Physics-based Differentiable Depth Sensor Simulation

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

Abstract

Gradient-based algorithms are crucial to modern computer-vision and graphics applications, enabling learning-based optimization and inverse problems. For example, photorealistic differentiable rendering pipelines for color images have been proven highly valuable to applications aiming to map 2D and 3D domains. However, to the best of our knowledge, no effort has been made so far towards extending these gradient-based methods to the generation of depth (2.5D) images, as simulating structured-light depth sensors implies solving complex light transport and stereo-matching problems. In this paper, we introduce a novel end-to-end differentiable simulation pipeline for the generation of realistic 2.5D scans, built on physics-based 3D rendering and custom block-matching algorithms. Each module can be differentiated w.r.t. sensor and scene parameters; e.g., to automatically tune the simulation for new devices over some provided scans or to leverage the pipeline as a 3D-to-2.5D transformer within larger computer-vision applications. Applied to the training of deep-learning methods for various depth-based recognition tasks (classification, pose estimation, semantic segmentation), our simulation greatly improves the performance of the resulting models on real scans, thereby demonstrating the fidelity and value of its synthetic depth data compared to previous static simulations and learning-based domain adaptation schemes.

1. Introduction

Progress in computer vision has been dominated by deep neural networks trained over large amount of data, usually labeled. The deployment of these solutions into real-world applications is, however, often hindered by the cost (time, manpower, access, etc.) of capturing and annotating exhaustive training datasets of target objects or scenes. To partially or completely bypass this hard data requirement, an increasing number of solutions are relying on synthetic images rendered from 3D databases for their training [15, 59, 39, 51, 69, 50], leveraging advances in computer graphics [58, 49]. Indeed, physics-based rendering methods are slowly but surely closing the visual gap between real and synthetic color image distributions, simulating complex optical phenomena (e.g., realistic light transport, lens aberrations, Bayer demosaicing, etc.). While these tools still require domain knowledge to be properly parameterized for each new use-case (w.r.t. scene content, camera properties, etc.), their positive impact on the training of color-based visual recognition algorithms has been well documented already [9, 23].

The same cannot be said about depth-based applications. Unlike color camera that captures light intensity, structured-light depth sensors rely on stereo-vision mechanisms to measure the per-pixel distance between their focal plane and elements in the scene. They are useful for geometry-sensitive applications (e.g., robotics), but little effort has been made towards closing the realism gap w.r.t. synthetic depth (2.5D) scans or understanding their impact on the training of depth-based recognition methods. Some simulation pipelines [19, 35, 51] and domain adaptation schemes [63, 16, 62, 5, 71, 69] have been proposed; but the former methods require extensive domain knowledge [51, 71] to be set up whereas some of the latter need relevant real images for their training [63, 16, 62, 4], and all fail to generalize to new sensors [19, 35] or scenes [4, 71].

Borrowing from both simulation and learning-based principles, we propose herein a novel pipeline that virtually replicates depth sensors and can be optimized for new use-cases either manually (e.g., providing known intrinsic pa-
2. Related work

Physics-based Simulation for Computer Vision. Researchers have already demonstrated the benefits of physics-based rendering of color images to deep-learning methods [23, 9], leveraging the extensive progress of computer graphics in the past decades. However, unlike color cameras, the simulation of depth sensors have not attracted as much attention. While it is straightforward to render synthetic 2.5D maps from 3D scenes (c.f. z-buffer graphics methods [60]), such perfect scans do not reflect the structural noise and measurement errors impairing real scans, leaving recognition methods trained on this synthetic modality ill-prepared to handle real data [51, 71, 50].

Early works [29, 14] tackling this *realism gap* tried to approximate the sensors’ noise with statistical functions that could not model all defects. More recent pipelines [19, 35, 51] are leveraging physics-based rendering to mimic the capture mechanisms of these sensors and render realistic depth scans, comprehensively modeling vital factors such as sensor noise, material reflectance, surface geometry, etc. These works also highlighted the value of proper 2.5D simulation for the training of more robust recognition methods [51, 50]. However, extensive domain knowledge (w.r.t. sensor and scene parameters) is required to properly configure these simulation tools. Unspecified information and unaccounted-for phenomena (e.g., unknown or patented software run by the target sensors) can only be manually approximated, impacting the scalability to new use-cases.

With DDS, we mitigate this problem by enabling the pipeline to learn missing parameters or optimize provided ones by itself. This is made possible by the recent progress in differentiable rendering, with techniques modelling complex ray-tracing and light transport phenomena with continuous functions and adequate sampling [40, 38, 72, 28]. More specifically, we build upon Li et al. rendering framework [38] based on ray-tracing and Monte-Carlo sampling.
Domain Adaptation and Randomization. Similar to efforts w.r.t. color-image domains, scientists have also been proposing domain-adaptation solutions specific to depth data, replacing or complementing simulation tools to train recognition methods. Most solutions rely on unsupervised conditional generative adversarial networks (GANs) [18] to learn a mapping from synthetic to real image distributions [5, 68, 36] or to extract features supposedly domain-invariant [17, 71]. Based on deep neural architectures trained on an unlabeled subset of target real data, these methods perform well over the specific image distribution inferred from these samples, but do not generalize beyond (i.e., they fail to map synthetic images to the real domain if the input images differ too much w.r.t. training data).

