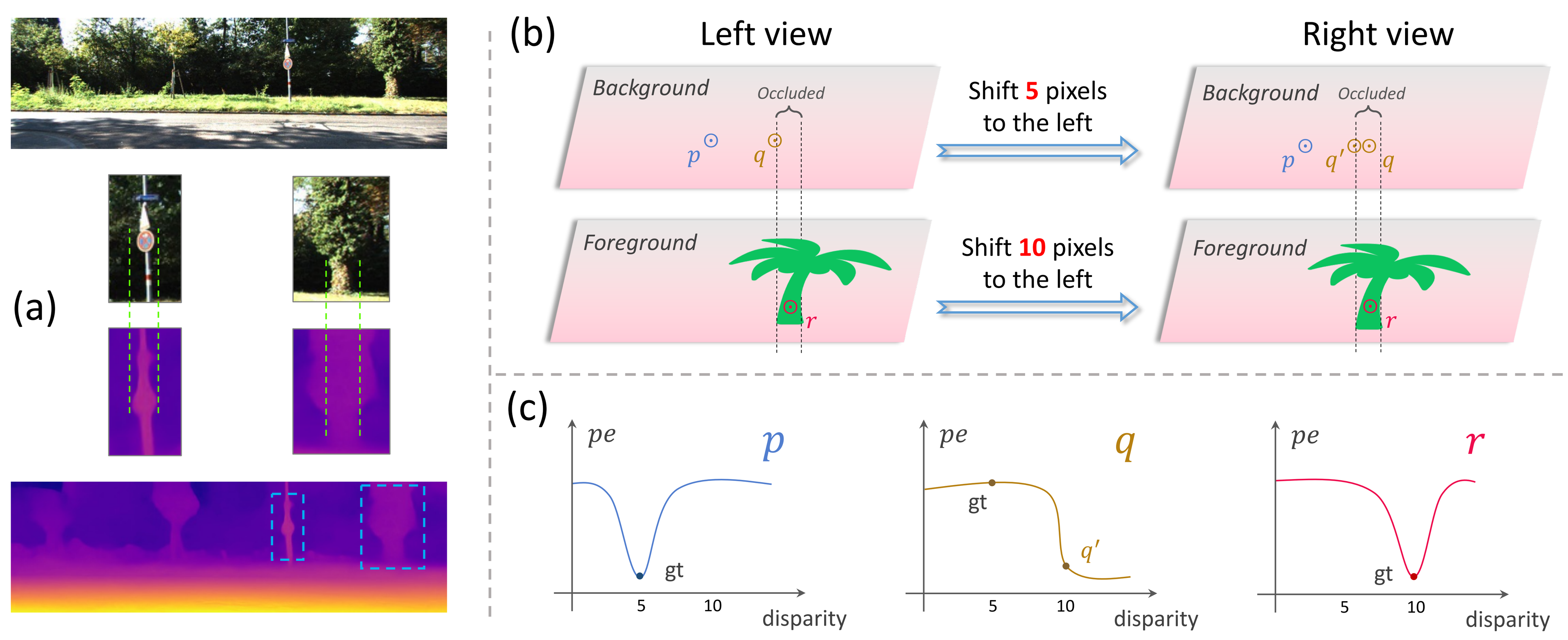
Self-Supervised Monocular Depth Estimation: Solving the Edge-Fattening Problem

Xingyu Chen[1] Ruonan Zhang[1] Ji Jiang[1] Yan Wang[1] Ge Li[1] Thomas H. Li ✉[1,2,3]

[1]SECE, Peking University [2]Advanced Institute of Information Technology, Peking University

[3]Information Technology R&D Innovation Center of Peking University

What does the Edge-Fattening look like? (a) And Why? (b & c)

