Advances in Artificial Intelligence and Machine Learning; Research 3 (2) 977-995 Received 17-03-2023; Accepted 29-04-2023; Published 07-04-2023

CAD2Render: A Synthetic Data Generator for Training Object Detection and Pose Estimation Models in Industrial Environments.*

Steven Moonen steven.moonen@uhasselt.be
Expertise Centre for Digital Media
Hasselt University - tUL - Flanders Make
Hasselt, Martelarenlaan 42, Belgium

Bram Vanherle bram.vanherle@uhasselt.be
Expertise Centre for Digital Media
Hasselt University - tUL - Flanders Make
Hasselt, Martelarenlaan 42, Belgium

Joris de Hoog joris.dehoog@flandersmake.be
Flanders Make
Leuven, Gaston Geenslaan 8- B-3001, Belgium

Taoufik Bourgana taoufik.bourgana@flandersmake.be
Flanders Make
Leuven, Gaston Geenslaan 8- B-3001, Belgium

Abdellatif Bey-Temsamani abdellatif.bey-temsamani@flandersmake.be
Flanders Make
Leuven, Gaston Geenslaan 8- B-3001, Belgium

Nick Michiels nick.michiels@uhasselt.be
Expertise Centre for Digital Media
Hasselt University - tUL - Flanders Make
Hasselt, Martelarenlaan 42, Belgium

Corresponding Author: Steven Moonen

Abstract

Computer vision systems become more wide spread in the manufacturing industry for automating tasks. As these vision systems use more and more machine learning opposed to the classic vision algorithms, streamlining the process of creating the training datasets become more important. Creating large labeled datasets is a tedious and time consuming process that makes it expensive. Especially in a low-volume high-variance manufacturing environment. To reduce the costs of creating training datasets we introduce CAD2Render, a GPU-accelerated synthetic data generator based on the Unity High Definition Render Pipeline

* This is an extended and substantially revised version of the paper "CAD2Render: A Modular Toolkit for GPU-Accelerated Photorealistic Synthetic Data Generation for the Manufacturing Industry" previously published in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops, 2023, pp. 583-592

Citation: Steven Moonen, et al. CAD2Render: A Synthetic Data Generator for Training Object Detection and Pose Estimation Models in Industrial Environments. Advances in Artificial Intelligence and Machine Learning. 2023;3(2):58.

(HDRP). CAD2Render streamlines the process of creating highly customizable synthetic datasets with a modular design for a wide range of variation settings. We validate our toolkit by showcasing the performance of AI vision models trained purely with synthetic data. The performance is tested on object detection and pose estimation problems in a variate of industrial relevant use cases. The code for CAD2Render is available at `https://github.com/EDM-Research/CAD2Render`

Keywords: Synthetic Data, Computer vision, Machine Learning, Graphics

1. INTRODUCTION

Machine vision has been around in the industrial landscape for a couple of decades and the recent surge in popularity has made the technology a major innovation driver for manufacturers. Most of the approaches are relying on classic vision and are finetuned towards the inspection system. Although they are able to achieve great performance, they require full control of the environment where all the operating conditions stay constant. As such, they are prone to failure when variables like lighting or backgrounds might change. Machine learning algorithms, on the other hand, can be trained to handle such variations, but in turn, have some major challenges that need to be addressed before being fully adopted in this industrial domain.

A first key challenge is the need for large annotated training sets, which are tedious, time consuming and very costly to acquire. In the majority of cases, the data has to be annotated manually, which can lead to bias or errors caused by the human annotator. This limitation is more pronounced in the manufacturing industry, as they have taken some major steps forward in flexible assembly and product manufacturing, allowing them to transform their pipeline towards flexible low-volume and high-variance production. In the extreme case, each produced product can be of a different shape (e.g. prosthesis manufacturing), where there is simply no time to manually capture and annotate datasets. A second challenge is the complex outlook of the materials used in manufacturing. Products are often made of metallic-like materials that cause detailed and complex reflections. It is very challenging for a vision technique to generalize to all the possible and complex lighting effects.

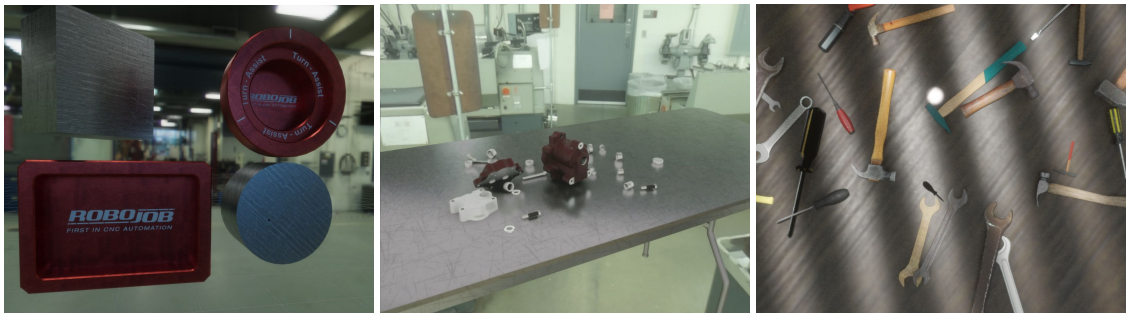

Figure 1: Annotated training set examples of different use cases generated with CAD2Render. left: CNC fabrication, middle: compressor parts, right: tool detection.

