Physics-based Differentiable Depth Sensor Simulation (Supplementary Material)

Benjamin Planche Rajat Vikram Singh[‡]
Siemens Technology
benjamin.planche@siemens.com, rajats@alumni.cmu.edu

In this supplementary material, we provide further implementation details for reproducibility, as well as additional qualitative and quantitative results.

A. Implementation

A.1. Practical Details

Our framework is implemented using PyTorch [21], for seamless integration with optimization and recognition methods. Inference and training procedures are performed on a GPU-enabled backend machine (with two NVIDIA Tesla V100-SXM2 cards). Differentiable ray-tracing and 3D data processing are performed by the *Redner* tool [18] kindly provided by Li *et al*. [19]. Optional learning-based post-processing is performed by two convolutional layers, resp. with 32 filters of size 5×5 and 32 filters of size 1×1. The first layer takes as input a 3-channel image composed of the simulated depth map, as well as its noise-free depth map and shadow map (all differentiably rendered by *DDS*).

When optimizing *DDS* (in a supervised or unsupervised manner), we use *Adam* [12] with a learning rate of 0.001 and no weight decay. For supervised optimization, we opt for a combination of *Huber* loss [10] and gradient loss [11] (the latter comparing the pseudo-gradient maps obtained from the depth scans by applying *Sobel* filtering). For unsupervised optimization, we adopt the training scheme and losses from *PixelDA* [2], *i.e*., training *DDS* against a discriminator network and in collaboration with the task-specific recognition CNN.

A.2. Computational Optimization

On top of the solutions mentioned in the main paper w.r.t. reducing the computational footprint of *DDS*, we further optimize our pipeline by parallelizing the proposed block-matching algorithm. Since the correspondence search performed by our method is purely horizontal (*c.f*. horizontal epipolar lines), compared images I_c and I_o can be split into m pairs $\{I_{c,j}, I_{o,j}\}_{j=1}^{m}$ with:

$$I_c = \begin{bmatrix} I_{c,0} \\ I_{c,1} \\ \dots \\ I_{c,m} \end{bmatrix} \quad ; \quad I_o = \begin{bmatrix} I_{o,0} \\ I_{o,1} \\ \dots \\ I_{o,m} \end{bmatrix}, \tag{1}$$

i.e., horizontally splitting the images into m pairs. The stereo block-matching procedure can be performed on each pair independently, enabling computational parallelization (*e.g*., fixing m as the number of available GPUs). Note that to account for block size $w \times w$, each horizontal splits $I_{c,j+1}$ and $I_{o,j+1}$ overlaps the previous ones (resp. $I_{c,j}$ and $I_{o,j}$) by w pixels (for notation clarity, Equation 1 does not account for this overlapping).

A.3. Simulation Parameters

The results presented in the paper are obtained by providing the following simulation parameters to *DDS* (both as fixed parameters to the off-the-shelf instances and as initial values to the optimized versions):

Microsoft Kinect V1 Simulation:

- Image ratio $\frac{H}{W} = \frac{4}{3}$;
- Focal length $f_\lambda = 572.41$px;
- Baseline distance $b = 75$mm;
- Sensor range $[z_{min}, z_{max}] = [400\text{mm}, 4000\text{mm}]$;
- Block size $w = 9$px;
- Emitted light intensity factor $\eta_c = 1.5 \times 10^6$;
- Shadow bias $\xi = 5$mm;
- Softargmax temperature parameter $\beta = 15.0$;
- Subpixel refinement level $n_{sub} = 2$;

Matterport Pro2 Simulation:

- Image ratio $\frac{H}{W} = \frac{5}{4}$;
- Focal length $f_\lambda = 1075.43$px;
- Baseline distance $b = 75$mm;

[‡]Now at NVIDIA.