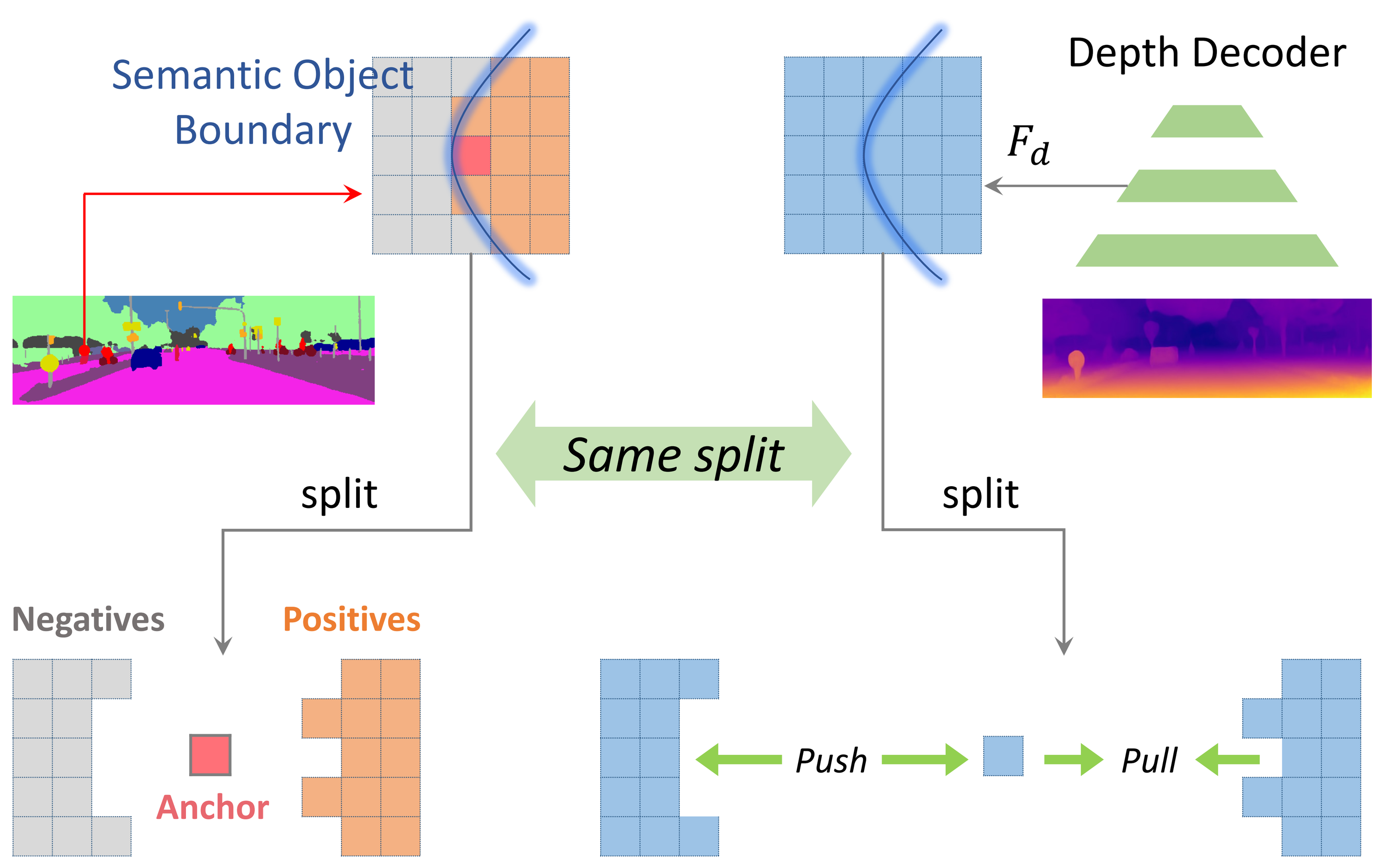


(a) Example of the edge-fattening issue. The depth predictions of foreground objects (*e.g.* the tree-trunk and poles) are 'fatter' than the objects themselves.

(b) In the left view, pixel p and q are located in the background with a disparity of 5 pixels, and q will be occluded if any further to the right. r is on the tree with a disparity of 10 pixels.