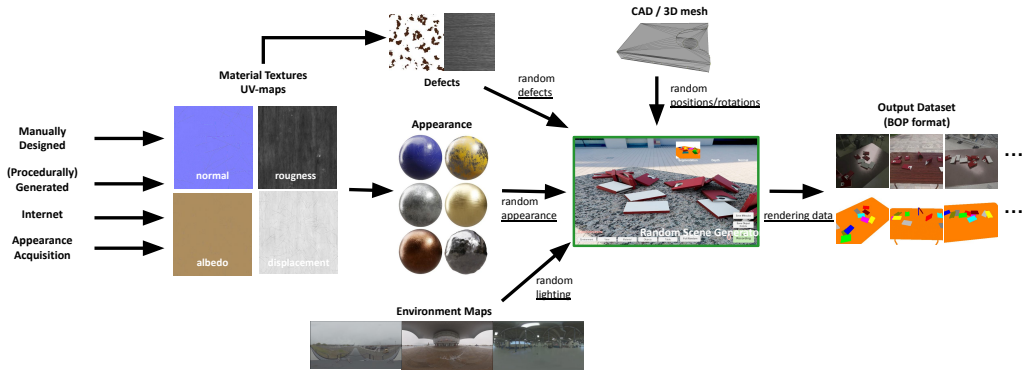

Figure 2: Schematic overview of CAD2Render. Environment maps, CAD files and material properties are imported in CAD2Render and used to create a high variety of 3D scene's. Further variations to the material textures can be created by introducing defects like rust or scratches. The scenes are then rendered with a path tracer to create a dataset for training machine learning algorithms.

A common approach to cope with small training datasets is data augmentation: slightly distorting the available data points to create new points that still belong to the same category. While it can provide better performance, it still requires a minimum of data and it does not take into account that the actual products are three-dimensional objects and their visual appearance is governed by complex material properties, lighting conditions, and geometrical detail. This is especially relevant in the manufacturing industry with the metallic-like materials.

This paper exploits the availability of CAD models and the domain knowledge of the manufacturing process, together with the recent advancements in real-time ray tracing, to propose a modular toolkit called CAD2Render. The proposed toolkit is able to automatically generate large amounts of photorealistic training images with extensive visual variations and their accompanying annotations for machine learning purposes. These annotations are generated without human errors or bias. A schematic view of the toolkit is given in FIGURE 2. We selected two industrial relevant algorithms for validation, i.e. bin picking by means of object detection and pose estimation, and keypoint detection. Both are important tasks for automating industrial setups, requiring complex vision algorithms. We show that we can achieve a performance capable of solving these tasks in an industrial relevant environment when training on datasets generated by CAD2Render.

2. RELATED WORK

Decreasing training set sizes is a popular area of research. Data augmentation is a widely adopted approach to increase the variability in datasets. By slightly distorting the few available images, new examples are created without the need of relabeling the data. The most common variations used are randomly cropping, rotating, scaling, mirroring, color balancing, and adjusting brightness of the entire collection of training pictures to create many slightly modified copies [1, 2]. More recently, similar basic image-based variations are generated using deep learning [3]. All of these

modifications are limited by the information contained in the original 2D pictures, and by the fact that each modification induces a loss of quality. On the other hand, the actual products are three-dimensional and their visual appearance is governed by complex material properties, lighting conditions, and geometrical detail.

Because of the scarcity of training data, recent works propose techniques for training machine learning models purely on synthetic data and have shown that it can achieve similar results compared to SOTA [4, 5]. An important conclusion they draw is that realism in the synthetic data is a key factor. Tobin et al. [6], demonstrate the importance of domain randomization when using synthetic data. In their work they show that the domain gap between real and synthetic data can be bridged by introducing enough variations in the synthetic data generation. The machine learning algorithms will see the domain gap as yet another variation of the synthetic data.

Rendering photorealistic images is a complex task and all the settings for 3D geometry, lighting and materials have to be meticulously modelled in order to achieve convincing photorealism [7]. In addition, rendering large datasets is a time consuming task, certainly when the generator is based on a ray tracing algorithm to generate the data. This can be a limiting problem because, in low-volume and high-variance manufacturing, new datasets need to be created in a short amount of time. To speed up this task, it can be distributed on a computer cluster or parallelized on a GPU. Kubric created by Greff et al. [8], is a data generation pipeline that is designed to both work on a single computer to facilitate prototyping or small dataset generation, as well as to run on large computer clusters to speed up the generation. This is only useful when you have access to a computer cluster. Isaac sim [7], solves this by using GPU accelerated ray tracing. There focus is on large scene simulations of an entire factory hall. While they do support many object variations these need to be manually implemented with there python API.

Jeong et al.[9], used a NeRF variant to extend a dataset of real images with new viewpoints. This limits the amount of real data required for or a sufficiently large dataset. With NeRF they are able to represent hundreds of photorealistic images in a single format, also reducing the storage size required for the dataset. The data synthesized with this technique can reach better photorealism then rendering a scene from scratch. It also doesn't require an expert optimizing the appearance of a digital scene. However, this technique is not able to add highly customizable variations to the data.

Other approaches that use 3D rendering focus on a narrow scope of application, such as side view rendering for face detection [10], stereo rendering for depth estimation [11], vehicle detection [12, 13], or large scale factory simulations [7]. In contrast, most products in industrial manufacturing settings have complex material properties and as a result undergo intricate lighting effects when changing the viewpoint or lighting properties. To include these variations, this paper will focus on algorithms to varying this complex light propagation to cost-effectively synthesize large amounts of training data with accurate and realistic variations of lighting conditions, viewpoints, surface properties, etc. 3D CAD models provided by the manufacturers will be utilized to support this process.