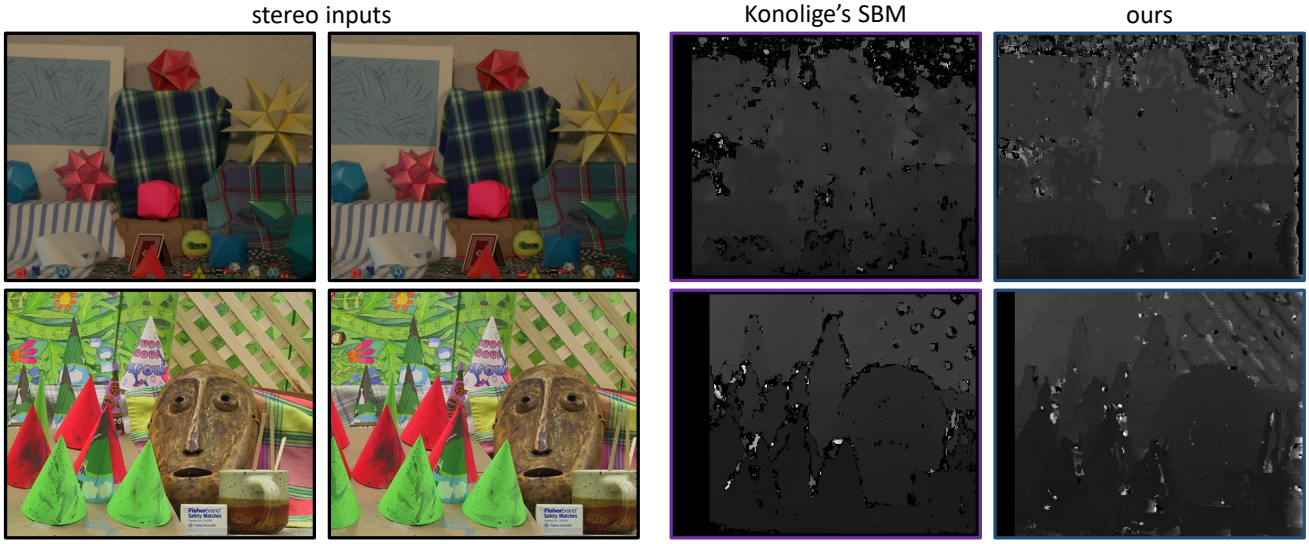

Figure S1: **Comparison of block-matching solutions applied to depth regression from stereo color images**. Our soft block-matching algorithm is compared to Konolige's one [13, 14] often used in depth simulation.

- Sensor range $[z_{min}, z_{max}] = [400\text{mm}, 8000\text{mm}]$;
- Block size $w = 11\text{px}$;
- Emitted light intensity factor $\eta_c = 1.5 \times 10^{12}$;
- Shadow bias $\xi = 1\text{mm}$;
- Softargmax temperature parameter $\beta = 25.0$;
- Subpixel refinement level $n_{sub} = 4$;

Note that device-related parameters come from the sensors' manufacturers or previous *Kinect* studies [16, 15]. Other parameters have been manually set through empirical evaluation. For the structured-light pattern, we use the *Kinect* pattern image reverse-engineered by Reichinger [23].

B. Additional Results

B.1. Application to RGB Stereo Matching

Figure S1 provides a glimpse at how the proposed differentiable block-matching algorithm can perform in a standalone fashion and be applied to problems beyond the stereo analysis of structured-light patterns. In this figure, our algorithm is applied to the depth measurement of complex stereo color images (without its sub-pixel refinement step, since it relies on ray-tracing). We compare it to the standard stereo block-matching algorithm proposed by Konolige [13, 14] and used by previous depth sensor simulations [5, 22]. Stereo color images come from the *Middlebury Stereo* dataset [25, 24, 8]. We can appreciate the relative performance of the proposed method, in spite of its excessive quantization (hence the additional sub-pixel refinement proposed in the paper and highlighted in Figure S2) and approximations for higher-frequency content. We can also observe artifacts for pixels with ambiguous correspondences due to the softargmax-based reduction performed by our method (whereas Konolige's algorithm yields null values when the correspondences are too ambiguous).