(c) p and r are OK - their *gt* disparity is the global minimum of the photometric error. q suffers from edge-fattening issue. Since q is occluded by the tree in the right view, the photometric error of its *gt* disparity 5 is large. The photometric loss function therefore struggles to find another location that has a small loss, *i.e.*, shifting another 5 pixels to reach the nearest background pixel q'. However, q' is not the true correspondence of q. As a result, disparity of the background q equals to that of the foreground r, leading to the edge-fattening issue.

Basic Idea of the Patch-based Triplet Loss



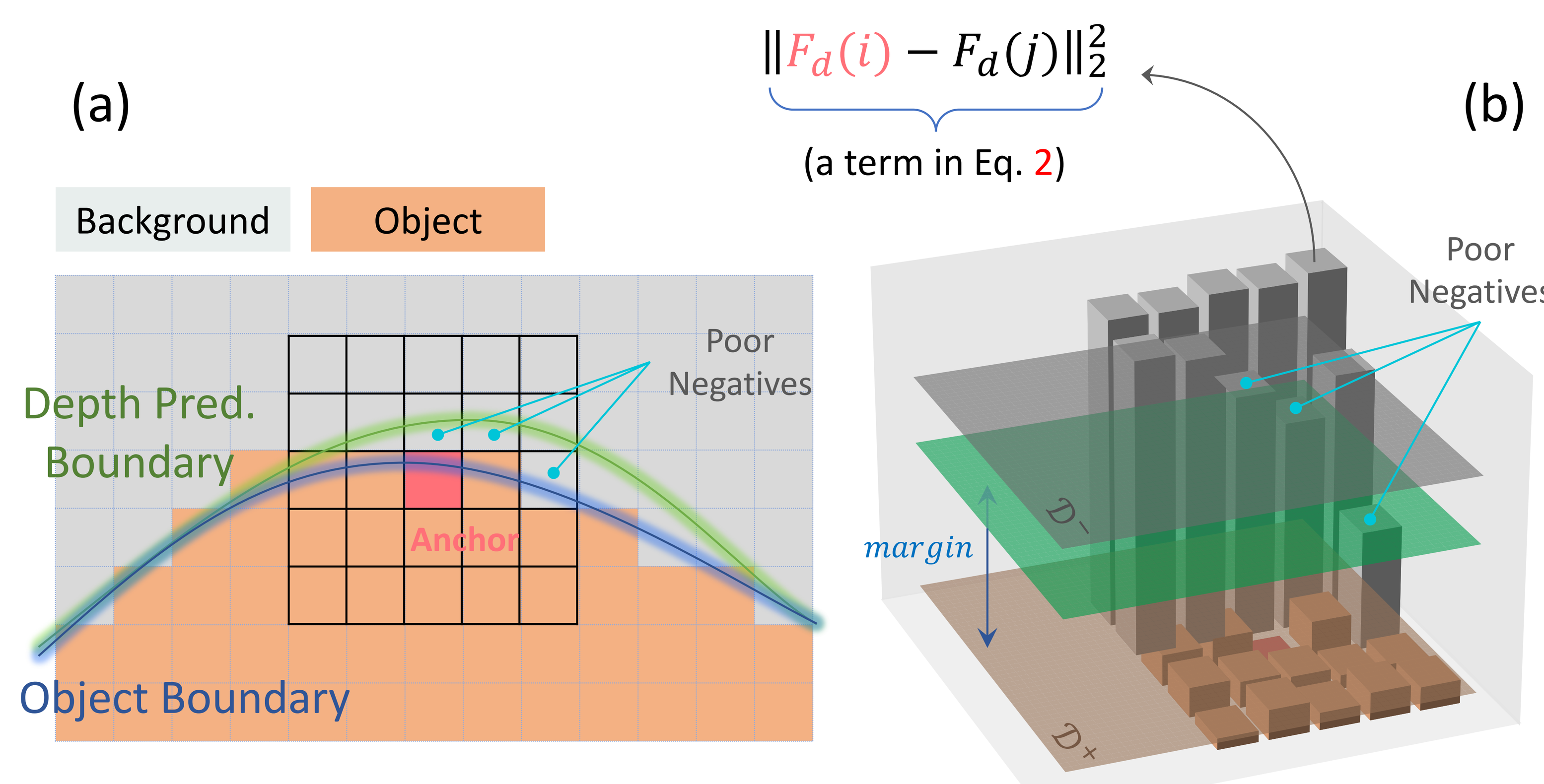
For each pixel i in the semantic boundary region, we group the local patch of its corresponding depth feature (F_d) into a triplet according to the semantic patch ($\mathcal{P}_i$). Next, the triplet loss $\mathcal{L}_{tri}$ minimizes anchor-positive distance ($\mathcal{D}^+$) and maximizes the anchor-negative distance ($\mathcal{D}^-$) until there is a margin m, *i.e.*