Some attempts to develop more scalable domain adaptation methods, i.e., detached from a specific real image domain (and therefore to the need for real training data), led to domain randomization techniques [61]. These methods apply randomized transformations (handcrafted [61, 70, 71] or learned [69]) to augment the training data, i.e., performing as an adversarial noise source that the recognition methods are trained against. The empirically substantiated claim behind is that, with enough variability added to the training set, real data may afterwards appear just as another noisy variation to the models. We can, however, conceptually understand the sub-optimal nature of these unconstrained domain adaptation techniques, which consider any image transform in the hope that they will be valuable to the task, regardless of their occurrence probability in real data.

By constraining the transforms and their trainable parameters to the optical and algorithmic phenomena actually impacting real devices, DDS can converge much faster towards the generation of images that are both valuable to learning frameworks and photorealistic.

3. Methodology

As illustrated in Figure 3, structured-light devices measure the scene depth in their field of view by projecting a light pattern onto the scene with their emitter. Their camera—tuned to the emitted wavelength(s)—captures the pattern’s reflection from the scene. Using the original pattern image \( I_o \) and the captured one \( I_c \) (usually filtered and undistorted) as a stereo signal, the devices infer the depth at every pixel by computing the discrepancy map between the images, i.e., the pixel displacements along the epipolar lines from one image to the other. The perceived depth \( z \) can be directly computed from the pixel disparity \( d \) via the formula \( z = \frac{b f}{d} \), with \( b \) baseline distance between the two focal centers and \( f \) focal length shared by the device’s emitter and camera. Note that depth sensors use light patterns that facilitate the discrepancy estimation, usually performed by block-matching algorithms [12, 32]. Finally, most depth sensors perform some post-processing to computationally refine their measurements (e.g., using hole-filling techniques to compensate for missing data).

In this paper, we consider the simulation of structured-light depth sensors as a function \( Z = G(\Phi) \), with \( \Phi = \{\Phi_s, \Phi_c, \Phi_e\} \) set of simulation parameters. \( G \) virtually reproduces the aforementioned sensing mechanisms, taking as inputs a virtual 3D scene defined by \( \Phi_s \) (e.g., scene geometry and materials), the camera’s parameters \( \Phi_c \) (e.g., intrinsic and extrinsic values) and the emitter’s \( \Phi_e \) (e.g., light pattern image or function \( \gamma_e \), distance \( b \) to the camera); and returns a synthetic depth scan \( Z \) as seen by the sensor, with realistic image quality/noise. We propose a simulation function \( G \) differentiable w.r.t. \( \Phi \), so that given any loss function \( L \) computed over \( Z \) (e.g., distance between \( Z \) and equivalent scan \( \hat{Z} \) from a real sensor), the simulation parameters \( \Phi \) can be optimized accordingly through gradient descent. The following section describes the proposed differentiable pipeline step by step, as shown in Figures 2 and 3.

3.1. Pattern Capture via Differentiable Ray-Tracing

To simulate realistic pattern projection and capture in a virtual 3D scene, we leverage recent developments in physics-based differentiable rendering [40, 38, 72, 28]. Each pixel color \( \gamma_e \) observed by the device camera is formalized as an integration over all light paths from the scene passing through the camera’s pixel filter (modelled as a con-
tinuous function \(k\), following the rendering equation:

\[
\gamma_e(\Phi) = \iiint k(x, y, \omega, \Phi_e)L(x, y, \omega; \Phi) \, dx \, dy \, d\omega, \tag{1}
\]

with \((x, y)\) continuous 2D coordinates in the viewpoint system, \(\omega\) light path direction, and \(L\) the radiance function modelling the light rays coming from the virtual scene (e.g., from ambient light and emissive/reflective surfaces) [38]. At any unit surface \(V\) projected onto \((x, y)\) (in viewpoint coordinate system), the radiance \(L\) with direction \(\omega\) is, therefore, itself integrated over the scene content:

\[
L(x, y, \omega; \Phi) = \int_{S^2} L_i(x, y, \omega; \Phi) f_\sigma(V, \omega; \omega_i) \, d\sigma(\omega_i) + L_V(x, y, \omega; \Phi_s), \tag{2}
\]

with \(L_V\) radiance emitted by the surface (e.g., for the structured-light-emitter or other light sources embodied in the scene), \(L_i\) incident radiance, \(f_\sigma\) bidirectional reflectance distribution function [46], \(d\sigma\) solid-angle measure, and \(S^2\) unit sphere [72]. As proposed by Li et al. [38], Monte Carlo sampling is used to estimate these integrals and their gradients: for continuous components of the integrand (e.g., inner surface shading), usual area sampling with automatic differentiation is applied, whereas discontinuities (e.g., surface edges) are handled via custom edge sampling.

More specific to our application, we simulate the structured-light pattern projection onto the scene and its primary contribution \(L_e\) to \(L\) for each unit surface \(V\) as:

\[
L_e(x, y, \omega; \Phi) = \gamma_e(x_e, y_e, \Phi_e)\eta(V, \Phi_e), \tag{3}
\]

with \((x_e, y_e, z_e)^T = M_eV\) projection of \(V\) into the pattern image coordinate system defined by the projection matrix \(M_e\), \(\gamma_e\) continuous representation of the structured-light pattern emitted by the sensor, and \(\eta\) light intensity (e.g., as a function of the distance to the emitter). In other words, for surfaces visible to the camera, we trace rays from them to the light emitter to measure which elements of its pattern are lighting the surfaces (c.f. steps 1-3 in Figure 3).