B.2. Realism Study

Qualitative Comparison. Additional Figure S2 depicts the control over the discrepancy/depth granularity provided by the hyper-parameter N_{sub} (level of subpixel refinement). Incidentally, this figure also shows the impact of non-modelled scene properties on the realism of the simulated scans. The 3D models of the target scenes provided by the dataset authors [1], used to render these scans, do not contain texture/material information and have various geometrical defects; hence some discrepancies between the real and synthetic representations (*e.g.*, first row of Figure S2: the real scan is missing data due to the high reflectivity of some ceiling elements; an information non-modelled in the provided 3D model). As our pipeline is differentiable not only w.r.t. the sensor's parameters but also the scene's ones, it could be in theory used to optimize/learn such incorrect or missing scene properties. In practice, this optimization would require careful framing and constraints (worth its own separate study) not to computationally explode , especially for complex, real-life scenes.

Figure S3 contains randomly picked synthetic and real images based on the *2D-3D-Semantic* dataset [1]. We can observe how the *DepthSynth* method proposed by Planche *et al*. [22] tends to over-induce noise, sometimes completely failing at inferring the depth through stereo block-matching. It may be due to the choice of block-matching algorithm [13, 14], as the authors rely on a popular but rather antiquated method, certainly not as robust

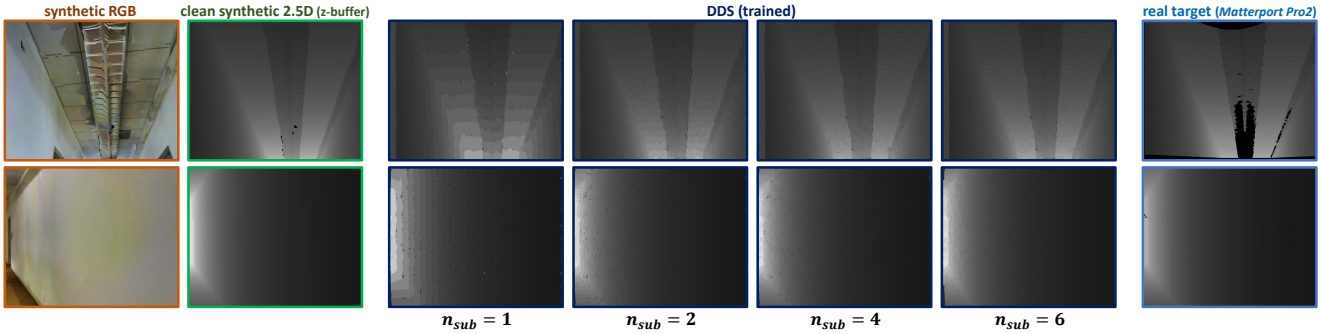

Figure S2: **Impact of proposed differentiable sub-pixel refinement on depth quantization**, depicted over the *2D-3D-Semantic* dataset [1].