$$\mathcal{D}^+(i) = \frac{1}{|\mathcal{P}_i^+|} \sum_{j \in \mathcal{P}_i^+} \| F_d(i) - F_d(j) \|_2^2, \quad (1)$$

$$\mathcal{D}^-(i) = \frac{1}{|\mathcal{P}_i^-|} \sum_{j \in \mathcal{P}_i^-} \| F_d(i) - F_d(j) \|_2^2, \quad (2)$$

$$\mathcal{L}_{tri} = [\mathcal{D}^+(i) - \mathcal{D}^-(i) + m]_+ \quad (3)$$

Contribution 1: Help the Poorest Negatives



(a) The *poor negatives* are the fatter ones. **(b)** 3-D Bars: The Euclidean distance between depth features of every pixels in the patch and the anchor. Yellow plane - $\mathcal{D}^+$. Grey plane - $\mathcal{D}^-$. Green plane - decision boundary whether this triplet participates in training (the hinge function in Eq. 3). The Green plane lies beneath the Grey, thus, no learning happens. Disappointingly, the 'fatter' *poor negatives* get no optimization. This is because of the *mean* operator - the low-proportion poor negatives' contributions are weakened by the large-proportion good negatives. Therefore, we change $\mathcal{D}^-$ into:

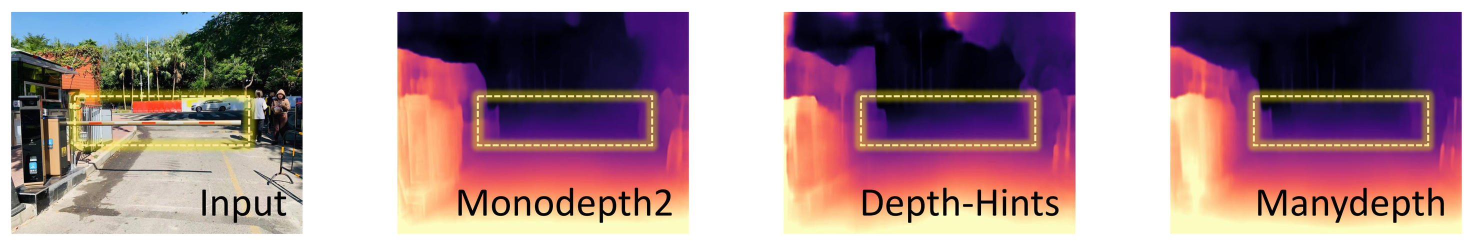
$$\mathcal{D}^{-\prime}(i) = \min_{j \in \mathcal{P}_i^-} \| F_d(i) - F_d(j) \|_2^2 . \quad (4)$$

How Powerful are these two small Redesigns?

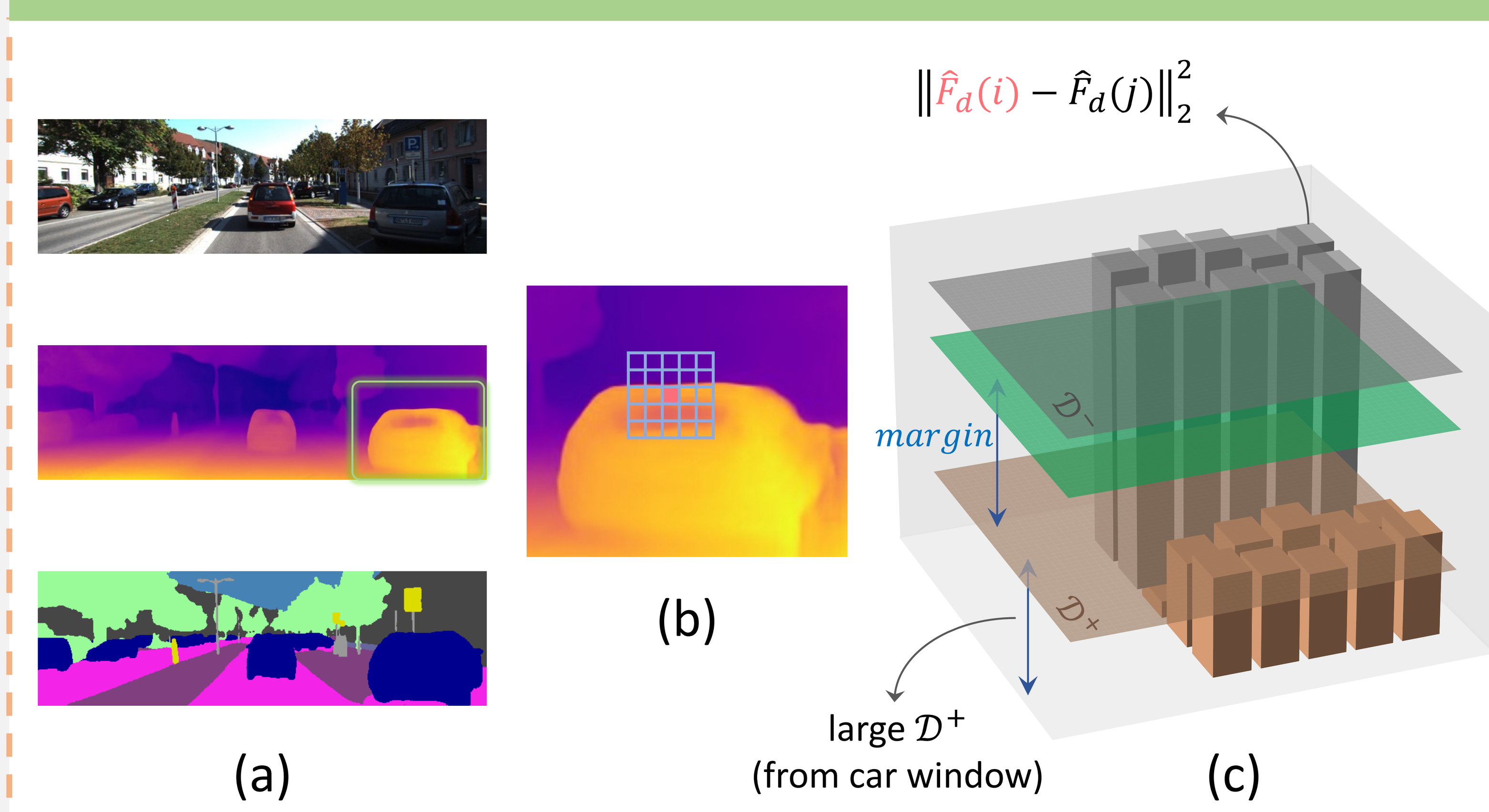
We integrate our method into almost all the state-of-the-arts these years. Models augmented with our redesigned triplet loss achieve better results on exactly all metrics, while exactly no extra inference computation is introduced at all.

