob/master/python-sdk/nuscenes/scripts/export_kitti.py

Table 18: **Stress Test** with rotational and xy -translation ego movement on KITTI Val cars. [Key: **Best**]

Set	Method	AP _{3D} IoU _{3D} ≥ 0.7 (★)			AP _{3D} IoU _{3D} ≥ 0.5 (★)		
		Easy	Mod	Hard	Easy	Mod	Hard
Subset (306)	GUP Net [49]	17.22	11.43	9.91	47.47	35.02	32.63
	DEVIANT	20.17	12.49	10.93	49.81	36.93	34.32
KITTI Val (3769)	GUP Net [49]	21.10	15.48	12.88	58.95	43.99	38.07
	DEVIANT	24.63	16.54	14.52	61.00	46.00	40.18

dataset is 3.82, while the normalized focal length $\bar{f}$ for the nuScenes dataset is 2.82. Thus, the KITTI and the nuScenes images have a different focal length [84].

M3D-RPN [4] does not normalize w.r.t. the focal length. So, we explicitly correct and divide the depth predictions of nuScenes images from the KITTI model by $3.82/2.82 = 1.361$ in the M3D-RPN [4] codebase. The GUP Net [49] and DEVIANT codebases use normalized coordinates *i.e.* they normalize w.r.t. the focal length. So, we do not explicitly correct the focal length for GUP Net and DEVIANT predictions.

We match predictions to the ground truths using the IoU_{2D} overlap threshold of 0.7 [67]. After this matching, we calculate the Mean Average Error (MAE) of the depths of the predicted and the ground truth boxes [67].

Stress Test with Rotational and/or xy -translation Ego Movement. Corollary 1 uses translation along the depth as the sole ego movement. This assumption might be valid for the current outdoor datasets and benchmarks, but is not the case in the real world. Therefore, we conduct stress tests on how tolerable DEVIANT and GUP Net [49] are when there is rotational and/or xy -translation movement on the vehicle.

First, note that KITTI and Waymo are already large-scale real-world datasets, and our own dataset might not be a good choice. So, we stick with KITTI and Waymo datasets. We manually choose 306 KITTI Val images with such ego movements and again compare performance of DEVIANT and GUP Net on this subset in Tab. 18. The average distance of the car in this subset is 27.69 m (± 16.59 m), which suggests a good variance and unbiasedness in the subset. Tab. 18 shows that both the DEVIANT backbone and the CNN backbone show a drop in the detection performance by about 4 AP points on the Mod cars of ego-rotated subset compared to the all set. This drop experimentally confirms the theory that both the DEVIANT backbone and the CNN backbone do not handle arbitrary 3D rotations. More importantly, the table shows that DEVIANT maintains the performance improvement over GUP Net [49] under such movements.

Also, Waymo has many images in which the ego camera shakes. Improvements on Waymo (Tab. 12) also confirms that DEVIANT outperforms GUP Net [49] even when there is rotational or xy -translation ego movement.

Comparison of Depth Estimates from Monocular Depth Estimators and 3D Object Detectors. We next compare the depth estimates from monocular depth estimators and depth estimates from monocular 3D object detectors on the foreground objects. We take a monocular depth estimator BTS [41] model trained on KITTI Eigen split. We next compare the depth error for all and fore-

Table 19: **Comparison of Depth Estimates** of monocular depth estimators and 3D object detectors on KITTI Val cars. Depth from a depth estimator BTS is not good for foreground objects (cars) beyond 20+ m range. [Key: **Best**, **Second Best**]

Method	Depth at	Ground Truth	Back+ Foreground			Foreground (Cars)		
			0-20	20-40	40-∞	0-20	20-40	40-∞
GUP Net [49]	3D Center	3D Box	—	—	—	0.45	1.10	1.85
DEVIANT	3D Center	3D Box	—	—	—	0.40	1.09	1.80
BTS [41]	Pixel	LiDAR	0.48	1.30	1.83	0.30	1.22	2.16

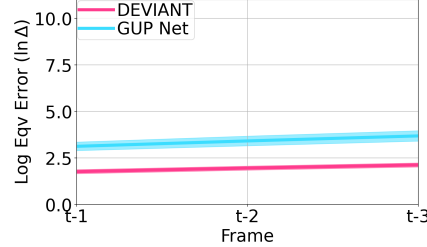


Fig. 11: **Equivariance error** (Δ) comparison for DEVIANT and GUP Net on previous three frames of the KITTI monocular videos at block 3 in the backbone.

ground objects (cars) on KITTI Val split using MAE ($\downarrow$) metric in Tab. 19 as in Tab. 6. We use the MSeg [39] to segment out cars in the driving scenes for BTS. Tab. 19 shows that the depth from BTS is not good for foreground objects (cars) beyond 20+ m range. Note that there is a data leakage issue between the KITTI Eigen train split and the KITTI Val split [69] and therefore, we expect more degradation in performance of monocular depth estimators after fixing the data leakage issue.

Equivariance Error for KITTI Monocular Videos. A better way to compare the scale equivariance of the DEVIANT and GUP Net [49] compared to Fig. 4, is to compare equivariance error on real images with depth translations of the ego camera. The equivariance error Δ is the normalized difference between the scaled feature map and the feature map of the scaled image, and is given by

$$\Delta = \frac{1}{N} \sum_{i=1}^N \frac{\|\mathcal{T}_{s_i} \Phi(h_i) - \Phi(\mathcal{T}_{s_i} h_i)\|_2^2}{\|\mathcal{T}_{s_i} \Phi(h_i)\|_2^2}, \quad (36)$$

where Φ denotes the neural network, $\mathcal{T}_{s_i}$ is the scaling transformation for the image i , and N is the total number of images. Although we do evaluate this error in Fig. 4, the image scaling in Fig. 4 does not involve scene change because of the absence of the moving objects. Therefore, evaluating on actual depth translations of the ego camera makes the equivariance error evaluation more realistic. We next carry out this experiment and report the equivariance error on three previous frames of the val images of the KITTI Val split as in [5]. We plot this equivariance error in Fig. 11 at block 3 of the backbones because the resolution at this block corresponds to the output feature map of size [96, 320]. Fig. 11 is similar to Fig. 4b, and shows that DEVIANT achieves lower equivariance error. Therefore,

Table 20: **Five Different Runs** on KITTI Val cars. [Key: **Average**]

Method	Run	IoU _{3D} ≥ 0.7						IoU _{3D} ≥ 0.5					
		AP _{3D R40} [%](†)			AP _{BEV R40} [%](†)			AP _{3D R40} [%](†)			AP _{BEV R40} [%](†)		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
GUP Net [49]	1	21.67	14.75	12.68	28.72	20.88	17.79	58.27	43.53	37.62	63.67	47.37	42.55
	2	21.26	14.94	12.49	28.39	20.40	17.43	59.20	43.55	37.63	64.06	47.46	42.67
	3	20.87	15.03	12.61	28.66	20.56	17.48	60.19	44.08	39.36	65.26	49.44	43.17
	4	21.10	15.48	12.88	28.58	20.92	17.83	58.95	43.99	38.07	64.60	47.76	42.97
	5	22.52	15.92	13.31	30.77	22.40	19.36	59.91	44.00	39.30	64.94	48.01	43.08
	Avg	21.48	15.22	12.79	29.02	21.03	17.98	59.30	43.83	38.40	64.51	48.01	42.89
DEVIANT	1	23.19	15.84	14.11	29.82	21.93	19.16	60.19	45.52	39.86	66.32	49.39	43.38
	2	23.33	16.12	13.54	31.22	22.64	19.64	61.59	46.33	40.35	67.49	50.26	43.98
	3	24.12	16.37	14.48	31.58	22.52	19.65	62.51	46.47	40.65	67.33	50.24	44.16
	4	24.63	16.54	14.52	32.60	23.04	19.99	61.00	46.00	40.18	65.28	49.63	43.50
	5	25.82	17.69	15.07	33.63	23.84	20.60	62.39	46.46	40.61	67.55	50.51	45.80
	Avg	24.22	16.51	14.34	31.77	22.79	19.81	61.54	46.16	40.33	66.79	50.01	44.16

 Table 21: **Experiments Comparison.**

Method	Venue	Multi-Dataset	Cross-Dataset	Multi-Backbone
GrooMeD-NMS [36]	CVPR21	—	—	—
MonoFlex [100]	CVPR21	—	—	—
CaDDN [62]	CVPR21	✓	—	—
MonoRCNN [67]	ICCV21	—	✓	—
GUP Net [49]	ICCV21	—	—	—
DD3D [57]	ICCV21	✓	—	✓
PCT [80]	NeurIPS21	✓	—	✓
MonoDistill [14]	ICLR22	—	—	—
MonoDIS-M [68]	TPAMI20	✓	—	—
MonoEF [103]	TPAMI21	✓	—	—
DEVIANT	—	✓	✓	✓

DEVIANT has better equivariance to depth translations (scale transformation s) than GUP Net [49] in real scenarios.

Model Size, Training, and Inference Times. Both DEVIANT and the baseline GUP Net have the same number of trainable parameters, and therefore, the same model size. GUP Net takes 4 hours to train on KITTI Val and 0.02 ms per image for inference on a single Ampere A100 (40 GB) GPU. DEVIANT takes 8.5 hours for training and 0.04 ms per image for inference on the same GPU. This is expected because SE models use more flops [74, 104] and, therefore, DEVIANT takes roughly twice the training and inference time as GUP Net.

Reproducibility. As described in Sec. 5.2, we now list out the five runs of our baseline GUP Net [49] and DEVIANT in Tab. 20. Tab. 20 shows that DEVIANT outperforms GUP Net in all runs and in the average run.

Experiment Comparison. We now compare the experiments of different papers in Tab. 21. To the best of our knowledge, the experimentation in DEVIANT is more than the experimentation of most monocular 3D object detection papers.

A3.2 Qualitative Results

KITTI. We next show some more qualitative results of models trained on KITTI Val split in Fig. 13. We depict the predictions of DEVIANT in image view on the left and the predictions of DEVIANT and GUP Net [49], and ground truth

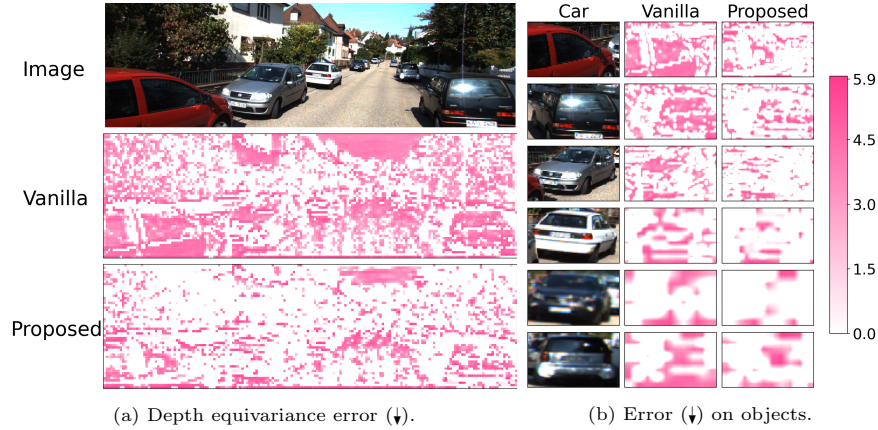


Fig. 12: (a) **Depth (scale) equivariance error** of vanilla GUP Net [49] and proposed DEVIANT. (See Sec. 5.2 for details) (b) **Error on objects**. The proposed backbone has less depth equivariance error than vanilla CNN backbone.

in BEV on the right. In general, DEVIANT predictions are more closer to the ground truth than GUP Net [49].

nuScenes Cross-Dataset Evaluation. We then show some qualitative results of KITTI Val model evaluated on nuScenes frontal in Fig. 14. We again observe that DEVIANT predictions are more closer to the ground truth than GUP Net [49]. Also, considerably less number of boxes are detected in the cross-dataset evaluation *i.e.* on nuScenes. We believe this happens because of the domain shift.

Waymo. We now show some qualitative results of models trained on Waymo Val split in Fig. 15. We again observe that DEVIANT predictions are more closer to the ground truth than GUP Net [49].

A3.3 Demo Videos of DEVIANT

Detection Demo. We next put a short demo video of our DEVIANT model trained on KITTI Val split at <https://www.youtube.com/watch?v=2D73ZBrU-PA>. We run our trained model independently on each frame of 2011_09_26_DRIVE_0009 KITTI raw [27]. The video belongs to the City category of the KITTI raw video. None of the frames from the raw video appear in the training set of KITTI Val split [36]. We use the camera matrices available with the video but do not use any temporal information. Overlaid on each frame of the raw input videos, we plot the projected 3D boxes of the predictions and also plot these 3D boxes in the BEV. We set the frame rate of this demo at 10 fps as in KITTI. The attached demo video demonstrates very stable and impressive results because of the additional equivariance to depth translations in DEVIANT which is absent in vanilla CNNs. Also, notice that the orientation of the boxes are stable despite not using any temporal information.

Equivariance Error Demo. We next show the depth equivariance (scale equivariance) error demo of one of the channels from the vanilla GUP Net and our

proposed method at <https://www.youtube.com/watch?v=70DIjQkuZvw>. As before, we report at block 3 of the backbones which corresponds to output feature map of the size [96, 320]. The equivariance error demo indicates more white spaces which confirms that DEVIANT achieves lower equivariance error compared to the baseline GUP Net [49]. Thus, this demo agrees with Fig. 12a. This happens because depth (scale) equivariance is additionally hard-baked into DEVIANT, while the vanilla GUP Net is not equivariant to depth translations (scale transformations).

Acknowledgements

This research was partially sponsored by the Ford Motor Company and the Army Research Office (ARO) grant W911NF-18-1-0330. This document’s views and conclusions are those of the authors and do not represent the official policies, either expressed or implied, of the Army Research Office or the U.S. government.

We deeply appreciate Max Welling from the University of Amsterdam for several pointers and discussions on equivariance and projective transformations. We also thank Ivan Sosnovik, Wei Zhu, Marc Finzi and Vidit for their inputs on scale equivariance.

Yan Lu and Yunfei Long helped setting up the GUP Net [49] and DETR3D [85] codebases respectively. Xuepeng Shi and Li Wang shared details of their cross-dataset evaluation [67] and Waymo experiments [80] respectively. Shengjie Zhu helped us with the monocular depth (BTS) [41] experiments. Shengjie Zhu, Vishal Asnani, Yiyang Su, Masa Hu, Andrew Hou and Arka Sadhu proof-read our manuscript.

We finally thank anonymous CVPR and ECCV reviewers for their feedback that shaped the final manuscript. One anonymous CVPR reviewer pointed out that Theorem 1 exists as Example 13.2 in [30], which we had wrongly claimed as ours in an earlier version.



Fig. 13: **KITTI Qualitative Results.** DEVIANT predictions in general are more accurate than GUP Net [49]. [Key: Cars, Cyclists and Pedestrians of DEVIANT; all classes of GUP Net, and Ground Truth in BEV].

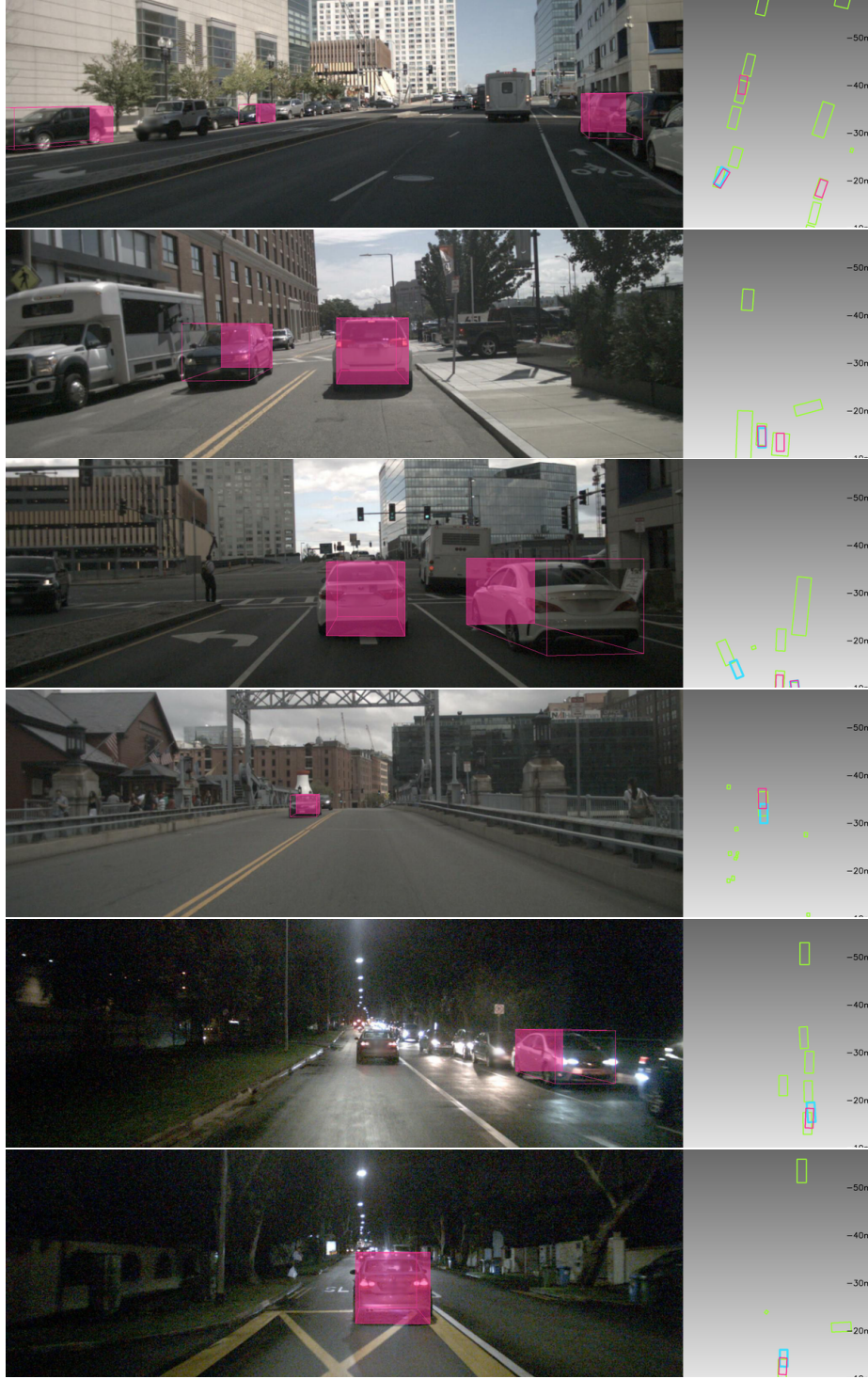


Fig. 14: **nuScenes Cross-Dataset Qualitative Results.** DEVIANT predictions in general are more accurate than GUP Net [49]. [Key: Cars of DEVIANT; Cars of GUP Net, and Ground Truth in BEV].

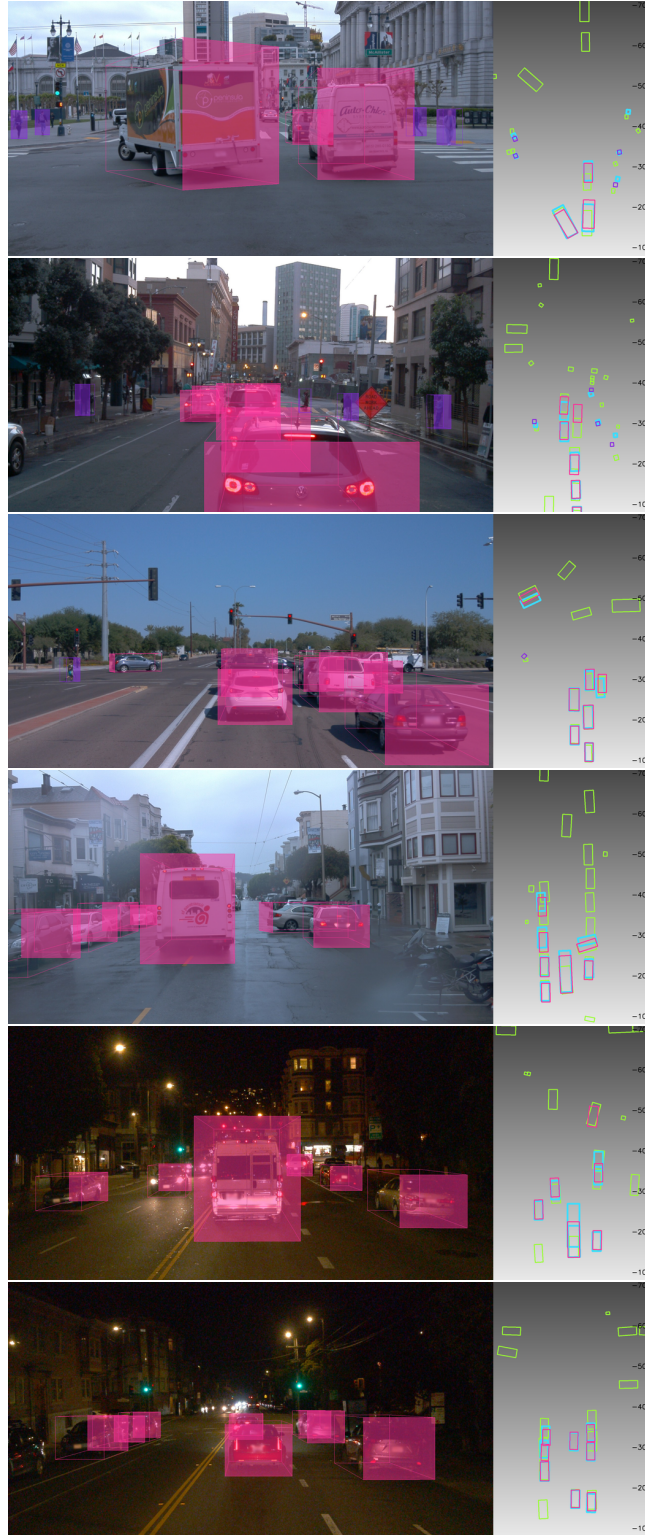


Fig. 15: **Waymo Qualitative Results.** DEVIANT predictions in general are more accurate than GUP Net [49]. [Key: Cars, Cyclists and Pedestrians of DEVIANT; all classes of GUP Net, and Ground Truth in BEV].