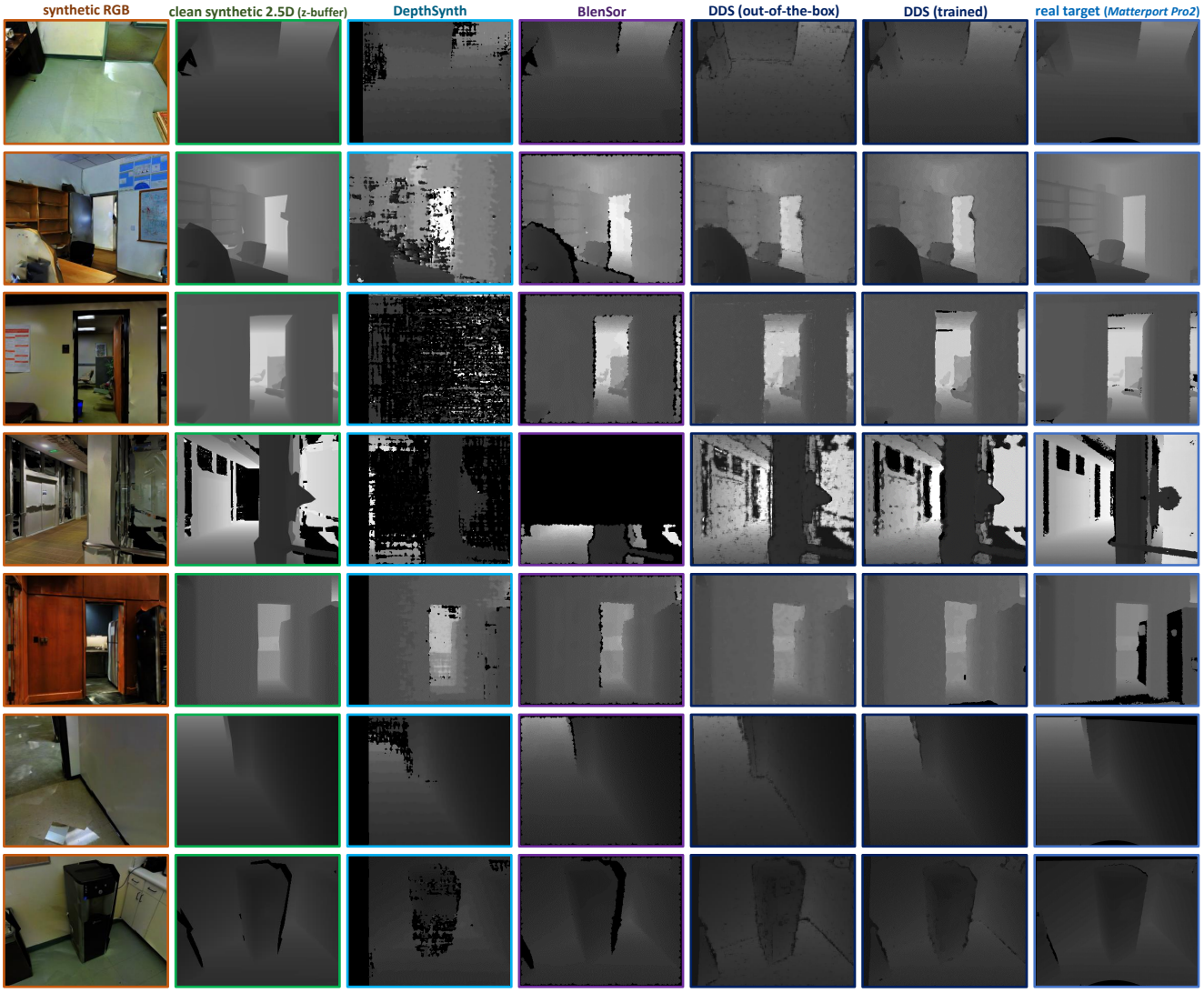

Figure S3: **Qualitative comparison of simulated scans.** Synthetic depth images rendered from reconstructed 3D indoor scenes of the *2D-3D-Semantic* dataset [1], compared to real scans from the *Matterport Pro2* sensor. Note that the *Pro2* device relies on 3 stacked depth sensors, hence the high accuracy and reduced shadow noise.

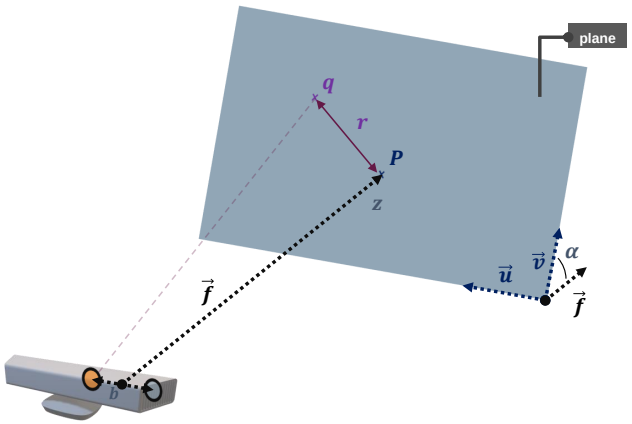

Figure S4: **Experimental setup for quantitative noise study of a depth sensor**, as proposed by Landau *et al*. [16].