Method	Pub.	PP	W × H	Data	Extra time	Abs Rel	Sq Rel	RMSE	RMSE log	δ_1	δ_2	δ_3
Monodepth2 M [10]	ICCV 2019	✗	640 × 192	M	–	0.115	0.903	4.863	0.193	0.877	0.959	0.981
+ Ours	–	✗	640 × 192	M	+ 0ms	**0.108**	**0.744**	**4.537**	**0.184**	**0.883**	**0.963**	**0.983**
Zhou *et al.* [51]	CVPR 2017	✗	640 × 192	M	–	0.183	1.595	6.709	0.270	0.734	0.902	0.959
+ Ours	–	✗	640 × 192	M	+ 0ms	**0.148**	**1.098**	**5.150**	**0.212**	**0.819**	**0.949**	**0.980**
Monodepth2 S [10]	ICCV 2019	✗	640 × 192	S	–	0.109	0.873	4.960	0.209	0.864	0.948	0.975
+ Ours	–	✗	640 × 192	S	+ 0ms	**0.107**	**0.826**	**4.822**	**0.201**	**0.866**	**0.953**	**0.978**
FSRE-Depth only SGT [24]	ICCV 2021	✗	640 × 192	M	–	0.113	0.836	4.711	0.187	0.878	0.960	0.982
+ Ours	–	✗	640 × 192	M	+ 0ms	**0.108**	**0.746**	**4.507**	**0.182**	**0.884**	**0.964**	**0.983**
Monodepth2 MS [10]	ICCV 2019	✗	640 × 192	MS	–	0.106	0.818	4.750	0.196	0.874	0.957	0.979
+ Ours	–	✗	640 × 192	MS	+ 0ms	**0.105**	**0.753**	**4.563**	**0.182**	**0.887**	**0.963**	**0.983**
Depth-Hints [45]	ICCV 2019	✗	640 × 192	S	–	0.109	0.845	4.800	0.196	0.870	0.956	**0.980**
+ Ours	–	✗	640 × 192	S	+ 0ms	**0.106**	**0.843**	**4.774**	**0.194**	**0.875**	**0.957**	**0.980**
HR-Depth [30]	AAAI 2020	✗	640 × 192	M	–	0.109	0.792	4.632	0.185	0.884	0.962	0.983
+ Ours	–	✗	640 × 192	M	+ 0ms	**0.107**	**0.760**	**4.522**	**0.182**	**0.886**	**0.964**	**0.984**
HR-Depth MS [30]	AAAI 2020	✗	640 × 192	MS	–	0.107	0.785	4.612	0.185	0.887	0.962	0.982
+ Ours	–	✗	640 × 192	MS	+ 0ms	**0.105**	**0.751**	**4.512**	**0.181**	**0.890**	**0.963**	**0.983**
ManyDepth [46]	CVPR 2021	✗	640 × 192	M	–	0.098	0.770	4.459	0.176	0.900	0.965	0.983
+ Ours	–	✗	640 × 192	M	+ 0ms	**0.093**	**0.665**	**4.272**	**0.171**	**0.907**	**0.967**	**0.984**
CADepth [47]	3DV 2021	✗	640 × 192	M	–	0.110	0.812	4.686	0.187	0.882	0.962	0.983
+ Ours	–	✗	640 × 192	M	+ 0ms	**0.105**	**0.745**	**4.530**	**0.181**	**0.888**	**0.965**	**0.984**

Future work: Can we trust in current depth estimators?



Contribution 2: Help the Poor Positives



Example of how the error of poor positives is sheltered by good negatives. **(b)** Image patch in (a), where the depth prediction of the car window (positives) is wrong - it has to be the same as the car body. **(c)** The average of $\mathcal{D}^-$ is so large that the grey plane lies above the green plane. That is, the triplet loss stops working. However, all poor positive car window pixels (the large $\mathcal{D}^+$) get no optimization. Therefore, we easily split $\mathcal{D}^+$ and $\mathcal{D}^-$, providing $\mathcal{D}^+$ with more direct optimizations:

$$\mathcal{L}_{tri}' = \mathcal{D}^+(i) + [m' - \mathcal{D}^-(i)]_+ \quad (5)$$