As highlighted in various studies [35, 34, 51, 50], due to the baseline distance between their emitter and camera, depth sensors suffer from shadow-related capture failure, i.e., when a surface \(V\) contributing to \(\gamma_e\) does not receive direct light from the emitter due to occlusion of the light rays by other scene elements (c.f. step 4 in Figure 3). Therefore, we propose a soft shadow mapping procedure [65, 1] that we model within the light intensity function \(\eta\) as follows:

\[
\eta(V) = \frac{\eta_e}{z_e}(1 - \sigma(z_e - \bar{z}_e - \xi)), \tag{4}
\]

with \(\sigma\) sigmoid operator (replacing the discontinuous step function used in traditional shadow mapping), \(\eta_e\) emitter intensity, and \(\bar{z}_e\) computed as \((x_e, y_e, \bar{z}_e)^T = M_eV_{col}\) where \(V_{col}\) is the first surface hit by the virtual ray thrown from the emitter focal center toward \(V\) (i.e., \(V_{col}\) superposed to \(V\) but closer in the emitter 2D coordinate system). We add an optimizable bias \(\xi \in \mathbb{R}\) to prevent shadow acne (shadow artifacts due to distance approximations) [8].

Estimating \(\gamma_e(\Phi)\) accounting for the scene and sensor properties \(\Phi\), we obtain the rasterized image \(I_e\). To cover non-modelled physics phenomena (e.g., lens defects) and according to previous works [19, 51], we also adopt an optional noise function \(f_n\) applied to \(I_e\), e.g., \(f_n(I_e) = I_e + \Delta I\), with \(\Delta I = \epsilon \sigma_n + \mu_n\), \(\{\mu_n, \sigma_n\} \in \Phi_e\), and \(\epsilon \sim \mathcal{N}(0, 1)\) (c.f. reparameterization trick [13, 42]).

### 3.1.1 Differentiable Stereo Block-Matching

Similar to real depth sensors, our pipeline then compares the computed \(I_e\) with a rasterized version \(I_o\) of the original pattern (both of size \(H \times W\)) to identify stereo-correspondences and infer the disparity map. Differentiable solutions to regress disparity maps from stereo signals have already been proposed, but these methods rely on CNN components to perform their task either more accurately [41, 6, 11] or more efficiently [30]. Therefore, they are bound to the image domain that they were trained over. Since our goal is to define a scene-agnostic simulation pipeline, we proposed instead an improved continuous implementation [30] of the classic stereo block-matching algorithm applied to disparity regression [32, 33], illustrated in Figure 3. The algorithm computes a matching cost volume \(C \in \mathbb{R}^{H \times W \times N_d}\) by sliding a \(w \times w\) window over the two images, comparing each block in \(I_e\) with the set of \(N_d\) blocks in \(I_o\) extracted along the same epipolar line. Considering standard depth sensors with the camera and emitter’s focal planes parallel, the epipolar lines appear horizontal in their image coordinate systems (with \(N_d = W\), simplifying the equation into:

\[
C(x, y, \delta) = \sum_{i=x+u}^{x+w} \sum_{j=y+v}^{y+w} \mathcal{L}_M(I_{c;i,j}; I_{o;i,j-\delta}), \tag{5}
\]

with \(\delta \in [y - N_d - w, y]\) horizontal pixel displacement and \(\mathcal{L}_M\) matching function (we opt for cross-correlation). Matrix unfolding operations are applied to facilitate volume inference. Formulating the task as a soft correspondence search, we reduce \(C\) into the disparity map \(d\) as follows: \(d(x, y) = \text{softargmax}_d C(x, y, \delta)\) with \(\text{softargmax}_d X = \sum_i \frac{\beta x_i}{\sum_i \beta x_i}\) and \(\beta \in \mathbb{R}\) optimizable parameter controlling the temperature of the underlying probability map. From this, we can infer the simulated depth scan \(Z = \frac{d}{d+2}\).

However, as it is, the block-matching method would rely on an excessively large cost volume \(H \times W \times W\) (i.e., with \(N_d = W\)) making inference and gradient computation impractical. We optimize the solution by considering
the measurement range \([z_{\text{min}}, z_{\text{max}}]\) of the actual sensor (e.g., provided by the manufacturer or inferred from focal length), reducing the correspondence search space accordingly, i.e., with \(\delta \in [d_{\text{min}}, d_{\text{max}}] = \lfloor [f_{\text{min}} b_{\text{max}}] \rfloor \lfloor [f_{\text{max}} b_{\text{min}}] \rfloor \) (dividing \(N_d\) tenfold for most sensors). The effective disparity range can be further reduced, e.g., by considering the min/max \(z\)-buffer values in the target 3D scene.