3. CAD2Render

CAD2Render is designed as a modular and highly customizable toolkit built upon the HDRP pipeline of Unity3D [14], for generating high quality synthetic data for deep learning purposes. It focuses on photorealism by including global illumination effects. A high level overview is provided in FIGURE 2. Inspired by the key insight of Tobin et al. [6], that domain randomization is a powerful tool to successfully exploit synthetic data for training deep learning models, we argue that CAD2Render should support a wide set of complex variations. CAD2Render supports variations such as model types, number of models, instancing, environments lighting, viewpoints, exposure, supporting structures, materials, material appearance, textures, etc. These variations are added in a modular fashion and can be enabled, disabled or extended in function of the use case.

3.1 Modular Variations

To facilitate the need for broad and complex variations in the training data, we introduce a wide range of modular randomizers that can introduce different types of variations in the synthetic data. For the clarity of this paper, we have categorized the modules based on pose, lighting, appearance and miscellaneous variations. This section describes how these variations are generated.

Camera Variations
The camera pose is randomly defined in spherical coordinates (θ, ϕ, r), in a sphere around a point of interest, orientated towards this point. The user can define minimum and maximum ranges for these parameters which are then uniformly sampled within this range. The intrinsic parameters of the camera can also be adjusted to match an existing camera. To further match a physical setup it is also possible to import exact camera poses from a BOP dataset.

Object Variations
The object pose is randomized by automatically spawning new objects in the scene. For each dataset, the user can setup a spawning volume, which defines the 3D region where new objects can be instantiated. Furthermore, the user specifies a ”model path” that contains the actual models to be spawned, in the form of prefabs. These prefabs can be very simple, just a mesh representation of the CAD model, or can be fully tailored to the use case. CAD2Render will randomly select a set of 3D models from this folder and instantiates them in the scene. The user can specify how many random object are spawned per generated image and if each model is unique or can be instantiated multiple times. In addition, the built-in Nvidia PhysX engine can be enabled or disabled to simulate the objects falling in a natural pose. If enabled, the scene requires a supporting structure, for example a table or pallet.

Lighting Variations
Training datasets need sufficient variation in lighting to make deep learning techniques robust for sudden changes in environmental effects [15]. Two types of variations in light are supported by CAD2Render. First, random high dynamic range environment maps are applied to each rendered image. The environment maps are randomly selected from a user specified path. Furthermore, randomized exposure and rotation of the environment map is supported. Second, the user can specify additional light source prefabs, acting as templates for additional 3D light sources. Dur-

ing rendering, for each generated image, parameters such as the number or 3D light sources, the intensity, position, rotation and radius can be set and randomized.

An example of lighting variations is given in FIGURE 3. The FIGURE shows examples of environment map variations and variations of intense highlights, shadows and color. We argue that this type of complex photorealistic variations in reflections, shadows and highlights are crucial to incorporate in the training set, because a trained pose or object detector needs to be able to differentiate between what is the actual object and what are the complex light effects that can confuse the model. The more complex light variations the model sees during training, the more it is robust to such changes in a real context. In addition, support for projector variations is available, where the projection of patterns or images can be simulated (example in FIGURE 1 right).

Appearance Variations

The appearance of each instantiated object can be varied on-the-fly. The assigned material properties, in the form of normal, roughness, albedo and displacement maps, can originate from different sources. They can be manually designed, extracted from real sample materials, extracted of the internet or (procedurally) generated (see FIGURE 2 on the left). In the case of the former three, a database of existing material models can be passed to the CAD2Render toolkit. In the case of the latter, variations of material textures can be automatically generated by the toolkit. At the moment, the toolkit supports three types of texture generators that are industrially relevant: scratches, rust and polishing lines. The settings of these generators can be tailored the the specific needs of the use case at hand. CAD2Render can randomly select and apply materials from a user defined path, similar to the environment maps. Additional variations can be set, with random parameters for HSV

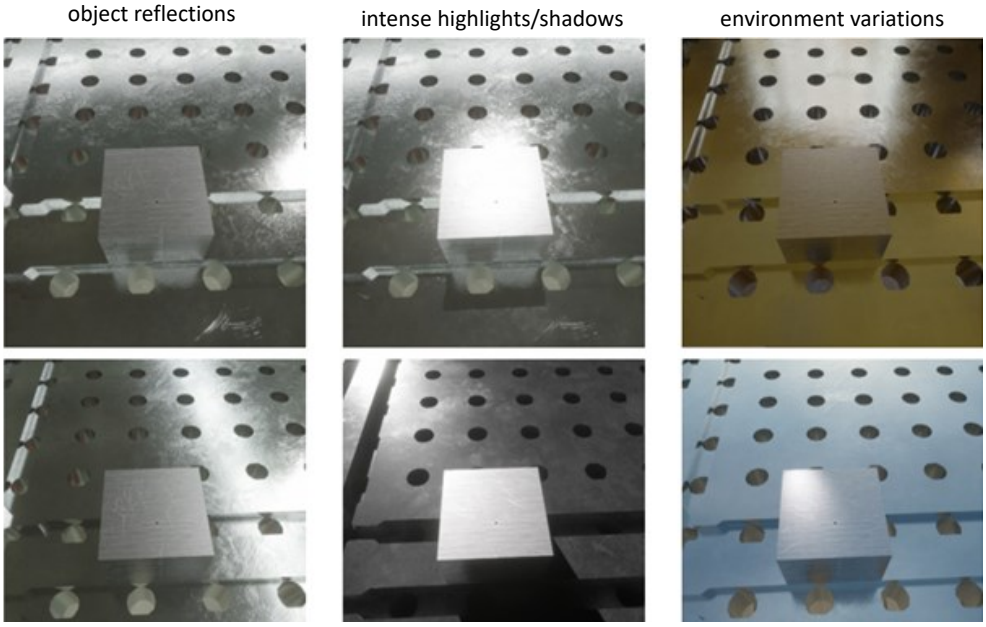

Figure 3: Complex lighting variations. Left: inter-reflections between object and and pallet. Middle: intense highlights (top) and hard shadows (bottom). Right: changes in environment lighting.