as the (unspecified) algorithm run by the target *Matterport Pro2* device. Our own block-matching solution is not much more robust (*c.f*. Figure S1) and also tends to over-induce noise in the resulting depth images. Until a more robust differentiable solution is proposed, *DDS* can, however, rely on its post-processing capability to compensate for the block mismatching and to generate images that are closer to the target ones, as shown in Figure S3 (penultimate column). As for the *BlenSor* simulation [5], its image quality is qualitatively good, though it cannot be configured, *e.g*., to reduce the shadow noise (the tool proposes a short list of pre-configured sensors that it can simulate). Moreover, for reasons unknown, the open-source version provided by the authors fails to properly render a large number of images from the *2D-3D-S* scenes, resulting in scans missing a large portion of the content (*c.f*. fourth row in Figure S3). This probably explains the low performance of the CNN for semantic segmentation trained over *BlenSor* data. Finally, unlike static simulations, the proposed solution can learn to tune down its inherent noise to model more precise sensors such as the multi-shot *Matterport* device (composed of 3 sensors).

Quantitative Comparison. Figure S4 illustrates the experimental setup described in Subsection 4.1 of the paper w.r.t. noise study. We consider a flat surface placed at distance z from the sensor, with a tilt angle α w.r.t. the focal plane (with $\vec{f}$ its normal).

Note that for this experiment, we use the experimental data collected and kindly provided by Landau *et al*. [16].

B.3. Applications to Deep Learning

Table S1 extends the results presented in the paper (Table 1), considering the cases when annotations are provided for the subset of real training images. In such a scenario, the segmentation method can be supervisedly trained either purely on the (rather limited) real data, or on a larger, more varied mix of real and synthetic data. The additional last three rows in Table S1 present the test results considering the latter option. We can observe how the CNN instances trained on such larger datasets—and more specifically the CNN instance trained on a mix of real and *DDS* data—are more accurate than the instance trained purely on real data.

Table S1: **Comparative study w.r.t. training usage (extending study in Table 1)**, measuring the accuracy of a CNN [6, 26, 27] performing semantic segmentation on real 2.5D scans from the indoor *2D-3D-S* dataset [1], as a function of the method used to render its training data and as a function of real *annotated* data availability (↑ = the higher the value, the better).

Train. Data Source	Mean Intersection-Over-Union (mIoU)↑								Pixel Acc.↑
	bookc.	*ceili.*	*chair*	*clutter*	*door*	*floor*	*table*	*wall*	
clean	.003	.018	.002	.087	.012	.052	.091	.351	35.3%
BlenSor [5]	.110	.534	.119	.167	.148	.561	.082	.412	51.6%
DepthS. [22]	.184	.691	.185	.221	.243	.722	.235	.561	65.3%
DDS	.218	.705	.201	.225	.240	.742	.259	.583	62.9%
DDS (train.)	**.243**	.711	**.264**	.255	.269	.794	.271	.602	69.8%
real	.135	.770	.214	.277	.302	.803	.275	**.661**	73.5%
BlenSor [5] + real	.143	.769	.213	.275	.306	**.817**	.271	.636	73.6%
DepthS. [22] + real	.222	.767	.234	.297	.325	.812	.273	.659	75.8%
DDS + real	**.279**	**.775**	**.245**	**.299**	**.356**	.815	**.280**	.659	**76.7%**

Similarly, Table S2 extends the results presented in the paper (Table 2) w.r.t. training of a CNN for instance classification and pose estimation over the *Cropped LineMOD* dataset [7, 2, 28]. Besides specifying the number of trainable parameters $|\Phi_D|$ that compose discriminator networks (for adversarial domain adaptation methods), we highlight the impact of adding pseudo-realistic clutter to the virtual scenes before rendering images, *i.e*., adding a flat surface as ground below the target object, and randomly placing additional 3D objects around it. Intuitive, the benefit of surrounding the target 3D objects with clutter (for single-object image capture) to the realism of the resulting synthetic images has already been highlighted by previous studies on RGB images [3, 9].

Our results presented in Table S2 extend these conclusions to the 2.5D domain, with a sharp accuracy increase of the resulting recognition models when adding pseudo-realistic clutter to the virtual scenes. This also highlights the importance, in visual simulation, of not only modeling realistic sensor properties but also of properly setting up the virtual scenes (*c.f*. discussion in previous Subsection B.2).

C. Acknowledgments

We would like to deeply thank Tzu-Mao Li for the help provided w.r.t. applying his *Redner* rendering tool [18, 19] to our needs. Finally, credits go to Pierre Yves P. [20] for the 3D *Microsoft Kinect* model used to illustrate some of the figures in our paper.

Table S2: **Comparative and ablative study (extending study in Table 2)**, measuring the impact of unsupervised domain adaptation, sensor simulation (Sim), and domain randomization (DR, *i.e.*, using randomized 2.5D transforms to the rendered images *c.f.* [29, 28] or adding random 3D clutter to the virtual scenes before rendering) on the training of a CNN [4] for depth-based instance classification and pose estimation on the *Cropped LineMOD* dataset [7, 2, 28].

		3D Clutter in Scene	Augmentations		Sim/DA Req.			Class. Accur.↑	Rot. Error↓
			offline	online	$\boldsymbol{X}^r_{trn}$	$\lvert\boldsymbol{\Phi}\rvert$	$\lvert\boldsymbol{\Phi}_D\rvert$		
	Basic							21.3%	91.8°
				DR				39.6%	73.3°
		✓						46.8%	67.0°
		✓		DR				70.7%	53.1°
Dom. Adap.	*PixelDA* [2]			GAN	✓	1.96M	693k	65.8%	56.5°
		✓		GAN	✓	1.96M	693k	85.7%	40.5°
	DRIT++ [17]		GAN		✓	12.3M	33.1M	36.2%	91.9°
			GAN	DR	✓	12.3M	33.1M	62.5%	89.1°
		✓	GAN		✓	12.3M	33.1M	68.0%	60.8°
		✓	GAN	DR	✓	12.3M	33.1M	87.7%	39.8°
	DeceptionNet [28]			DR		1.54M		37.3%	59.8°
		✓		DR		1.54M		80.2%	54.1°
Sensor Simulation	*DepthSynth* [22]		Sim					17.1%	87.5°
			Sim	DR				45.6%	65.4°
		✓	Sim					71.5%	52.1°
		✓	Sim	DR				76.6%	45.4°
	BlenSor [5]		Sim					14.9%	90.1°
			Sim	DR				45.6%	65.3°
		✓	Sim					67.5%	63.4°
		✓	Sim	DR				82.6%	41.4°
	DDS (untrained)		Sim					15.6%	91.6°
			Sim	DR				50.0%	68.9°
		✓	Sim					69.7%	67.6°
		✓	Sim	DR				89.6%	39.7°
Combined	*DDS*		Sim		✓	4	693k	21.3%	80.9°
			Sim	DR	✓	4	693k	51.6%	63.3°
			Sim+conv		✓	2,535	693k	22.6%	78.7°
			Sim+conv	DR	✓	2,535	693k	54.3%	60.4°
		✓	Sim		✓	4	693k	81.2%	49.1°
		✓	Sim	DR	✓	4	693k	90.5%	39.4°
		✓	Sim+conv		✓	2,535	693k	85.5%	45.4°
		✓	Sim+conv	DR	✓	2,535	693k	93.0%	31.3°
	DDS + $(X,Y)^r_{trn}$	✓	Sim+conv	DR	✓	2,535	693k	**97.8%**	**25.1°**
	$(X,Y)^r_{trn}$	✓			✓			95.4%	35.0°

References

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 2, 3, 4

[2] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017. 1, 4, 5

[3] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019. 4

[4] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015. 5

[5] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: blender sensor simulation toolbox. In *Advances in Visual Computing*, pages 199–208. Springer, 2011. 2, 4, 5

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 4

[7] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*. Springer, 2012. 4, 5

[8] Heiko Hirschmuller and Daniel Scharstein. Evaluation of cost functions for stereo matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 2

[9] Tomáš Hodaň, Vibhav Vineet, Ran Gal, Emanuel Shalev, Jon Hanzelka, Treb Connell, Pedro Urbina, Sudipta N Sinha, and Brian Guenter. Photorealistic image synthesis for object instance detection. In *2019 IEEE International Conference on Image Processing (ICIP)*, pages 66–70. IEEE, 2019. 4

[10] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992. 1

[11] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *Proceedings of the European conference on computer vision (ECCV)*, pages 53–69, 2018. 1

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[13] Kurt Konolige. Small vision systems: Hardware and implementation. In *Robotics Research*, pages 203–212. Springer, 1998. 2

[14] Kurt Konolige. Projected texture stereo. In *2010 IEEE International Conference on Robotics and Automation*, pages 148–155. IEEE, 2010. 2

[15] Michael J Landau. *Optimal 6D Object Pose Estimation with Commodity Depth Sensors*. PhD thesis, University of Virginia, 2016. http://search.lib.virginia.edu/catalog/hq37vn57m. Accessed: 2020-10-20. 2

[16] Michael J Landau, Benjamin Y Choo, and Peter A Beling. Simulating kinect infrared and depth images. *IEEE transactions on cybernetics*, 46(12):3018–3031, 2015. 2, 4

[17] Hsin-Ying Lee, Hung-Yu Tseng, Qi Mao, Jia-Bin Huang, Yu-Ding Lu, Maneesh Singh, and Ming-Hsuan Yang. Drit++: Diverse image-to-image translation via disentangled representations. *International Journal of Computer Vision*, 128(10):2402–2417, 2020. 5

[18] Tzu-Mao Li. Github - redner: Differentiable rendering without approximation. `https://github.com/BachiLi/redner`, 2019. Accessed: 2021-03-16. 1, 4

[19] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018. 1, 4

[20] Pierre Yves P. Kinect sensor - 3d warehouse, 2014. `https://3dwarehouse.sketchup.com/model/32ab2192d875d85e58aeac7d536d442b/Kinect-sensor`. Accessed: 2021-03-17. 4

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 1

[22] Benjamin Planche, Ziyan Wu, Kai Ma, Shanhui Sun, Stefan Kluckner, Terrence Chen, Andreas Hutter, Sergey Zakharov, Harald Kosch, and Jan Ernst. Depthsynth: Real-time realistic synthetic data generation from cad models for 2.5 d recognition. In *3DV*. IEEE, 2017. 2, 4, 5

[23] A. Reichinger. Kinect pattern uncovered. `http://azttm.wordpress.com/2011/04/03`, 2011. Accessed: 2020-03-16. 2

[24] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 2

[25] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1):7–42, 2002. 2

[26] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. IEEE, 2018. 4

[27] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019. 4

[28] Sergey Zakharov, Wadim Kehl, and Slobodan Ilic. Deceptionnet: Network-driven domain randomization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 532–541, 2019. 4, 5

[29] Sergey Zakharov, Benjamin Planche, Ziyan Wu, Andreas Hutter, Harald Kosch, and Slobodan Ilic. Keep it unreal: Bridging the realism gap for 2.5 d recognition with geometry priors only. pages 1–11, 2018. 5