The computational budget saved through this scheme can instead be spent refining the depth map. Modern stereo block-matching algorithms perform fine-tuning steps to achieve sub-pixel disparity accuracy, though usually based on global optimization operations that are not directly differentiable [25, 44]. To improve the accuracy of our method without trading off its differentiability, we propose the following method adapted from [35]: Let \(n_{\text{sub}}\) be an hyperparameter representing the desired pixel fraction accuracy. We create \(\{I_{o,i}\}_{i=1}^{n_{\text{sub}}}\) lookup table of patterns containing a single flat surface parallel to the sensor focal planes placed at distance \(d_{\text{min},i} = d_{\text{min}} + i\) (hence a global disparity of \(i/n_{\text{sub}}\) between \(I_o\) and \(I_{o,i}\)). At simulation time, block-matching is performed between \(I_c\) and each \(I_{o,i}\), interlacing the resulting cost volumes and reducing them at once into the refined disparity map.

Finally, similar to the noise function optionally applied to \(I_c\) after capture, our pipeline allows \(Z\) to be post-processed, if non-modelled functions need to be accounted for (e.g., device’s hole-filling operation). In the following experiments, we present different simple post-processing examples (none, normal noise, or shallow CNN).

4. Experiments

Through various experiments, we evaluate the photorealism of depth images rendered by DDS and their value w.r.t. training recognition method or solving inverse problems.

4.1. Realism Study

First, we qualitatively and quantitatively compare DDS results with real sensor scans and data from other pipelines.

Qualitative Comparison. Visual results are shared in Figures 1 and 4 (w.r.t. Microsoft Kinect V1 simulation), as well as in the supplementary material (w.r.t. Matterport Pro2). We can observe that off-the-shelf DDS reproduces the image quality of standard depth sensors (e.g., Kinect V1): DDS scans contain shadow noise, quantization noise, stereo block-mismatching, etc., similar to real images and previous simulations [19, 51] (c.f. empirical study of depth sensors’ noise performed by Planche et al. [51]). Figure 4 and supplementary material further highlight how, unlike static simulations, ours can learn to tune up or down its inherent noise to better model sensors of various quality.

Quantitative Comparison. Reproducing the experimental protocol of previous 2.5D simulation methods [34, 51], we statistically model the depth error incurred by DDS as function of various scene parameters, and compare with empirical and statistical models from real sensor data.
• Protocol. Studying the Microsoft Kinect VI sensor, Landau et al. [35, 34] proposed the following protocol (further illustrated in the supplementary material). In real and simulated world, a flat surface is placed in front of the sensor. The surface is considered as a plane \((P, \vec{u}, \vec{v})\) with \(P = (0, 0, z)\), \(\vec{u} = (1, 0, 0)\), and \(\vec{v} = (0, \sin \alpha, \cos \alpha)\) in camera coordinate system (i.e., a plane at distance \(z\) and tilt angle \(\alpha\) w.r.t. focal plane). For each image captured in this setup, the standard depth error for each pixel \(q\) is computed as function of the distance \(z\), the tilt angle \(\alpha\), and the radial distance \(r\) to the focal center. Like Landau et al. [35, 34] and Planche et al. [51], we compare the noise functions of our method with those of the actual Kinect VI sensor, as well as the noise functions computed for other state-of-the-art simulation tools (BlenSor [19], Landau’s [35], and DepthSynth [51]) and noise models proposed by researchers studying this sensor (Menna et al. [43], Nguyen et al. [45] and Choo et al. [7, 34]).

• Results. In Figure 5, we first plot the error as a function of the radial distance \(r\) to the focal center. DDS performs realistically: like other physics-based simulations [19, 51], it reproduces the noise oscillations, with their amplitude increasing along with distance \(z\)—a phenomenon impairing real sensors, caused by pattern distortion. We also plot the standard error as a function of the distance \(z\) and of the incidence angle \(\alpha\). While our simulated results are close to the real ones w.r.t. distance, we can observe that noise is slightly over-induced w.r.t. tilt angle. The larger the angle, the more stretched the pattern appears on the surface, impairing the block-matching procedure. Most algorithms fail matching overly-stretched patterns (cf. exponential error in the figure), but our custom differentiable block-matching solution is unsurprisingly less robust to block skewing than the multi-pass methods used in other simulations [19, 51]. This could be tackled by adopting some more advanced block-matching strategies from the literature and rewriting them as continuous functions. This would however increase the computational footprint of the overall simulation and would only benefit applications where high photorealism is the end target. In the next experiments, we instead focus on deep-learning applications.

4.2. Applications to Deep Learning

We now illustrate how deep-learning solutions can benefit from our simulation method. We opt for various key recognition tasks over standard datasets, comparing the performance of well-known CNNs as a function of the data and the domain adaptation framework used to train them.

2.5D Semantic Segmentation. We start by comparing the impact of simulation tools on the training of a standard CNN for depth-based semantic segmentation.

• Dataset. For this task, we choose the 2D-3D-Semantic dataset by Armeni et al. [3] as it contains RGB-D indoor scans shot with a Matterport Pro2 sensor, as well as the camera pose annotations and the reconstructed 3D models of the 6 scenes. It is, therefore, possible to render synthetic images aligned with the real ones. We split the data into training/testing sets as suggested by 2D-3D-S authors [3] (fold #1, i.e., 5 training scenes and 1 testing one). For the training set, we assume that only the 3D models, images and their pose labels are available (not the ground-truth semantic masks). Note also that for the task, we consider only the 8 semantic classes (out of 13) that are discernible in depth scans (e.g., board are indistinguishable from wall in 2.5D scans) and present in the training scenes.