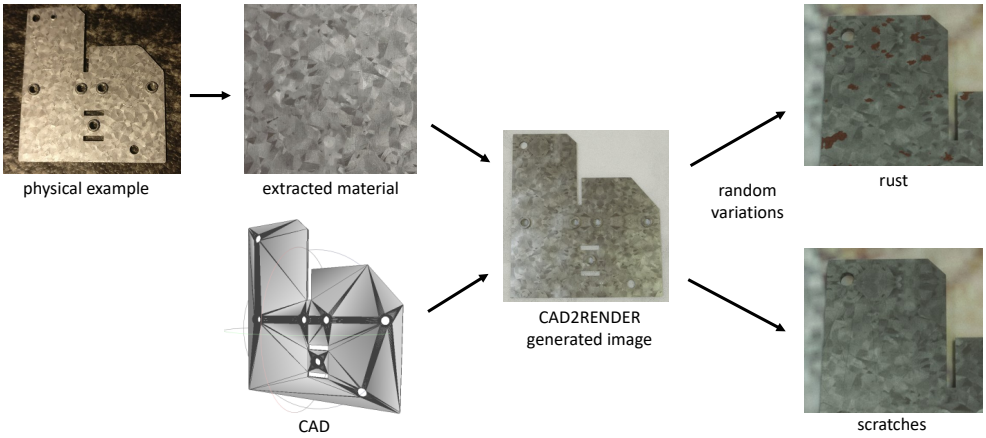

Figure 4: Example of rust and scratch variations. An extracted path material from a physical example (top left) is used in combination with a CAD file (bottom left) to generate synthetic images (middle). On top of the extracted material, we can add different material effects such as rust and scratches (right).

offsets, rust, polishing lines, scratches, etc. The database of materials can be provided by the user or can be based on online sources such as the Measured Material Library for Unity HDRP [16].

The simulation of rust and scratches is inspired by the work of Mihaylov [17]. FIGURE 4 shows examples of rust and scratch variations. In this example, we start from a basic material texture, extracted from a real physical example. The extracted material textures are adjusted to include imperfections such as rust or scratches. These variations are generated during the execution with the help of Simplex noise [18]. The noise map is used to mark areas where the imperfections need to be generated. To allow for more variations in material texture, we have implemented a texture resampling algorithm, based on the work of Opara et al. [19]. The goal is to use an input texture and to generate a new texture that looks similar but has some distinct variations. The proposed approach rearranges the pixels of the input texture and tries to limit any obvious seams. The advantage of this technique is that any type of variation can be modeled as long as an example texture is available. Because the technique of Opara et al. [19], is designed for offline rendering it is too slow (in the order of minutes) for synthetic data generation. To improve generation time, we have optimized the algorithm for GPU, allowing it to run at interactive frame rates. FIGURE 5 illustrates the approach of the resampling algorithms. First random patches are copied from an example image (on the left) to the desired output texture. This will create obvious seams where two patches meet (middle part). Then a pixel based algorithm is ran recursively to, step by step, improve the resulting texture (on the right). Each pixel determines the difference in pixel colors from their current neighborhood compared to the neighborhood they had in the original texture. Then every pixel will calculate the neighborhood difference of multiple pixels suggested by other pixels in the neighborhood and change to the pixel with the lowest difference. This is done multiple times in a row, reducing the radius of the neighborhood with each iteration.

Miscellaneous

To conclude, there are some miscellaneous dataset settings that can be set, such as: image resolu-

tion, rendering profile (see Section 3.2), post processing profile for white balancing, tonemapping, gamma correction, camera exposure, settings for export, number of physics frames and render frames before export, etc. It is important to note that CAD2Render is built upon Unity and the flexibility of the underlying game engine allows the user to implement or optimize any additional requirements for the use case at hand. This will possibly allow it to be applicable to other domains than the manufacturing industry as well, as long as CAD models are available.

3.2 Rendering Profiles and GPU Acceleration

The quality and speed of the renders are highly customizable because CAD2Render is based on the High Definition Rendering Pipeline of Unity [14]. Unity has a built-in GPU accelerated path tracer that can be used for rendering when photorealism is important. The main drawback of using pathtracing for the generation of the images is the time it takes to render. For some applications a large number of images might be needed. Complementary research has experimented with the amount of CAD2Render images needed to train object detection models and has shown that a large amount of images are beneficial when no domain knowledge is used [20].

DLSS 2.0 and/or NVIDIA OptiX Denoiser/Intel Open Image Denoiser can be used to reduce the time it takes to generate a converged image. To further increase the generation speed, two other rendering modes can be used: rasterization or hybrid. Rasterization relies on the classic rendering pipeline and hybrid mode is rasterization with limited ray tracing support for shadows, reflections and ambient occlusion. Changing the rendering approach to rasterization or hybrid can have a considerable positive impact on generation time, but can introduce different types of artifacts. The

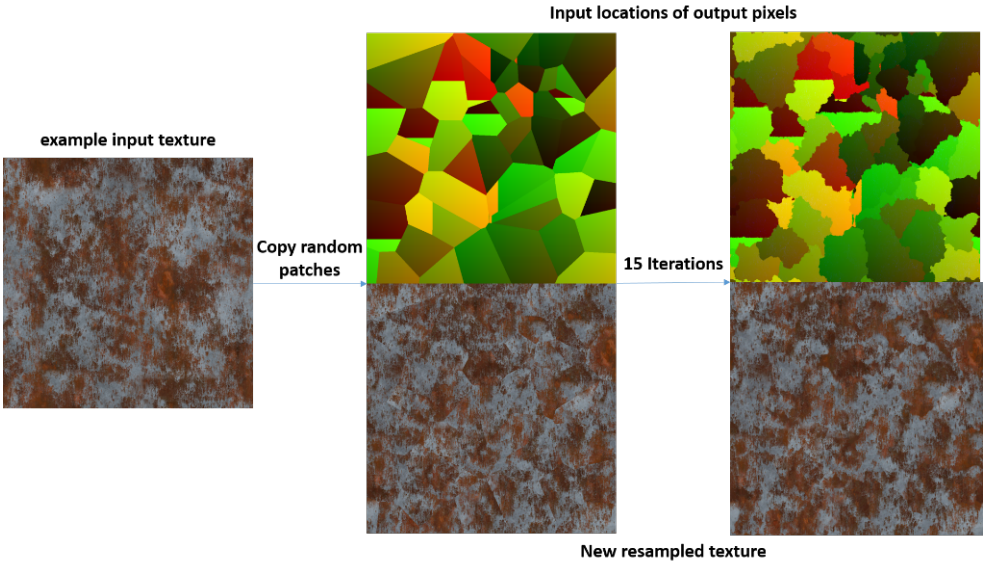

Figure 5: Texture resampling. Left: starting material texture. Middle: first iterations of copied patches. Some noticeable hard edges can be perceived. Right: refined result after 15 iterations.

hybrid renderer regularly fails to show all reflections, shadows and highlights that would be present in images generated with the path tracer. The path tracer can however introduce noise in areas where the path tracer results converge slowly. A denoiser can reduce this type of artifact.

TABLE 1 gives an overview of the rendering time compared to the resolution between path tracing (at 500 rays per pixel) and rasterization. All measurements were taken with the template scene of the CAD2Render repository on a RTX2070S GPU and an i7-10700KF CPU. Based on empirical observations, enabling the denoiser and rendering $1/10$ of the samples will introduce no noticeable artifacts or noise, allowing for an additional speedup factor of 10. However, further research is required to prove this does not impact the performance of the machine learning models.

3.3 Exporting to BOP Format

To allow for easy application of the generated datasets, CAD2Render exports the annotated dataset to the standardized BOP format [21]. This file format contains RGB images, object and camera poses, camera parameters, instance segmentation, depth maps and 3D models. The BOP format and its accompanying toolkit are originally designed for easy-of-use benchmarking for pose estimation, but due to the rich annotations it can be used for other tasks, such as object detection, segmentation and depth estimation.

3.4 Importing BOP Dataset for Digital Twin Creation

CAD2Render supports the import of existing BOP datasets as well. This is especially useful for creating a digital twin dataset of a real dataset. This feature makes it easier to research the domain gap between synthetic and real images. The Dataset of Industrial Metal Objects [22], is one such digital twin dataset, generated by CAD2Render.

3.5 Workflow

First the CAD models are loaded as prefabs into the CAD2Render Unity project. Then the parameters of the randomizers are set. The most common parameters are the camera, object, material and light parameters. Here the default parameters can be used but adding domain knowledge in the data

Table 1: Generation time and storage size for a dataset of 10.000 images. Comparison between various resolutions and render mode (path tracing compared to rasterization). Measured on a RTX2070 Super GPU and an i7-10700KF CPU at 3.80GHz.

Rendering 10.000 images					
Resolution	Path tracing 500 samples time (hours)		Rasterization time (hours)		Memory (GB)
	With DLSS 2.0	Without DLSS 2.0	with DLSS 2.0	without DLSS 2.0	
1280 x 720	7.9	7.8	2.3	2.3	5.4
1920 x 1080	8.3	13.2	2.4	2.4	11.3
3840 x 2160	22.3	30.0	3.4	3.6	43.8

generation can drastically reduce the amount of data that is required to train the AI-models [20]. For a detailed list of all possible parameters, we refer to the github page of CAD2Render. Here all parameters are explained in detail.
The generated data is then used to train the AI-models. For an increased performance in the resulting models it is required to evaluate any cases where the models fail and generate extra data for these edge cases. This workflow is illustrated in FIGURE 6.

4. VALIDATION

Validation of CAD2Render is done on two industrially relevant use cases: bin picking and 2D keypoint detection. The former requires solutions for both object detection and pose estimation. The latter is a useful approach to detect important landmarks. The validation results are trained solely on synthetic data and tested on real data. Bin picking is applied to metallic objects on a table. Keypoint detection is done for two use cases: tools in use in natural environments and assembly validation on a top down view of a work piece. This wide range of applications highlights the customizability of the toolkit.

4.1 Bin Picking of Metal Objects

The proposed method has been validated in a setup representing an industrial pick and place application, using a collaborative robot (cobot), equipped with a suction cup and a static camera. The identification of the objects and subsequent position and pose estimation was done with two state of the art networks: YoloV4 [23], for object detection and PVNET [24], for pose estimation. These models are applied sequentially on the input image, where the object detector will calculate the crop in which the object is found. This crop is then used as input for the pose estimator, calculating a full coordinate set for the object in the image. In the case where multiple items are found by the object detector, each crop is processed individually. FIGURE 7 shows a high-level overview of how both networks are used. The proposed method is highly relevant for industrial applications, as traditionally, detection of custom parts relies on the usage of a 3D camera that will generate a 3D point cloud. In this point cloud, the Iterative Closest Point algorithm is used [25], to match a CAD model to a shape in the point cloud. However, this technique fails when not

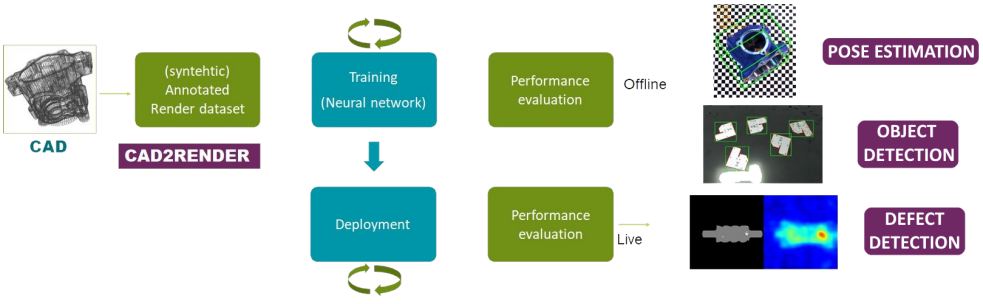

Figure 6: Overview of the workflow starting from the cad file towards the different end algorithms.

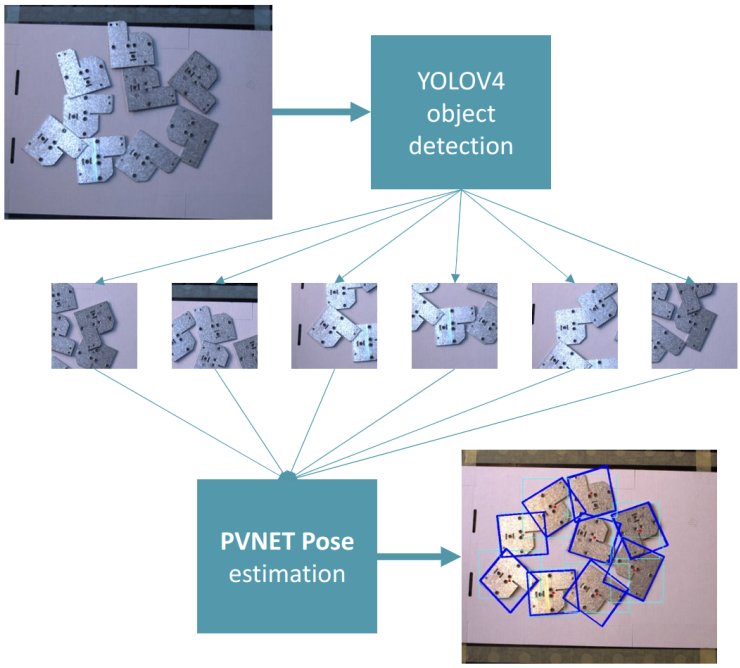

Figure 7: Overview of the flow of the validation algorithm

enough volumetric information is available. This is the case for the items presented in this chapter, as these are very thin (1.5mm) and are lost in the measurement noise. Another large benefit of the proposed technique is that hardware cost is significantly lower as only a simple 2D camera and lens is needed (the combination of which, in our setup, costs less than 800€), while industrial 3D cameras are significantly more expensive. Off course, in both scenarios there exists a need for an external computer with dedicated graphics card.

Dataset Description

A large dataset of 20.000 images was generated. The images contain from 1 to 10 identical items. The item used for this study is a small stamped metal piece, approximately 7 by 7 centimeter and not symmetrical. The rendered images feature random camera angles, The height variations where within 30cm of the real setup and the angles where randomized within 20°. The light was randomized with 20 different environment maps with randomized exposures. For the material properties we used photogrammetry to extract the albedo from images.

Validation Setup

Quantifying the accuracy of the combined pipeline has to be performed with the appropriate hardware considerations in mind. More precisely, an accurate calibration of the camera pinhole model [26], is vitally important. For this, a ChArUco board was used, which is a combination of a chequerboard and ArUco markers.This board is also used to perform the extrinsic calibration, where the relationship between camera pixels and the world geometry is established by calculating the transformation from the camera coordinate system to the world coordinate system. A reference point (or origin point) is established from which geometrical distances can be calculated.

The validation method then follows the following steps: (1) carefully place the item on the grid; (2) use the pose estimator to estimate the location of a corner of the item (loop for 10 times); (3) calculate geometrical position of this point with respect to the origin point; (4) calculate difference with known location. The camera used in this setup was an IDS UI-3280CP Rev. 2, which has a 2456x2054 pixel sensor and a global shutter.

Validation Results

The object was placed at various locations of the ChArUco board, generally at 30mm intervals. As no rotations of the object were performed, the validation is only valid for the position estimations and not for the rotation estimation. FIGURE 8 shows the result of the validation methodology explained in this section. It can be seen that there is a specific area in the image outside which the subsequent estimations of the keypoint show relatively large deviations.

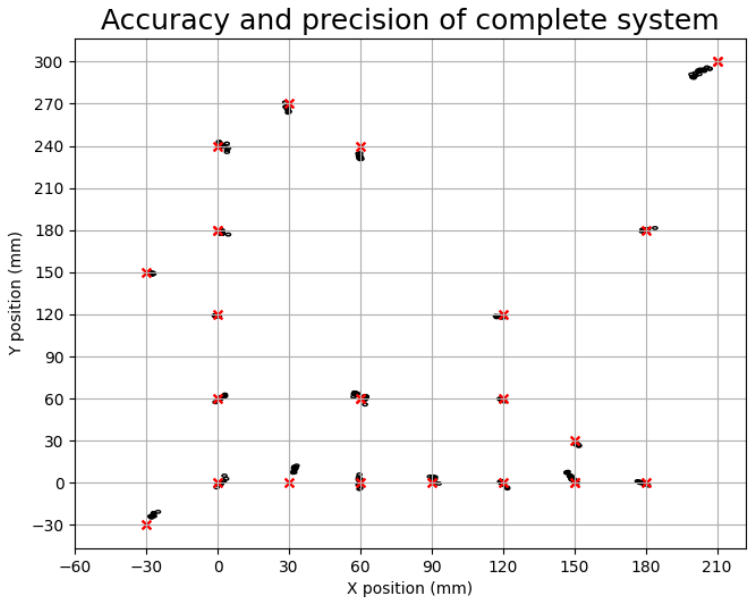

Figure 8: Result of the validation methodology. The red X represents the ground-truth, while the black dots are the results of subsequent estimations by the Pose Estimator.

Table 2: Validation Results of the position estimation for Metal Plates on the ChArUco board.

pos	Std Dev [mm]	Max dev [mm]	Min dev [mm]
x	0.91	7.50	0.040
y	1.37	9.92	0.002
z	11.16	57.90	0.220

TABLE 2 shows the statistical analysis of the combined results. It can be seen that the standard deviation in the Y-direction is slightly larger compared to the X-direction, which can be explained by the fact that the used camera sensor is not square and therefore there are more pixels in the Y-direction. This means that these pixels fall in the more heavily curved part of the lens, increasing the deviation. More generally, it is shown that, provided good camera calibration, the models can

be accurate to close to 1 mm in both X and Y directions, which will be more than good enough in most pick-and-place applications. The estimation in the Z-direction is large, however this is a classic issue with height estimations from 2D images that can be improved in the future with a multi-camera setup.

Robustness against harsh light conditions
To demonstrate the robustness of the developed algorithms against harsh lighting conditions, a simple physical setup was conceived on which an object at a known location and orientation could be subjected to either high or low lighting conditions. A dataset was rendered following the method explained in 3.1, this time using a slightly larger metallic object. A simple method of counting the saturated pixels on the surface of the object under test gave an approximate value of light intensity. From TABLE 3, we can see that under all but the most harsh conditions, the XY-deviation, calculated as the Euclidean distance between the ground-truth and the estimated location, is low. It can be observed that when the light intensity reaches more than 70%, the deviation reaches around 1cm, which can be considered to be extreme.

Table 3: Deviation of the estimated XY-location of an object under a wide range of light intensities. The intensity specifies the percentage of pixels are over- or under-saturated.

Intensity (%)	XY Deviation [cm]
0.000	0.276
35.440	0.340
44.090	0.742
69.990	0.565
73.170	1.460
83.660	2.444

4.2 Object Detection of Tools on a Workbench

Here we use an top-down camera to detect which tools are on a workbench. For the detection a YoloV4 network is trained on screwdrivers and two types of wrenches. During the training only synthetic images are used in combination with data augmentation. Then the trained networks are validated on the real images which are manually annotated. In the real images a wide variety of objects are present, including objects that where not in the training data.

Dataset Description
The training data is created using the CAD2Render tool. An example of the training images is provided in FIGURE 9. In these images multiple tools are rendered including ones that are not present in the validation images. A top down view with an angle of less then 20°compared to straight down. The material properties where created in combination with the CAD models by a 3D artist. A total of 50.000 images where created with an additional 10.000 images for fine tuning. The images for fine tuning where created by copying the rendered tools on top of a clean plate of the validation setup.

Validation Setup
For the validation a top down camera is placed a 2 meters above a workbench. Tools are placed in random positions with some random objects and boxes around that might be present in an industrial

Figure 9: Example images of the training data for tool type detection.

environment. Some extra variate is added by using a projector to display random patterns on the setup. Multiple images are taken of the same setup with different projector patterns. Afterwords the tools are then manually annotated.

Validation Results
The network we used for validating this use case was trained for 20.000 iteration with a 90-10% split in the train and test set. Afterwards another 20.000 training iterations are executed on the fine tuning dataset. The resulting network got a mean average precision of 83% at an Intersection over Union of 0.5. In FIGURE 10 the output of the network is given.

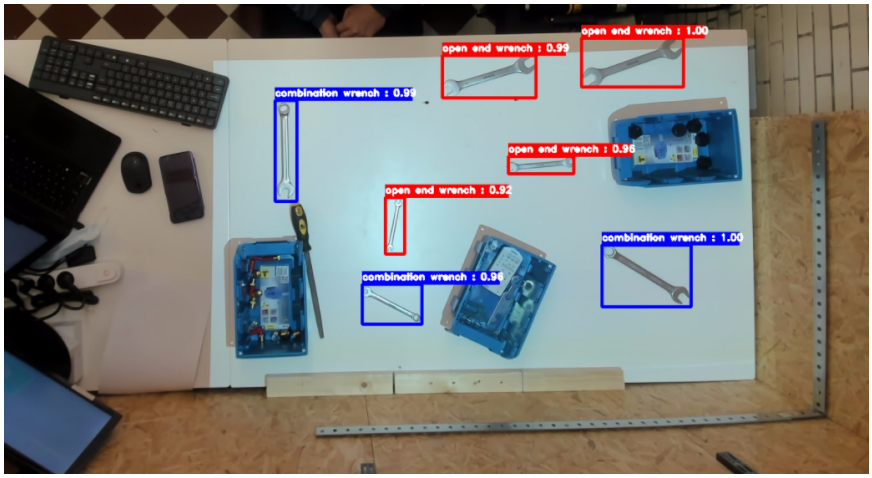

Figure 10: Result of the tool type detector on real images.

4.3 2D Keypoint Detection of handheld tools

As an additional validation case, CAD2Render was used to train models for the problem of semantic 2D keypoint detection. To find the landmarks, a UNET [27], type architecture with intermediate supervision was trained to generate probability maps for each semantic keypoint location. The problem of tool keypoint detection was investigated by Vanherle et al. [28], using the CAD2Render tool, for more details consult their paper. In this use-case we attempt to find the location of certain sementic keypoints of hand tools. The tools considered are a screwdriver, hammer, wrench and

combination wrench. For each of these tools we find a number of keypoints by training a model for each tool.

Dataset Description
To train such models, a large amount of data is needed. For each tool we collected a few textured 3D models from the internet. The 3D tools from the internet did not closely resemble the target tools, but did belong to the same class of tool. The CAD2Render toolkit was used to generate 20.000 images for each tool. The tools were randomly spawned in a space with a random environment map as background. Additionally, a few random objects from ShapeNet [29], were also spawned in the space to simulate occlusions. For this validation case, the faster hybrid renderer was used. FIGURE 11 shows some examples of synthetically generated images from the tool keypoint dataset.

Figure 11: A few examples from the tool keypoint detection dataset created by CAD2Render.

Validation Setup
To verify whether the datasets generated by CAD2Render are suitable to train a keypoint detection model on, we test the performance of the trained model on real images of tools. For each of the four tools we captured 50 real photographs, and manually annotated these images. To properly test the robustness of the trained models objects were photographed in wide variety of poses, lighting conditions, camera angles and backgrounds. Additionally, occlusions and truncations are introduced. The model trained on the synthetic data was then used to detect keypoint locations in the real images.

Validation Results
To measure the model's performance we use the Percentage of Correct Keypoints (PCK) [30], metric with an α of 1.0. The results are shown in TABLE 4. The models were trained on synthetic images, created randomly without taking domain knowledge into account. Yet, these models are able to detect keypoints in the unseen real images with good accuracy. This shows that the CAD2Render toolkit is able to create synthetic images that are suitable for this problem space and that images produced by the faster hybrid rendering mode can produce good models as well. Additionally, research has shown that models trained on images generated by CAD2Render perform better on this task than models trained on images generated by simple 2D image augmentations [31]. This shows the benefit of using 3D information to generate training data.

Table 4: Accuracy of the keypoint detection models for each tool trained on the synthetic data. Performance is measured in $PCK_{0.1}$ over the validation set of real images [28].

Tool	$PCK_{0.1}$
Screwdriver	86.1
Wrench	88.9
Combination Wrench	86.1
Hammer	84.4

5. LIMITATIONS AND FUTURE WORK

Although we performed extensive experiments showing the good performance of models trained on data generated by CAD2Render, a direct comparison to other methods is missing. Future work should compare the performance of models trained using our method to the performance of models trained on data generated by other state of the art methods. Additionally, parameters such as speed of rendering and ease of use should also be taken into account. Additionally, we would like to investigate the impact of the different quality rendering modes in CAD2render on the final model performance. A number of training data variations were introduced to help improve generalization. A study on the impact of these variations on downstream performance is beyond the scope of this paper. For an in depth analysis of the impact of light and pose variations on object detection performance, for datasets generated by the CAD2Render tool, we refer to complementary research [20].

6. CONCLUSION

This paper proposed a novel toolkit for synthetic data generation, that can generate a vast amount of complex photorealistic variations, including changes in lighting, appearance and pose. It is cost-effective and optimized for rendering speed on consumer hardware by exploiting the recent advancements in real-time raytracing and denoising, which is essential for fast deployment in low-volume and high-variance manufacturing. We showcase that data generated with CAD2Render can be used to train ai-models with high accuracy when performing object detection or pose estimation. Our toolkit is designed for industrial use cases but can be utilized in other domains if high quality 3D models are available. Since it allows for the import and export of datasets in a standardized fashion, it can generate synthetic simulations of existing datasets, a so called digital twin. As such, it can be an enabling technology for future research on sim2real and how to close the domain gap between the real and synthetic world. Future improvements would be to include more realistic variations in the appearance of the objects based on extracted appearance of real physical examples. This would possibly reduce the sim2real domain gap that still exists in the generated data.

7. ACKNOWLEDGEMENT

This research was realized in the framework of the PILS SBO project (Products Inspection by Little Supervision), funded by Flanders Make, the strategic research Centre for the Manufacturing

Industry in Belgium; and the Special Research Fund (BOF, mandate ID BOF20OWB24) of Hasselt University.

References

[1] Wong SC, Gatt A, Stamatescu V, McDonnell MD. Understanding Data Augmentation for Classification: When to Warp? In: International Conference on Digital Image Computing: Techniques and Applications (DICTA). 2016;2016:1-6.

[2] Xu Y, Jia R, Mou L, Li G, Yunchuan C, Lu Y et al. Improved Relation Classification by Deep Recurrent Neural Networks With Data Augmentation. Clin Orthop Relat Res. 2016. Arxiv Preprint: https://arxiv.org/pdf/1601.03651.pdf

[3] Perez L, Wang J. The Effectiveness of Data Augmentation in Image Classification Using Deep Learning. 2017. Arxiv Preprint: https://arxiv.org/pdf/1712.04621.pdf

[4] Movshovitz-Attias Y, Kanade T, Sheikh Y. How Useful Is Photo-Realistic Rendering for Visual Learning? In: Hua G, Jégou H, editors. Computer vision – ECCV. Cham. Springer International Publishing; 2016. p. 202-17.

[5] Shafaei A, Little JJ, Schmidt M. Play and Learn: Using Video Games to Train Computer Vision Models. In: Wilson RC, Hancock ER, Smith WAP, editors. Proceedings of the British machine vision conference. York, UK: BMVC, September 19-22, 2016. BMVA Press; 2016.

[6] