SeaBird: Segmentation in Bird's View with Dice Loss Improves Monocular 3D Detection of Large Objects

Supplementary Material

Contents

A1. Additional Explanations and Proofs

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

A1.1. Proof of Converged Value

We first bound the converged value from the optimal value. These results are well-known in the literature [44, 85]. We reproduce the result from using our notations for completeness.

$$
\begin{aligned}
&\mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \\
&= \mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu} + {}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*\right\|_2^2\right) \\
&= \mathbb{E}\left(\left({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu} + {}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*\right)^T \left({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu} + {}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*\right)\right) \\
&= \mathbb{E}(({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu})^T({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu})) + \mathbb{E}(({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)^T({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)) \\
&\quad + 2\mathbb{E}(({}^{\mathcal{L}}\mathbf{w}_\infty - {}^{\mathcal{L}}\boldsymbol{\mu})^T({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)) \\
&= \mathrm{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) + \mathbb{E}(({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)^T({}^{\mathcal{L}}\boldsymbol{\mu} - \mathbf{w}_*)) \qquad (5)
\end{aligned}
$$

where ${}^{\mathcal{L}}\boldsymbol{\mu} = \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_\infty)$ is the mean of the layer weight and $\mathrm{Var}(\mathbf{w})$ denotes the variance of $\sum_j w_j^2$.

SGD. We begin the proof by writing the value of ${}^{\mathcal{L}}\mathbf{w}_t$ at every step. The model uses SGD, and so, the weight ${}^{\mathcal{L}}\mathbf{w}_t$ after t gradient updates is

$$
{}^{\mathcal{L}}\mathbf{w}_t = \mathbf{w}_0 - s_1\,{}^{\mathcal{L}}\mathbf{g}_1 - s_2\,{}^{\mathcal{L}}\mathbf{g}_2 - \cdots - s_t\,{}^{\mathcal{L}}\mathbf{g}_t, \qquad (6)
$$

where ${}^{\mathcal{L}}\mathbf{g}_t$ denotes the gradient of $\mathbf{w}$ at every step t. Assume the loss function under consideration $\mathcal{L}$ is $\mathcal{L} = f(\mathbf{w}_t\mathbf{h} - z) = f(\eta)$. Then, we have,

$$
\begin{aligned}
{}^{\mathcal{L}}\mathbf{g}_t &= \frac{\partial \mathcal{L}}{\partial \mathbf{w}_t} \\
&= \frac{\partial \mathcal{L}(\mathbf{w}_t\mathbf{h} - z)}{\partial \mathbf{w}_t} \\
&= \frac{\partial \mathcal{L}(\mathbf{w}_t\mathbf{h} - z)}{\partial (\mathbf{w}_t\mathbf{h} - z)} \frac{\partial (\mathbf{w}_t\mathbf{h} - z)}{\partial \mathbf{w}_t} \\
&= \frac{\partial \mathcal{L}(\eta)}{\partial \eta}\mathbf{h} \\
&= \mathbf{h}\frac{\partial \mathcal{L}(\eta)}{\partial \eta} \\
\implies {}^{\mathcal{L}}\mathbf{g}_t &= \mathbf{h}\epsilon, \qquad (7)
\end{aligned}
$$

with $\epsilon = \dfrac{\partial \mathcal{L}(\eta)}{\partial \eta}$ is the gradient of the loss function wrt noise.

Expectation and Variance of Gradient ${}^{\mathcal{L}}\mathbf{g}_t$ Since the image $\mathbf{h}$ and noise η are statistically independent, the image and the noise gradient η are also statistically independent. So, the expected gradients

$$
\mathbb{E}({}^{\mathcal{L}}\mathbf{g}_t) = \mathbb{E}(\mathbf{h})\mathbb{E}(\epsilon) = 0. \qquad (8)
$$

Note that if the loss function is an even function (symmetric about zero), its gradient ϵ is an odd function (anti-symmetric about 0), and so its mean $\mathbb{E}(\epsilon) = 0$.

Next, we write the gradient variance $\mathrm{Var}({}^{\mathcal{L}}\mathbf{g}_t)$ as

$$
\begin{aligned}
\mathrm{Var}({}^{\mathcal{L}}\mathbf{g}_t) = \mathrm{Var}(\mathbf{h}\epsilon) &= \mathbb{E}(\mathbf{h}^T\mathbf{h})\mathbb{E}(\epsilon^2) - \mathbb{E}^2(\mathbf{h})\mathbb{E}^2(\epsilon) \\
&= \mathbb{E}(\mathbf{h}^T\mathbf{h})\left[\mathrm{Var}(\epsilon) + \mathbb{E}^2(\epsilon)\right] \\
&\quad - \mathbb{E}^2(\mathbf{h})\mathbb{E}^2(\epsilon) \\
\implies \mathrm{Var}({}^{\mathcal{L}}\mathbf{g}_t) &= \mathbb{E}(\mathbf{h}^T\mathbf{h})\mathrm{Var}(\epsilon) \quad \text{as } \mathbb{E}(\epsilon) = 0 \qquad (9)
\end{aligned}
$$

Expectation and Variance of Converged Weight ${}^{\mathcal{L}}\mathbf{w}_t$ We first calculate the expected converged weight as

$$
\begin{aligned}
\mathbb{E}({}^{\mathcal{L}}\mathbf{w}_t) &= \mathbb{E}(\mathbf{w}_0) + \left(\sum_{j=1}^{t} s_j \mathbb{E}\left({}^{\mathcal{L}}\mathbf{g}_j\right)\right), \text{using Eq. (6)} \\
&= \mathbf{0} \quad \text{using Eq. (8)}
\end{aligned}
$$

$$\implies \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_\infty) = \lim_{t\to\infty} \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_t)$$
$$\implies \mathbb{E}({}^{\mathcal{L}}\mathbf{w}_\infty) = {}^{\mathcal{L}}\boldsymbol{\mu} = \mathbf{0} \tag{10}$$

We finally calculate the variance of the converged weight. Because the SGD step size is independent of the gradient, we write using Eq. (6),

$$\begin{aligned}\text{Var}({}^{\mathcal{L}}\mathbf{w}_t) = \text{Var}(\mathbf{w}_0) + s_1^2\text{Var}\left(\mathbf{g}_1\right) + s_2^2\text{Var}\left(\mathbf{g}_2\right) \\ + \cdots + s_t^2\text{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right)\end{aligned} \tag{11}$$

Assuming the gradients ${}^{\mathcal{L}}\mathbf{g}_t$ are drawn from an identical distribution, we have

$$\begin{aligned}&\text{Var}({}^{\mathcal{L}}\mathbf{w}_t) = \text{Var}(\mathbf{w}_0) + \left(\sum_{j=1}^{t} s_j^2\right)\text{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right) \\ \implies &\text{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) = \lim_{t\to\infty}\text{Var}({}^{\mathcal{L}}\mathbf{w}_t) \\ &= \text{Var}(\mathbf{w}_0) + \left(\lim_{t\to\infty}\sum_{j=1}^{t} s_j^2\right)\text{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right) \\ \implies &\text{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) = \text{Var}(\mathbf{w}_0) + s\text{Var}\left({}^{\mathcal{L}}\mathbf{g}_t\right)\end{aligned} \tag{12}$$

An example of square summable step-sizes of SGD is $s_j = \frac{1}{j}$, and then the constant $s = \sum_{j=1} s_j^2 = \frac{\pi^2}{6}$. This assumption is also satisfied by modern neural networks since their training steps are always finite.

Substituting Eq. (9) in Eq. (12), we have

$$\text{Var}({}^{\mathcal{L}}\mathbf{w}_\infty) = \text{Var}(\mathbf{w}_0) + s\mathbb{E}(\mathbf{h}^T\mathbf{h})\text{Var}(\epsilon) \tag{13}$$

Substituting mean and variances from Eqs. (10) and (13) in Eq. (5), we have

$$\begin{aligned}\mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) &= \text{Var}(\mathbf{w}_0) + s\mathbb{E}(\mathbf{h}^T\mathbf{h})\text{Var}(\epsilon) \\ &\quad + \mathbb{E}(||\mathbf{w}_*||^2) \\ &= s\mathbb{E}(\mathbf{h}^T\mathbf{h})\text{Var}(\epsilon) + \text{Var}(\mathbf{w}_0) \\ &\quad + \mathbb{E}(||\mathbf{w}_*||^2) \\ \implies \mathbb{E}\left(\left\|{}^{\mathcal{L}}\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) &= c_1\text{Var}(\epsilon) + c_2,\end{aligned} \tag{14}$$

where $\epsilon = \dfrac{\partial\mathcal{L}(\eta)}{\partial\eta}$ is the gradient of the loss function wrt noise, and $c_1 = s\mathbb{E}(\mathbf{h}^T\mathbf{h})$ and c_2 are terms independent of the loss function $\mathcal{L}$.

A1.2. Comparison of Loss Functions

Eq. (1) shows that different losses $\mathcal{L}$ lead to different $\text{Var}(\epsilon)$. Hence, comparing this term for different losses asseses the quality of losses.

A1.2.1 Gradient Variance of MAE Loss

The result on MAE ($\mathcal{L}_1$) is well-known in the literature [44, 85]. We reproduce the result from [44, 85] using our notations for completeness.

The $\mathcal{L}_1$ loss is

$$\begin{aligned}&\mathcal{L}_1(\eta) = |\hat{z} - z|_1 = |{}^{\mathcal{L}}\mathbf{w}_t\mathbf{h} - z|_1 = |\eta|_1 \\ \implies &\epsilon = \frac{\partial\mathcal{L}_1(\eta)}{\partial\eta} = \text{sgn}(\eta)\end{aligned} \tag{15}$$

Thus, $\epsilon = \text{sgn}(\eta)$ is a Bernoulli random variable with $p(\epsilon) = 1/2$ for $\epsilon = \pm 1$. So, mean $\mathbb{E}(\epsilon) = 0$ and variance $\text{Var}(\epsilon) = 1$.

A1.2.2 Gradient Variance of MSE Loss

The result on MSE ($\mathcal{L}_2$) is well-known in the literature [44, 85]. We reproduce the result from [44, 85] using our notations for completeness. The $\mathcal{L}_2$ loss is

$$\begin{aligned}&\mathcal{L}_2(\eta) = 0.5|\hat{z} - z|^2 = 0.5|\eta|^2 = 0.5\eta^2 \\ \implies &\epsilon = \frac{\partial\mathcal{L}_2(\eta)}{\partial\eta} = \eta\end{aligned} \tag{16}$$

Thus, $\epsilon = \eta$ is a normal random variable [85]. So, mean $\mathbb{E}(\epsilon) = 0$ and variance $\text{Var}(\epsilon) = \text{Var}(\eta) = \sigma^2$.

A1.2.3 Gradient Variance of Dice Loss. (Proof of Lemma 2)

Proof. We first write the gradient of dice loss as a function of noise (η) as follows:

$$\epsilon = \frac{\partial\mathcal{L}_{dice}(\eta)}{\partial\eta} = \begin{cases}\dfrac{\text{sgn}(\eta)}{\ell} & , |\eta| \le \ell \\ 0 & , |\eta| \ge \ell\end{cases} \tag{17}$$

The gradient of the loss ϵ is an odd function and so, its mean $\mathbb{E}(\epsilon) = 0$. Next, we write its variance $\text{Var}(\epsilon)$ as

$$\begin{aligned}\text{Var}(\epsilon) = \text{Var}(\eta) &= \frac{1}{\ell^2}\int_{-\ell}^{\ell}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{\eta^2}{2\sigma^2}}d\eta \\ &= \frac{2}{\ell^2}\int_{0}^{\ell}\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{\eta^2}{2\sigma^2}}d\eta \\ &= \frac{2}{\ell^2}\int_{0}^{\ell/\sigma}\frac{1}{\sqrt{2\pi}}e^{-\frac{\eta^2}{2}}d\eta \\ &= \frac{2}{\ell^2}\left[\int_{-\infty}^{\ell/\sigma}\frac{1}{\sqrt{2\pi}}e^{-\frac{\eta^2}{2}}d\eta - \frac{1}{2}\right]\end{aligned}$$

$$= \frac{2}{\ell^2}\left[\Phi\left(\frac{\ell}{\sigma}\right) - \frac{1}{2}\right] \tag{18}$$

where, Φ is the normal CDF

We write the CDF $\Phi(x)$ in terms of error function Erf as:

$$\Phi(x) = \frac{1}{2} + \frac{1}{2}\text{Erf}\left(\frac{x}{\sqrt{2}}\right) \tag{19}$$

for $x \geq 0$. Next, we put $x = \frac{\ell}{\sigma}$ to get

$$\Phi\left(\frac{\ell}{\sigma}\right) = \frac{1}{2} + \frac{1}{2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \tag{20}$$

Substituting above in Eq. (18), we obtain

$$\begin{aligned} \text{Var}(\epsilon) &= \frac{2}{\ell^2}\left[\frac{1}{2} + \frac{1}{2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) - \frac{1}{2}\right] \\ \implies \text{Var}(\epsilon) &= \frac{1}{\ell^2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \end{aligned} \tag{21}$$

A1.3. Proof of Lemma 3

Proof. It remains sufficient to show that

$$\begin{aligned} &\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2\right) \leq \mathbb{E}\left(\|{}^r\mathbf{w}_\infty - \mathbf{w}_*\|_2\right) \\ \implies &\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \leq \mathbb{E}\left(\|{}^r\mathbf{w}_\infty - \mathbf{w}_*\|_2^2\right) \end{aligned} \tag{22}$$

Using Lemma 1, the above comparison is a comparison between the gradient variance of the loss wrt noise $\text{Var}(\epsilon)$. Hence, we compute the gradient variance of the loss $\mathcal{L}$, *i.e.*, $\text{Var}(\epsilon)$ of regression and dice losses to derive this lemma.

Case 1 $\sigma \leq 1$**:** Given Tab. 1, if $\sigma \leq 1$, the minimum deviation in converged regression model comes from the $\mathcal{L}_2$ loss. The difference in the estimates of regression loss and the dice loss

$$\begin{aligned} &\mathbb{E}\left(\|{}^r\mathbf{w}_\infty - \mathbf{w}_*\|_2^2\right) - \mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \\ &\propto \sigma^2 - \frac{1}{\ell^2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \end{aligned} \tag{23}$$

Let σ_m be the solution of the equation $\sigma^2 = \frac{1}{\ell^2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right)$. Note that the above equation has unique solution σ_m since σ^2 is a strictly increasing function wrt σ for $\sigma > 0$, while $\frac{1}{\ell^2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right)$ is a strictly decreasing function wrt σ for $\sigma > 0$. If the noise has $\sigma \geq \sigma_m$, the RHS of the above equation ≥ 0, which means dice loss converges better than the regression loss.

Case 2 $\sigma \geq 1$**:** Given Tab. 1, if $\sigma \geq 1$, the minimum deviation in converged regression model comes from the $\mathcal{L}_1$ loss. The difference in the regression and dice loss estimates:

$$\begin{aligned} &\mathbb{E}\left(\|{}^r\mathbf{w}_\infty - \mathbf{w}_*\|_2^2\right) - \mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \\ &\propto 1 - \frac{1}{\ell^2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \end{aligned} \tag{24}$$

If the noise has $\sigma \geq \frac{\sqrt{2}}{\ell}\text{Erf}^{-1}(\ell^2)$, the RHS of the above equation ≥ 0, which means dice loss is better than the regression loss. For objects such as cars and trailers which have length $\ell > 4m$, this is trivially satisfied.

Combining both cases, dice loss outperforms the $\mathcal{L}_1$ and $\mathcal{L}_2$ losses if the noise deviation σ exceeds the critical threshold σ_c, *i.e.*

$$\sigma > \sigma_c = \max\left(\sigma_m, \frac{\sqrt{2}}{\ell}\text{Erf}^{-1}(\ell^2)\right). \tag{25}$$

A1.4. Proof of Theorem 1

Proof. Continuing from Lemma 3, the advantage of the trained weight obtained from dice loss over the trained weight obtained from regression losses further results in

$$\begin{aligned} &\text{Var}({}^d\mathbf{w}_\infty) \leq \text{Var}({}^r\mathbf{w}_\infty) \\ \implies &\mathbb{E}(|{}^d\mathbf{w}_\infty\mathbf{h} - z|) \leq \mathbb{E}(|{}^r\mathbf{w}_\infty\mathbf{h} - z|) \\ \implies &\mathbb{E}(|\,{}^d\hat{z} - z|) \leq \mathbb{E}(|\,{}^r\hat{z} - z|) \\ \implies &\mathbb{E}({}^d\text{IoU}_{3D}) \geq \mathbb{E}({}^r\text{IoU}_{3D}), \end{aligned} \tag{26}$$

assuming depth is the only source of error. Because AP_{3D} is an non-decreasing function of IoU_{3D}, the inequality remains preserved. Hence, we have ${}^d\text{AP}_{3D} \geq {}^r\text{AP}_{3D}$. □

Thus, the average precision from the dice model is better than the regression model, which means a better detector.

A1.5. Properties of Dice Loss.

We next explore the properties of model in Lemma 3 trained with dice loss. From Lemma 1, we write

$$\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) = c_1\text{Var}(\epsilon) + c_2$$

Substituting the result of Lemma 2, we have

$$\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) = \frac{c_1}{\ell^2}\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) + c_2 \tag{27}$$

Paper [3] says that for a normal random variable X with mean 0 and variance 1 and for any $x > 0$, we have

$$\begin{aligned} &\frac{\sqrt{4+x^2} - x}{2}\sqrt{\frac{1}{2\pi}}e^{-\frac{x^2}{2}} \leq P(X > x) \\ \implies &\frac{1}{x + \sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq P(X > x) \\ \implies &\frac{1}{x + \sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq 1 - P(X \leq x) \end{aligned}$$

Table 8. **Assumption comparison** of Theorem 1 vs Mono3D models.

	Theorem 1	Mono3D Models
Regression	Linear	Non-linear
Noise η PDF	Normal	Arbitrary
Noise & Image	Independent	Dependent
Object Categories	1	Multiple
Object Size ℓ	Ideal	Non-ideal
Error	Depth	All 7 parameters
Loss $\mathcal{L}$	$\mathcal{L}_1, \mathcal{L}_2$, dice	Smooth $\mathcal{L}_1, \mathcal{L}_2$, dice, CE
Optimizers	SGD	SGD, Adam, AdamW
Global Optima	Unique	Multiple

$$\implies \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq 1-\frac{1}{2}-\int_0^x \frac{1}{\sqrt{2\pi}}e^{-\frac{X^2}{2}}dX$$

$$\implies \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq \frac{1}{2}-\int_0^x \frac{1}{\sqrt{2\pi}}e^{-\frac{X^2}{2}}dX$$

$$\implies \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq \frac{1}{2}-\int_0^{\frac{x}{\sqrt{2}}} \frac{1}{\sqrt{\pi}}e^{-X^2}dX$$

$$\implies \frac{1}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}} \leq \frac{1}{2}-\frac{1}{2}\text{Erf}\left(\frac{x}{\sqrt{2}}\right)$$

$$\implies \text{Erf}\left(\frac{x}{\sqrt{2}}\right) \leq 1-\frac{2}{x+\sqrt{4+x^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{x^2}{2}}$$

Substituting $x=\frac{\ell}{\sigma}$ above, we have,

$$\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \leq 1-\frac{2\sigma}{\ell+\sqrt{4\sigma^2+\ell^2}}\sqrt{\frac{2}{\pi}}e^{-\frac{\ell^2}{2\sigma^2}} \tag{28}$$

Case 1: Upper bound. The RHS of Eq. (28) is clearly less than 1 since the term in the RHS after subtraction is positive. Hence,

$$\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \leq 1$$

Substituting above in Eq. (27), we have

$$\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \leq \frac{c_1}{\ell^2}+c_2 \tag{29}$$

Clearly, the deviation of the trained model with the dice loss is inversely proportional to the object length ℓ. The deviation from the optimal is less for large objects.

Case 2: Infinite Noise variance $\sigma^2 \to \infty$. Then, one of the terms in the RHS of Eq. (28) $\frac{2\sigma}{\ell+\sqrt{4\sigma^2+\ell^2}} \to 1$. Moreover, $\frac{\ell}{\sigma} \to 0 \implies e^{-\frac{\ell^2}{2\sigma^2}} \approx \left(1-\frac{\ell^2}{2\sigma^2}\right)$. So, RHS of Eq. (28) becomes

$$\text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \approx 1-\sqrt{\frac{2}{\pi}}\left(1-\frac{\ell^2}{2\sigma^2}\right)$$

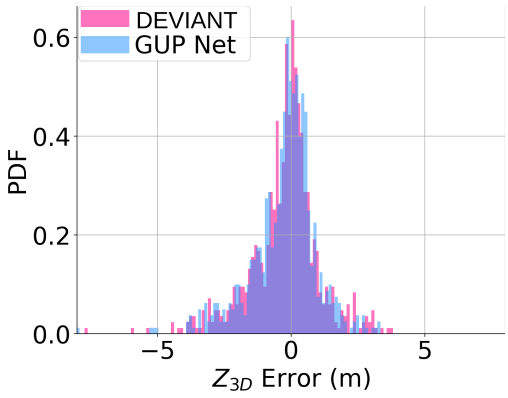

Figure 6. **Depth error histogram** of released GUP Net and DEVIANT [43] on the KITTI Val cars. The histogram shows that depth error is close to the Gaussian random variable.

$$\implies \text{Erf}\left(\frac{\ell}{\sqrt{2}\sigma}\right) \approx \left(1+\sqrt{\frac{2}{\pi}}+\sqrt{\frac{2}{\pi}}\frac{\ell^2}{2\sigma^2}\right) \tag{30}$$

Substituting above in Eq. (27), we have

$$\mathbb{E}\left(\left\|{}^d\mathbf{w}_\infty - \mathbf{w}_*\right\|_2^2\right) \approx \frac{c_1}{\ell^2}\left(1+\sqrt{\frac{2}{\pi}}+\sqrt{\frac{2}{\pi}}\frac{\ell^2}{2\sigma^2}\right) + c_2 \tag{31}$$

Thus, the deviation from the optimal weight is inversely proportional to the noise deviation σ^2. Hence, the deviation from the optimal weight decreases as σ^2 increases for the dice loss. This property provides noise-robustness to the model trained with the dice loss.

A1.6. Notes on Theoretical Result

Assumption Comparisons. The theoretical result of Theorem 1 relies upon several assumptions. We present a comparison between the assumptions made by Theorem 1 and those underlying Mono3D models, in Tab. 8. While our analysis depends on these assumptions, it is noteworthy that the results are apparent even in scenarios where the assumptions do not hold true. Another advantage of having a linear regression setup is that this setup has a unique global minima (because of its convexity).

Nature of Noise η**.** Theorem 1 assumes that the noise η is a normal random variable $\mathcal{N}(0,\sigma^2)$. To verify this assumption, we take the two SoTA released models GUP Net [63] and DEVIANT [43] on the KITTI [25] Val cars. We next plot the depth error histogram of both these models in Fig. 6. This figure confirms that the depth error is close to the Gaussian random variable. Thus, this assumption is quite realistic.

Theorem 1 Requires Assumptions? We agree that Theorem 1 requires assumptions for the proof. However, our

theory does have empirical support; most Mono3D works have no theory. So, our theoretical attempt for Mono3D is a step forward! We leave the analysis after relaxing some or all of these assumptions for future avenues.

Does Theorem 1 Hold in Inference? Yes, Theorem 1 holds even in inference. Theorem 1 relies on the converged weight ${}^{\mathcal{L}}\mathbf{w}_{\infty}$, which in turn depends on the training data distribution. Now, as long as the training and testing data distribution remains the same (a fundamental assumption in ML), Theorem 1 holds also during inference.

A1.7. More Discussions

SeaBird improves because it removes depth estimation and integrates BEV segmentation. We clarify to remove this confusion. First, SeaBird also estimates depth. SeaBird depth estimates are better because of good segmentation, a *form* of depth (thanks to dice loss). Second, predicted BEV segmentation needs processing with the 3D head to output depth; so it can not replace depth estimation. Third, integrating segmentation over all categories degrades Mono3D performance ([50] and our Tab. 5 Sem. Category).

Why evaluation on outdoor datasets? We experiment with outdoor datasets in this paper because indoor datasets rarely have large objects (mean length $> 6m$).

A2. Implementation Details

Datasets. Our experiments use the publicly available KITTI-360, KITTI-360 PanopticBEV and nuScenes datasets. KITTI-360 is available at https://www.cvlibs.net/datasets/kitti-360/download.php under CCA-NonCommercial-ShareAlike (CC BY-NC-SA) 3.0 License. KITTI-360 PanopticBEV is available at http://panoptic-bev.cs.uni-freiburg.de/ under Robot Learning License Agreement. nuScenes is available at https://www.nuscenes.org/nuscenes under CC BY-NC-SA 4.0 International Public License.

Data Splits. We detail out the detection data split construction of the KITTI-360 dataset.

- *KITTI-360 Test split*: This detection benchmark [52] contains 300 training and 42 testing windows. These windows contain 61,056 training and 9,935 testing images. The calibration exists for each frame in training, while it exists for every 10^{th} frame in testing. Therefore, our split consists of 61,056 training images, while we run monocular detectors on 910 test images (ignoring uncalibrated images).
- *KITTI-360 Val split*: The KITTI-360 detection Val split partitions the official train into 239 train and 61 validation windows [52]. The original Val split [52] contains 49,003 training and 14,600 validation images. However, this original Val split has the following three issues:

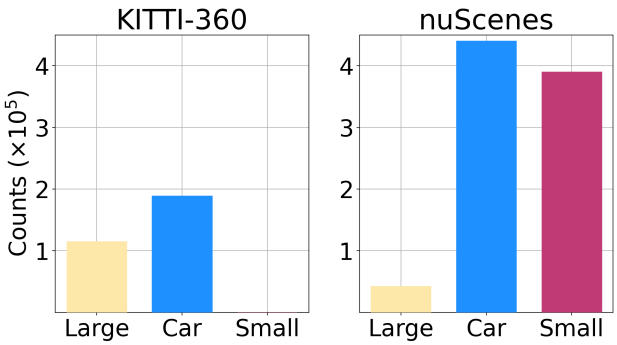

Figure 7. **Skewness in datasets.** The ratio of large objects to other objects is approximately $1:2$ in KITTI-360 [52], while the skewness is about $1:21$ in nuScenes [7].

  - Data leakage (common images) exists in the training and validation windows.
  - Every KITTI-360 image does not have the corresponding BEV semantic segmentation GT in the KITTI-360 PanopticBEV [27] dataset, making it harder to compare Mono3D and BEV segmentation performance.
  - The KITTI-360 validation set has higher sampling rate compared to the testing set.

  To fix the data leakage issue, we remove the common images from training set and keep them only in the validation set. Then, we take the intersection of KITTI-360 and KITTI-360 PanopticBEV datasets to ensure that every image has corresponding BEV segmentation segmentation GT. After these two steps, the training and validation set contain 48,648 and 12,408 images with calibration and semantic maps. Next, we subsample the validation images by a factor of 10 as in the testing set. Hence, our KITTI-360 Val split contains 48,648 training images and 1,294 validation images.

Augmentation. We keep the same augmentation strategy as our baselines for the respective models.

Pre-processing. We resize images to preserve their aspect ratio.

- *KITTI-360.* We resize the $[376, 1408]$ sized KITTI-360 images, and bring them to the $[384, 1438]$ resolution.
- *nuScenes.* We resize the $[900, 1600]$ sized nuScenes images, and bring them to the $[256, 704]$, $[512, 1408]$ and $[640, 1600]$ resolutions as our baselines [116, 121].

Libraries. I2M and PBEV experiments use PyTorch [77], while BEVerse and HoP use MMDetection3D [18].

Architecture.

- *I2M+SeaBird.*I2M [83] uses ResNet-18 as the backbone with the standard Feature Pyramid Network (FPN) [53] and a transformer to predict depth distribution. 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 four feature maps levels in this FPN. We use the

Box Net with ResNet-18 [29] as the detection head.

- *PBEV+SeaBird.*PBEV [27] uses EfficientDet [91] as the backbone. We use Box Net with ResNet-18 [29] as the detection head.
- *BEVerse+SeaBird.* BEVerse [116] uses Swin transformers [59] as the backbones. We use the original heads without any configuration change.
- *HoP+SeaBird.* HoP [121] uses ResNet-50, ResNet-101 [29] and V2-99 [74] as the backbones. Since HoP does not have the segmentation head, we use the one in BEVerse as the segmentation head.

We initialize the CNNs and transformers from ImageNet weights except for V2-99, which is pre-trained on 15 million LiDAR data.. We output two and ten foreground categories for KITTI-360 and nuScenes datasets respectively.

Training. We use the training protocol as our baselines for all our experiments. We choose the model saved in the last epoch as our final model for all our experiments.

- *I2M+SeaBird.* Training uses the Adam optimizer [38], a batch size of 30, an exponential decay of 0.98 [83] and gradient clipping of 10 on single Nvidia A100 (80GB) GPU. We train the BEV Net in the first stage with a learning rate 1.0×10^{-4} for 50 epochs [83] . We then add the detector in the second stage and finetune with the first stage weight with a learning rate $0.5{\times}10^{-4}$ for 40 epochs. Training on KITTI-360 Val takes a total of 100 hours. For Test models, we finetune I2M Val stage 1 model with train+val data for 40 epochs.
- *PBEV+SeaBird.* Training uses the Adam optimizer [38] with Nesterov, a batch size of 2 per GPU on eight Nvidia RTX A6000 (48GB) GPU. We train the PBEV with the dice loss in the first stage with a learning rate 2.5×10^{-3} for 20 epochs. We then add the Box Net in the second stage and finetune with the first stage weight with a learning rate $2.5{\times}10^{-3}$ for 20 epochs. PBEV decays the learning rate by 0.5 and 0.2 at 10 and 15 epoch respectively. Training on KITTI-360 Val takes a total of 80 hours. For Test models, we finetune PBEV Val stage 1 model with train+val data for 10 epochs on four GPUs.
- *BEVerse+SeaBird.* Training uses the AdamW optimizer [62], a sample size of 4 per GPU, the one-cycle policy [116] and gradient clipping of 35 on eight Nvidia RTX A6000 (48GB) GPU [116]. We train the segmentation head in the first stage with a learning rate $2.0{\times}10^{-3}$ for 4 epochs. We then add the detector in the second stage and finetune with the first stage weight with a learning rate 2.0×10^{-3} for 20 epochs [116]. Training on nuScenes takes a total of 400 hours.
- *HoP+SeaBird.* Training uses the AdamW optimizer [62], a sample size of 2 per GPU, and gradient clipping of 35 on eight Nvidia A100 (80GB) GPUs [121]. We train the segmentation head in the first stage with a learning rate 1.0×10^{-4} for 4 epochs. We then add the detector in the second stage and finetune with the first stage weight with a learning rate 1.0×10^{-4} for 24 epochs [116]. nuScenes training takes a total of 180 hours. For Test models, we finetune val model with train+val data for 4 more epochs.

Table 9. **Error analysis** on KITTI-360 Val.

Oracle							AP_{3D} 50 [%](↑)			AP_{3D} 25 [%](↑)		
x	y	z	l	w	h	θ	AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP
							8.71	43.19	25.95	35.76	52.22	43.99
✓							9.78	41.63	25.70	36.07	50.63	43.35
	✓						9.57	46.08	27.82	34.65	53.03	43.84
		✓					9.90	42.32	27.11	39.66	53.08	46.37
✓	✓	✓					19.90	47.37	33.63	41.84	52.53	47.19
			✓	✓	✓		9.49	45.67	27.58	33.43	51.53	42.48
✓	✓	✓	✓	✓	✓		37.09	46.27	41.68	44.58	51.15	47.87
✓	✓	✓	✓	✓	✓	✓	37.02	47.03	42.02	44.46	51.50	47.98

Losses. We train the BEV Net of SeaBird in Stage 1 with the dice loss. We train the final SeaBird pipeline in Stage 2 with the following loss:

$$\mathcal{L} = \mathcal{L}_{det} + \lambda_{seg}\mathcal{L}_{seg}, \tag{32}$$

with $\mathcal{L}_{seg}$ being the dice loss and λ_{seg} being the weight of the dice loss in the baseline. We keep the $\lambda_{seg} = 5$. If the segmentation loss is itself scaled such as PBEV uses the $\mathcal{L}_{seg}$ as 7, we use $\lambda_{seg} = 35$ with detection.

Inference. We report the performance of all KITTI-360 and nuScenes models by inferring on single GPU card. 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 is 50 per image for KITTI-360 models. We use score threshold of 0.1 for KITTI-360 models and class dependent threshold for nuScenes models as in [116]. KITTI-360 evaluates on windows and not on images. So, we use a 3D center-based NMS [42] to convert image-based predictions to window-based predictions for SeaBird and all our KITTI-360 baselines. This NMS uses a threshold of 4m for all categories, and keeps the highest score 3D box if multiple 3D boxes exist inside a window.

A3. Additional Experiments and Results

We now provide additional details and results of the experiments evaluating SeaBird 's performance.

A3.1. KITTI-360 Val Results

Error Analysis. We next report the error analysis of the SeaBird in Tab. 9 by replacing the predicted box data with the oracle box data as in [66]. We consider the GT box to be an oracle box for predicted box if the euclidean distance is less than $4m$. In case of multiple GT being matched to one box, we consider the oracle with the minimum distance. Tab. 9 shows that depth is the biggest source of error

Table 10. **Complexity analysis** on KITTI-360 Val.

Method	Mono3D	Inf. Time (s)	Param (M)	Flops (G)
GUP Net [63]	✓	0.02	16	30
DEVIANT [43]	✓	0.04	16	235
I2M [83]	×	0.01	40	80
I2M+SeaBird	✓	0.02	53	130
PBEV [27]	×	0.14	24	229
PBEV+SeaBird	✓	0.15	37	279

for Mono3D task as also observed in [66]. Moreover, the oracle does not lead to perfect results since the KITTI-360 PanopticBEV GT BEV semantic is only upto $50m$, while the KITTI-360 evaluates all objects (including objects beyond $50m$).

Computational Complexity Analysis. We next compare the complexity analysis of SeaBird pipeline in Tab. 10. For the flops analysis, we use the fvcore library as in [43].

Naive baseline for Large Objects. We next compare SeaBird against a naive baseline for large objects detection, such as by fine-tuning GUP Net only on larger objects. Tab. 11 shows that SeaBird pipelines comfortably outperform this baseline as well.

Does denoising BEV images help? Another potential addition to the SeaBird framework is using a denoiser between segmentation and detection heads. We use the MIRNet-v2 [111] as our denoiser and train the BEV segmentation head, denoiser and detection head in an end-to-end manner. Tab. 12 shows that denoising does not increase performance but the inference time. Hence, we do not use any denoiser for SeaBird.

Sensitivity to Segmentation Weight. We next study the impact of segmentation weight on I2M+SeaBird in Tab. 13 as in Sec. 4.2. Tab. 13 shows that $\lambda_{seg} = 5$ works the best for the Mono3D of large objects.

Reproducibility. We ensure reproducibility of our results by repeating our experiments for 3 random seeds. We choose the final epoch as our checkpoint in all our experiments as [43]. Tab. 14 shows the results with these seeds. SeaBird outperforms SeaBird without dice loss in the median and average cases. The biggest improvement shows up on larger objects.

A3.2. nuScenes Results

Extended Val Results. Besides showing improvements upon existing detectors in Tab. 7 on the nuScenes Val split, we compare with more recent SoTA detectors with large backbones in Tab. 16.

Dice vs regression on depth estimation methods. We report HoP +R50 config, which uses depth estimation and compare losses in Tab. 15. Tab. 15 shows that Dice model again outperforms regression loss models.

SeaBird Compatible Approaches. SeaBird conditions the detection outputs on segmented BEV features and so, requires foreground BEV segmentation. So, all approaches which produce latent BEV map in Tabs. 6 and 7 are compatible with SeaBird. However, approaches which do not produce BEV features such as SparseBEV [55] are incompatible with SeaBird.

A3.3. Qualitative Results

KITTI-360. We now show some qualitative results of models trained on KITTI-360 Val split in Fig. 8. We depict the predictions of PBEV+SeaBird in image view on the left, the predictions of PBEV+SeaBird, the baseline MonoDETR [114], predicted and GT boxes in BEV in the middle and BEV semantic segmentation predictions from PBEV+SeaBird on the right. In general, PBEV+SeaBird detects more larger objects (buildings) than GUP Net [63].

nuScenes. We now show some qualitative results of models trained on nuScenes Val split in Fig. 9. As before, we depict the predictions of BEVerse-S+SeaBird in image view from six cameras on the left and BEV semantic segmentation predictions from SeaBird on the right.

KITTI-360 Demo Video. We next put a short demo video of SeaBird model trained on KITTI-360 Val split compared with MonoDETR at `https://www.youtube.com/watch?v=SmuRbMbsnZA`. We run our trained model independently on each frame of KITTI-360. None of the frames from the raw video appear in the training set of KITTI-360 Val split. 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, predicted and GT boxes in BEV in the middle and BEV semantic segmentation predictions from PBEV+SeaBird. We set the frame rate of this demo at 5 fps similar to [43]. The attached demo video demonstrates impressive results on larger objects.

A4. Acknowledgements

This research was partially sponsored by the Bosch Research North America, Bosch Center for AI 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 ARO or the U.S. government.

We thank several members of the Computer Vision community for making this project possible. We deeply appreciate Rakesh Menon, Vidit, Abhishek Sinha, Avrajit Ghosh, Andrew Hou, Shengjie Zhu, Rahul Dey, Saurabh Kumar and Ayushi Raj for several invaluable discussions during this project. Rakesh suggested the MonoDLE [66] baseline for KITTI-360 models because MonoDLE normalizes loss with GT box dimensions. Shengjie, Avrajit, Rakesh, Vidit, and Andrew proofread our manuscript and suggested several changes. Shengjie helped us parse the KITTI-360

Table 11. **KITTI-360 Val results with naive baseline finetuned for large objects.** SeaBird pipelines comfortably outperform this naive baseline on large objects. [Key: **Best**, **Second Best**, †= Retrained]

Method	Venue	$AP_{3D}50$ [%](↑)			$AP_{3D}25$ [%](↑)			BEV Seg IoU [%](↑)		
		AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
GUP Net†[63]	ICCV21	0.54	**45.11**	22.83	0.98	50.52	25.75	–	–	–
GUP Net (Large FT)†[63]	ICCV21	0.56	–	0.28	2.56	–	1.28	–	–	–
I2M+SeaBird	CVPR24	**8.71**	**43.19**	**25.95**	**35.76**	**52.22**	**43.99**	**23.23**	**39.61**	**31.42**
PBEV+SeaBird	CVPR24	**13.22**	42.46	**27.84**	**37.15**	**52.53**	**44.84**	**24.30**	**48.04**	**36.17**

Table 12. **Impact of denoising** BEV segmentation maps with MIRNet-v2 [111] on KITTI-360 Val with I2M+SeaBird. Denoising does not help. [Key: **Best**]

Denoiser	$AP_{3D}50$ [%](↑)			$AP_{3D}25$ [%](↑)			BEV Seg IoU [%](↑)		
	AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
✓	2.73	**43.77**	23.25	14.34	51.23	32.79	21.42	**39.72**	30.57
×	**8.71**	43.19	**25.95**	**35.76**	**52.22**	**43.99**	**23.23**	39.61	**31.42**

Table 13. **Segmentation loss weight λ_{seg} sensitivity** on KITTI-360 Val with I2M+SeaBird. $\lambda_{seg} = 5$ works the best. [Key: **Best**]

λ_{seg}	$AP_{3D}50$ [%](↑)			$AP_{3D}25$ [%](↑)			BEV Seg IoU [%](↑)		
	AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
0	4.86	**45.09**	24.98	26.33	52.31	39.32	0	7.07	3.54
1	7.07	41.71	24.39	32.92	52.9	42.91	23.78	40.58	32.18
3	7.26	43.45	25.36	34.47	**52.54**	43.51	**23.40**	**40.15**	**31.78**
5	**8.71**	43.19	**25.95**	**35.76**	52.22	**43.99**	23.23	39.61	31.42
10	7.69	43.41	25.55	34.22	50.97	42.60	22.15	39.83	30.99

Table 14. **Reproducibility results** on KITTI-360 Val with I2M+SeaBird. SeaBird outperforms SeaBird without dice loss in the median and average cases. [Key: **Best**, **Second Best**]

Dice	Seed	$AP_{3D}50$ [%](↑)			$AP_{3D}25$ [%](↑)			BEV Seg IoU [%](↑)		
		AP_{Lrg}	AP_{Car}	mAP	AP_{Lrg}	AP_{Car}	mAP	Large	Car	M_{For}
×	111	3.81	44.63	24.22	24.96	53.15	39.06	0	5.99	3.00
	444	4.86	45.09	24.98	26.33	52.31	39.32	0	7.07	3.54
	222	5.79	46.71	26.25	24.32	54.06	39.19	0	5.32	2.66
	Avg	4.82	45.58	**25.15**	25.20	53.17	**39.19**	0	6.13	3.06
✓	111	7.87	44.03	25.95	33.55	53.93	43.74	22.64	40.64	31.64
	444	8.71	43.19	25.95	35.76	52.22	43.99	23.23	39.61	31.42
	222	8.71	42.87	25.79	34.71	51.72	43.22	22.74	40.01	31.38
	Avg	8.43	43.36	**25.90**	34.67	52.62	**43.65**	22.87	40.09	31.48

Table 15. **Dice vs regression on methods with depth estimation**. Dice model again outperforms regression loss models, particularly for large objects. [Key: **Best**, **Second Best**]

Resolution	Method	BBone	Venue	Loss	AP_{Lrg} (↑)	AP_{Car} (↑)	AP_{Sml} (↑)	mAP (↑)	NDS (↑)
256×704	HoP+SeaBird	R50	ICCV23	–	**27.4**	**57.2**	**46.4**	**39.9**	**50.9**
			–	$\mathcal{L}_1$	27.0	57.1	46.5	39.7	50.7
			–	$\mathcal{L}_2$	Did Not Converge				
			CVPR24	Dice	**28.2**	**58.6**	**47.8**	**41.1**	**51.5**

dataset, while Andrew helped in the KITTI-360 leaderboard evaluation. We also thank Prof. Yiyi Liao from Zhejiang University for discussions on the KITTI-360 conventions and evaluation protocol. We finally thank anonymous NeurIPS and CVPR reviewers for their exceptional feedback and constructive criticism that shaped this final manuscript.

Table 16. **nuScenes Val Detection results**. SeaBird pipelines outperform the baselines, particularly for large objects. [Key: **Best**, **Second Best**, B= Base, S= Small, T= Tiny, ♙= Released, *= Reimplementation, §= CBGS]

Resolution	Method	BBone	Venue	AP_{Lrg} (↑)	AP_{Car} (↑)	AP_{Sml} (↑)	mAP (↑)	NDS (↑)
256×704	CAPE♙[104]	R50	CVPR23	18.5	53.2	38.1	31.8	44.2
	PETRv2 [58]	R50	ICCV23	–	–	–	34.9	45.6
	SOLOFusion♙§[75]	R50	ICLR23	26.5	**57.3**	**48.5**	**40.6**	49.7
	BEVerse-T♙[116]	Swin-T	ArXiv	18.5	53.4	38.8	32.1	46.6
	BEVerse-T+SeaBird	Swin-T	CVPR24	19.5	54.2	41.1	33.8	48.1
	HoP♙[121]	R50	ICCV23	**27.4**	57.2	46.4	39.9	**50.9**
	HoP+SeaBird	R50	CVPR24	**28.2**	**58.6**	**47.8**	**41.1**	**51.5**
512×1408	3DPPE [89]	R101	ICCV23	–	–	–	39.1	45.8
	STS [101]	R101	AAAI23	–	–	–	43.1	52.5
	P2D [37]	R101	ICCV23	–	–	–	43.3	52.8
	BEVDepth [48]	R101	AAAI23	–	–	–	41.8	53.8
	BEVDet4D [31]	R101	ArXiv	–	–	–	42.1	54.5
	BEVerse-S♙[116]	Swin-S	ArXiv	20.9	56.2	42.2	35.2	49.5
	BEVerse-S+SeaBird	Swin-S	CVPR24	24.6	58.7	45.0	38.2	51.3
	HoP*[121]	R101	ICCV23	**31.4**	**63.7**	**52.5**	**45.2**	**55.0**
	HoP+SeaBird	R101	CVPR24	**32.9**	**65.0**	**53.1**	**46.2**	**54.7**
640×1600	BEVDet [32]	V2-99	ArXiv	29.6	61.7	48.2	42.1	48.2
	PETRv2 [58]	R101	ICCV23	–	–	–	42.1	52.4
	CAPE♙[104]	V2-99	CVPR23	31.2	63.2	51.9	44.7	54.4
	BEVDet4D§[31]	Swin-B	ArXiv	–	–	–	42.6	55.2
	HoP*[121]	V2-99	ICCV23	**36.5**	**69.1**	**56.1**	**49.6**	**58.3**
	HoP+SeaBird	V2-99	CVPR24	**40.3**	**71.7**	**58.8**	**52.7**	**60.2**
900×1600	FCOS3D[93]	R101	ICCVW21	–	–	–	34.4	41.5
	PGD [94]	R101	CoRL21	–	–	–	36.9	42.8
	DETR3D [97]	R101	CoRL21	22.4	60.3	41.1	34.9	43.4
	PETR [57]	R101	ECCV22	–	–	–	37.0	44.2
	BEVFormer [50]	R101	ECCV22	27.7	48.5	34.5	41.5	51.7
	PolarFormer [36]	V2-99	AAAI23	–	–	–	50.0	56.2

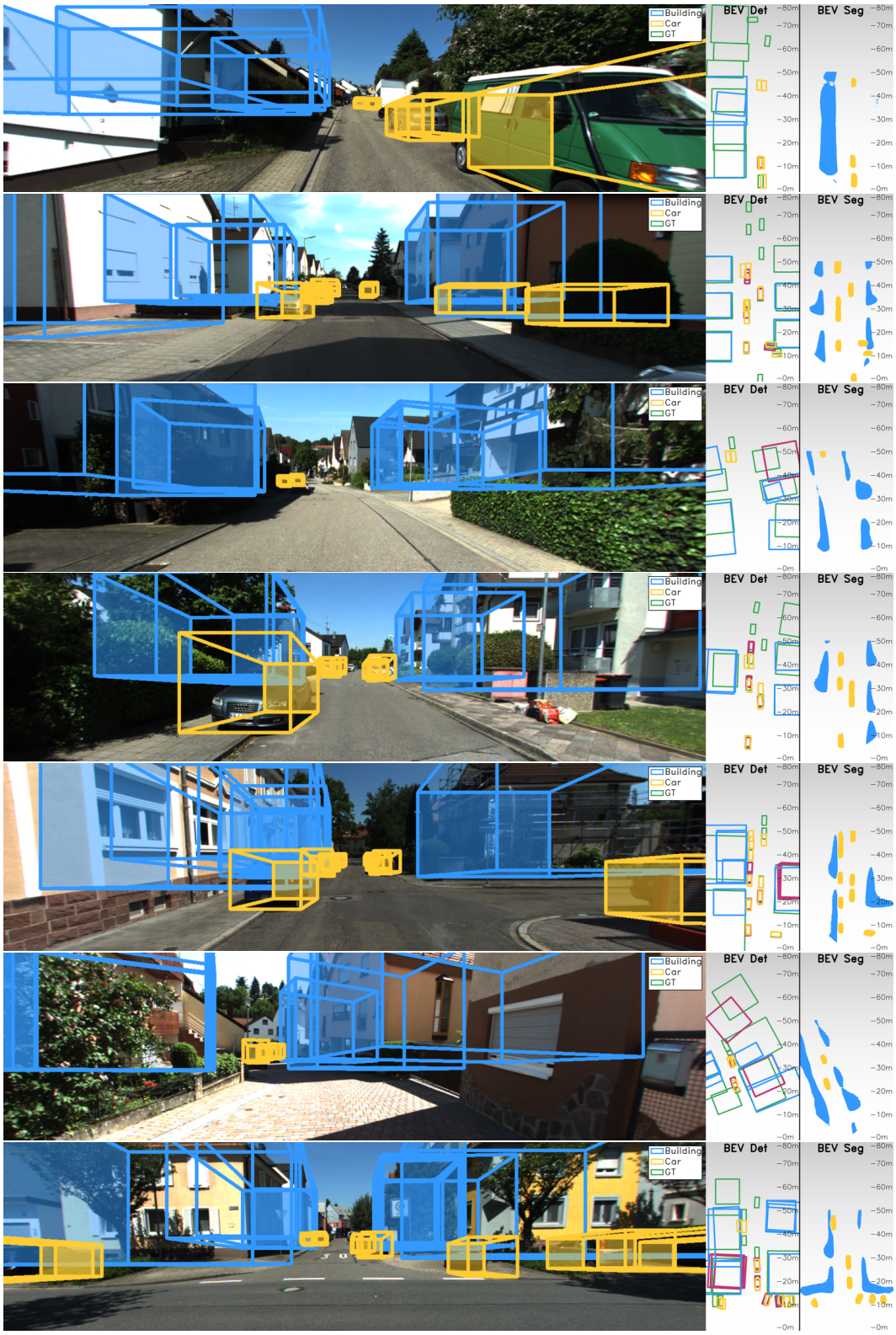

Figure 8. **KITTI-360 Qualitative Results**. PBEV+SeaBird detects more large objects (buildings) than MonoDETR [114]. We depict the predictions of PBEV+SeaBird in the image view on the left, the predictions of PBEV+SeaBird, the baseline MonoDETR [114], and ground truth in BEV in the middle, and BEV semantic segmentation predictions from PBEV+SeaBird on the right. [Key: Buildings and Cars of PBEV+SeaBird; all classes of MonoDETR [114], and Ground Truth in BEV].

Figure 9. **nuScenes Qualitative Results**. The first row shows the front_left, front, and front_right cameras, while the second row shows the back_left, back, and back_right cameras. [Key: Cars, Vehicles, Pedestrian, Cones and Barrier of BEVerse-S+SeaBird at 200×200 resolution in BEV].