• Protocol. Using the 3D models of the 5 training scenes, we render synthetic 2.5D images and their corresponding semantic masks using a variety of methods from the literature [2, 19, 51]. DDS is both applied off-the-shelf (only entering the Pro2 sensor’s intrinsic information), and after being optimized via supervised gradient descent (combining Huber and depth-gradient losses [24, 27]) against the real scans from one training scene (scene #3). Each synthetic dataset, and the dataset of real scans as upper-bound target, is then used to train an instance of a standard ResNet-based CNN [20, 64, 67] for semantic segmentation (we choose the Dice loss to make up for class imbalance [10]).

• Results. We measure the performance of each model instance in terms of per-class mean intersection-over-union [26, 53] and pixel accuracy. Results are shared in Table 1. We can observe how data from both untrained and trained DDS result in the most accurate recognition models (among those trained on purely synthetic data), with values on par or above those of the models trained on real annotated data for some classes. Even though DDS may not perfectly simulate the complex, multi-shot Matterport sensor, its ability to render larger and more diverse datasets can be easily leveraged to achieve high recognition accuracy.

| Train. Data Source | Mean Intersection-Over-Union (mIoU) | Pixel Acc. |
|-------------------|-----------------------------------|------------|
| clean             | 0.003 0.018 0.002 0.087 0.012 0.052 0.091 0.351 | 35.3%      |
| BlenSor (train.)  | .110 .534 .119 .167 .148 .561 .082 .412 | 51.6%      |
| DepthS. (train.)  | .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%      |

Classification and Pose Estimation. We now perform an extensive comparison, as well as partial ablation study, w.r.t.
the ubiquitous computer vision task of instance classification and pose estimation (ICPE) [66, 5, 70, 71].

- **Dataset.** For this task, we select the commonly-used Cropped LineMOD dataset [21, 66, 5], composed of 64 × 64 RGB-D image patches of 11 objects under various poses, captured by a Kinect VI sensor, in cluttered environments. Disregarding the RGB modality for this experiment, we split the dataset into a non-annotated training set $X_{trn}$ of 11,644 depth images, and a testing set $X_{rtn}$ of 2,919 depth images with their class and pose labels. The LineMOD dataset also provides a reconstructed 3D model of each object, used to render annotated synthetic training images. For fair comparison, all 3D rendering methods considered in this experiment are provided the same set of 47,268 viewpoints from which to render the images. These viewpoints are sampled from a virtual half-icosahedron centered on each target object, with 3 different in-plane rotations (i.e., rotating the camera around its optical axis) [66, 70, 71, 52].

- **Protocol.** For this experiment, we opt for the generic task CNN from [16], trained for object classification and rotation estimation via the loss $L_{icpe} = E_{x,(y,q)} [-y^T \log \hat{y} + \xi \log (1 - |q^T \hat{q}|)]$, where the first term is the class-related cross-entropy and the second term is the log of a 3D rotation metric for quaternions [5, 69], with $\xi$ pose loss factor, $x$ input depth image, $\{y, q\}$ resp. ground-truth one-hot class vector and quaternion, and $\{\hat{y}, \hat{q}\}$ resp. predicted values. Again, we measure the network’s classification accuracy and rotational error as a function of the data that it was trained on, extending the comparison to different online or offline augmentation and domain adaptation schemes (c.f. Figure 4 for visual comparison).

For domain adaptation solutions such as PixelDA [5] and DeceptionNet [69], the recognition network $T$ is trained against a generative network $G$ whose task is to augment the input synthetic images before passing them to $T$. This adversarial training framework, with $G$ trained unsupervisedly against $T$ [69] and/or a discriminator network $D$ [5, 71] using non-annotated real images $X_{trn}$, better prepares $T$ for its task on real data, i.e., training it on noisier and/or more realistic synthetic images. To further demonstrate the training of our simulation, this time in a less constrained, unsupervised setting, we reuse PixelDA training framework, replacing its ResNet-based [20] generator by DDS. Our method is, therefore, unsupervisedly trained along with the task network, so that DDS learns to render synthetic images increasingly optimized to help $T$ with its training. Three instance of DDS are thus compared: (a) off-the-shelf, (b) with $\Phi = \{\xi, \mu_n, \sigma_n, \beta\}$ (i.e., parameters w.r.t. shadows, normal noise, and softargmax) optimized unsupervisedly, and (c) same as the previous but adding 2 trainable convolution layers as post-processing ($|\Phi| = 2,535$ only in total).

- **Results.** Table 2 presents a detailed picture of state-of-the-art training solutions for scarce-data scenarios (basic or simulation-based image generation, static or GAN-based offline or online image transformations, etc.) and their performance on the task at hand. The various schemes are further sorted based on their requirements w.r.t. unlabeled real images and on the size of their parameter space.

| Augmentations | Sim/DA Req. | Class. Accur. | Rot. Error |
|---------------|-------------|---------------|------------|
| offline       | online      | $X_{trn}$ | $|\Phi|$ | |
| Basic         | DR          | 46.8%        | 70.7%      | 53.1% |
| PixelDA [5]   | GAN         | ✓ | 1.96M | 85.7% | 40.5% |
| DRIT++ [51]   | GAN DR      | ✓ | 12.3M | 68.0% | 60.8% |
| DecepNet [69] | DR          | 1.54M | 80.2% | 54.1% |
| Depths [31]   | Sim DR      | 71.5% | 76.6% | 45.4% |
| BlenSor [19]  | Sim DR      | 67.5% | 82.6% | 41.4% |
| DDS (untrained) | Sim DR | 69.7% | 67.6% | 39.7% |
| Combined      | Sim DR      | ✓ | 2,535 | 81.2% | 49.1% |
| DDS + conv    | Sim DR      | ✓ | 2,535 | 90.5% | 39.4% |
| DDS Sim + conv | Sim DR     | ✓ | 2,535 | 85.5% | 45.4% |
| DDS Sim + conv | Sim DR     | ✓ | 2,535 | 93.0% | 31.3% |
| DDS + Sim, V | Sim + conv  | ✓ | 2,535 | 97.8% | 25.1% |
| DDS + Sim, V  | Sim + conv  | ✓ | 2,535 | 95.4% | 35.0% |

The table confirms the benefits of rendering realistic data, with the recognition models trained against previous simulation methods [19, 51] performing almost as well as the instances trained with GAN-based domain adaptation techniques [5, 36] having access to a large set of relevant real images. In contrast to the latter methods, simulation tools have, therefore, superior generalization capability. A second interesting observation from the table is the value of online data augmentation (e.g., random distortion, occlusion, etc.) [71], regardless of the quality of synthetic images. It provides a significant accuracy boost on both tasks, virtually and inexpensively increasing the training set size and variability c.f. domain randomization theory [61]. In that regard, DeceptionNet [69], a learning-based domain randomization framework, performs satisfyingly well without the need for real data (though domain knowledge is required to adequately set the 2.5D transforms’ hyperparameters).

But overall, results highlight the benefits of combining all these techniques, which DDS can do seamlessly thanks to its gradient-based structure. Off-the-shelf, manually-parameterized DDS yields results similar to previous simulation tools when images are not further augmented but rises above all other methods when adding online augmen-
tions. Training DDS unsupervisedly along with $T$ further increases the performance, especially when intermittently applying a learned post-processing composed only of two convolutions. Opting for simple post-processing modules to compensate for non-modelled phenomena, we preserve the key role of simulation within DDS and, therefore, its generalization capability. Finally, we can note that, while the instance of $T$ trained with DDS still performs slightly worse than the one trained on real annotated images w.r.t. the classification task, it outperforms it on the pose estimation task. This is likely due to the finer pose distribution in the rendered dataset (47,268 different images covering every angle of the objects) compared to the smaller real dataset. The best performance w.r.t both tasks is achieved by combining the information in the real dataset with simulation-based data (c.f. penultimate line in Table 2).

Though computationally more intensive (a matter that can be offset by rendering images offline), our differentiable solution outperforms all other learning-based domain adaptation schemes, with a fraction of the parameters to train (therefore requiring fewer iterations to converge). Moreover, it is out-of-the-box as valuable as other depth simulation methods and outperforms them too when used within supervised or unsupervised training frameworks.

### 4.3. Optimization of Scene and Sensor Parameters

So far, we mostly focused on optimizing the simulation itself (e.g., shadow bias and noise parameters) in order to render more realistic images and improve CNNs training, rather than optimizing the scene or sensor parameters. To illustrate DDS capability w.r.t. such use-cases, we developed and performed a toy experiment, presented in Figure 6.

- **Protocol.** We consider the same scene setup as in Subsection 4.1 but assume that the target surface is tilted w.r.t. optical plan and only reflects red light frequencies, and that the depth sensor relies on a randomly generated dot pattern emitted with pseudo white light (mixture of wavelengths).

- **Results.** First, in Figure 6.B, we demonstrate how the scene geometry (i.e., the pose of the flat surface here) can be optimized via gradient descent to reduce the standard error of the simulated device (i.e., using the L1 distance between simulated depth maps and ground-truth noiseless ones as loss function). As expected, the surface is rotated back to be parallel to the focal plane, effectively preventing the stretching of the projected pattern and, therefore, block-matching issues (c.f. discussion in Subsection 4.1). In a second experiment, we consider the scene parameters fixed and instead try optimizing the depth sensor, focusing on its light pattern (i.e., to reduce sensing errors w.r.t. this kind of scenes, composed of tilted, red surfaces). Figure 6.C shows how the pattern image is optimized, quickly switching to red light frequencies, as well as more slowly adopting local patterns less impacted by projection-induced stretching.

We believe these toy examples illustrate the possible applications of simulation-based optimization of scene parameters (e.g., to reduce noise from surroundings when scanning an object) or sensor parameters (e.g., to build a sensor optimized to specific scene conditions).

### 5. Conclusion

In this paper we presented a novel simulation pipeline for structured-light depth sensors, based on custom differentiable rendering and block-matching operations. While directly performing as well as other simulation tools w.r.t. generating realistic training images for computer-vision applications, our method can also be further optimized and leveraged within a variety of supervised or unsupervised training frameworks, thanks to its end-to-end differentiability. Such gradient-based optimization can compensate for missing simulation parameters or non-modelled phenomena. Through various studies, we demonstrate the realistic quality of the synthetic depth images that DDS generates, and how depth-based recognition methods can greatly benefit from it to improve their end performance on real data, compared to other simulation tools or learning-based schemes used in scarce-data scenarios. Our results suggest that the proposed differentiable simulation and its standalone components further bridge the gap between real and synthetic depth data distributions, and will prove useful to larger computer-vision pipelines, as a transformer function mapping 3D data and realistic 2.5D scans.
References

[1] Tomas Akenine-Möller, Eric Haines, and Naty Hoffman. Real-time rendering. CRC Press, 2019. 4

[2] Aitor Aldoma, Zoltan-Csaba Marton, Federico Tombari, Walter Wohlkinger, Christian Potthast, Bernhard Zeisl, Radu Bogdan Rusu, Suat Gedikli, and Markus Vincze. Point cloud library. *IEEE Robotics & Automation Magazine*, 1070(9932/12), 2012. 6

[3] 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. 6, 12, 13, 14

[4] Konstantinos Boumansalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. *arXiv preprint arXiv:1612.05424*, 2016. 1

[5] Konstantinos Boumansalis, 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, 2, 3, 5, 7, 11, 15, 16

[6] Jiu-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5410–5418, 2018. 4

[7] Benjamin Choo, Michael Landau, Michael DeVore, and Peter A Beling. Statistical-analysis-based error models for the microsoft kinect3d depth sensor. *Sensors*, 14(9):17430–17450, 2014. 6

[8] Joey de Vries. LearnOpenGL - Shadow Mapping. https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping. Accessed: 2021-03-10. 4

[9] Maximilian Denninger, Martin Sundermeyer, Dominik Winkelbauer, Youssef Zidan, Dmitry Olefir, Mohamad Elbadrawy, Ahsan Lodhi, and Harinandan Katam. Blenderproc. *arXiv preprint arXiv:1911.01911*, 2019. 1, 2, 15

[10] Michal Drozdzal, Eugene Vorontsov, Gabriel Charttrand, Samuel Kadoury, and Chris Pal. The importance of skip connections in biomedical image segmentation. In *Deep learning and data labeling for medical applications*, pages 179–187. Springer, 2016. 6

[11] Shivam Duggal, Shlenong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4348–4393, 2019. 4

[12] Nils Einecke and Julian Eggert. A multi-block-matching approach for stereo. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 585–592. IEEE, 2015. 3

[13] Otto Fabius, Joost R van Amersfoort, and Diederik P Kingma. Variational recurrent auto-encoders. In *ICLR (Workshop)*, 2015. 4

[14] Maurice F Fallon, Hordur Johannsson, and John J Leonard. Point cloud simulation & applications, 2012. http://www.pointclouds.org/assets/icra2012/localization.pdf. Accessed: 2020-09-23. 2

[15] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Adv. Neural Inform. Process. Syst.*, pages 611–619, 2012. 1

[16] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189, 2015. 1, 7, 16

[17] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 3

[18] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 3

[19] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: blender sensor simulation toolbox. In *Advances in Visual Computing*, pages 199–208. Springer, 2011. 1, 2, 4, 5, 6, 7, 13, 14, 16

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 6, 7, 14

[21] 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. 1, 5, 7, 15, 16

[22] 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. 13

[23] Tomáš Hodaň, Vihvah 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. 1, 2, 15

[24] Peter J Huber. Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer, 1992. 6, 11

[25] Martin Humenberger, Christian Zinner, Michael Weber, Wilfried Kubiinger, and Markus Vincze. A fast stereo matching algorithm suitable for embedded real-time systems. *Computer Vision and Image Understanding*, 114(11):1180–1202, 2010. 5

[26] Paul Jaccard. The distribution of the flora in the alpine zone. 1. *New phytologist*, 11(2):37–50, 1912. 6

[27] 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. 6, 11

[28] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020. 2, 3
Supplementary Material

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 [48], 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 [37] kindly provided by Li et al. [38]. Optional learning-based post-processing is performed by two convolutional layers, resp. with 32 filters of size $5 \times 5$ and 32 filters of size $1 \times 1$. 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 [31] with a learning rate of 0.001 and no weight decay. For supervised optimization, we opt for a combination of Huber loss [24] and gradient loss [27] (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 [5], 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:

$$I_c = \begin{bmatrix} I_{c,0} \\ I_{c,1} \\ \vdots \\ I_{c,m} \end{bmatrix}, \quad I_o = \begin{bmatrix} I_{o,0} \\ I_{o,1} \\ \ldots \\ I_{o,m} \end{bmatrix}, \quad (6)$$

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 6 does not account for this overlapping).
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 [32, 33] often used in depth simulation.

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

**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 VI 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;
- Sensor range $[z_{min}, z_{max}] = [400\text{mm}, 8000\text{mm}]$;
- Block size $w = 11$px;
- Emitted light intensity factor $\eta_c = 1.5 \times 10^{12}$;
- Shadow bias $\xi = 1$mm;
- Softargmax temperature parameter $\beta = 25.0$;
- Subpixel refinement level $n_{sub} = 4$;
Figure S3: **Qualitative comparison of simulated scans.** Synthetic depth images rendered from reconstructed 3D indoor scenes of the 2D-3D-Semantic dataset [3], 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.

Note that device-related parameters come from the sensors’ manufacturers or previous Kinect studies [35, 34]. Other parameters have been manually set through empirical evaluation. For the structured-light pattern, we use the Kinect pattern image reverse-engineered by Reichinger [54].

**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 stand-alone 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 [32, 33] and used by previous depth sensor simulations [19, 51]. Stereo color images come from the Middlebury Stereo dataset [57, 56, 22]. 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 soft-argmax-based reduction performed by our method (whereas Konolige’s algorithm yields null values when the correspondences are too ambiguous).
Figure S4: Experimental setup for quantitative noise study of a depth sensor, as proposed by Landau et al. [35].

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 [3], 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 [3]. We can observe how the DepthSynth method proposed by Planche et al. [51] 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 [32, 33], as the authors rely on a popular but rather antiquated method, certainly not as robust 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 [19], 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 $\alpha$ w.r.t. the focal plane (with $\hat{f}$ its normal).

Note that for this experiment, we use the experimental data collected and kindly provided by Landau et al. [35].

### Table S1: Comparative study w.r.t. training usage (extending study in Table 1), measuring the accuracy of a CNN [20, 64, 67] performing semantic segmentation on real 2.5D scans from the indoor 2D-3D-S dataset [3], as a function of the method used to render its training data and as a function of real annotated data availability ($\uparrow$ is the higher the value, the better).

| Train. Data Source | 0.003 | 0.018 | 0.002 | 0.087 | 0.012 | 0.052 | 0.091 | 0.351 | 35.3% |
|--------------------|-------|-------|-------|-------|-------|-------|-------|-------|------|
| clean              |       |       |       |       |       |       |       |       |      |
| BlenSor[19]        | 0.110 | 0.534 | 0.179 | 0.167 | 0.148 | 0.561 | 0.082 | 0.412 | 51.6% |
| DepthS. [19]       | 0.184 | 0.691 | 0.185 | 0.221 | 0.243 | 0.722 | 0.235 | 0.561 | 65.3% |
| DDS                | 0.218 | 0.705 | 0.201 | 0.255 | 0.240 | 0.742 | 0.259 | 0.583 | 62.9% |
| DDS (train.)       | 0.243 | 0.711 | 0.264 | 0.255 | 0.269 | 0.794 | 0.271 | 0.602 | 69.8% |
| real               | 0.135 | 0.770 | 0.214 | 0.277 | 0.302 | 0.803 | 0.275 | 0.661 | 73.5% |
| BlenSor[19] + real | 0.143 | 0.759 | 0.213 | 0.275 | 0.306 | 0.817 | 0.271 | 0.636 | 73.6% |
| DepthS. [19] + real| 0.222 | 0.767 | 0.254 | 0.297 | 0.325 | 0.812 | 0.273 | 0.659 | 75.8% |
| DDS + real         | 0.279 | 0.775 | 0.249 | 0.299 | 0.356 | 0.815 | 0.280 | 0.659 | 76.7% |

As for the BlenSor simulation [19], 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).

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.
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 [21, 5, 69]. 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 [9, 23].

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 [37, 38] to our needs. Finally, credits go to Pierre Yves P. [47] 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.* [71, 69] or adding random 3D clutter to the virtual scenes before rendering) on the training of a CNN [16] for depth-based instance classification and pose estimation on the *Cropped LineMOD* dataset [21, 5, 69].

| 3D Clutter in Scene | Augmentations | Sim/DA Req. | Class. Accur. † | Rot. Error † |
|----------------------|---------------|-------------|----------------|--------------|
|                      | offline       | online      | $X_{trn}^r$ | $\Phi$ | $\Phi_D$ |
| Basic                |               | DR          | 21.3% | 91.8° | 39.6% | 73.3° |
|                      | ✓             | DR          | 46.8% | 67.0° | 70.7% | 53.1° |
| PixelDA [5]          | GAN           | ✓           | 1.96M | 693k | 65.8% | 56.5° |
|                      | ✓             | GAN         | 1.96M | 693k | 85.7% | 40.5° |
| DRIT++ [36]          | 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 [69]    | DR            | 1.54M       | 37.3% | 59.8° | 80.2% | 54.1° |
|                      | ✓             | DR          | 1.54M |       |       |
| DepthSynth [51]      | Sim           | Sim DR      | 17.1% | 87.5° | 45.6% | 65.4° |
|                      | ✓             | Sim DR      | 71.5% | 52.1° | 76.6% | 45.4° |
| BlenSor [19]         | Sim           | Sim DR      | 14.9% | 90.1° | 45.6% | 65.3° |
|                      | ✓             | Sim DR      | 67.5% | 63.4° | 82.6% | 41.4° |
| DDS (untrained)      | Sim           | Sim DR      | 15.6% | 91.6° | 50.0% | 68.9° |
|                      | ✓             | Sim DR      | 69.7% | 67.6° | 89.6% | 39.7° |
| DDS                  | Sim           | ✓ 4 693k   | 21.3% | 80.9° | 51.6% | 63.3° |
|                      | Sim DR        | ✓ 4 693k   | 81.2% | 49.1° | 90.5% | 39.4° |
|                      | Sim+conv      | ✓ 2,535 693k | 22.6% | 78.7° | 85.5% | 45.4° |
|                      | Sim+conv DR   | ✓ 2,535 693k | 54.3% | 60.4° | 93.0% | 31.3° |
| DDS + $(X, Y)_{trn}^r$ | ✓ Sim+conv DR | ✓ 2,535 693k | 97.8% | 25.1° |       |
| $(X, Y)_{trn}^r$     | ✓             | ✓           | 95.4% | 35.0° |       |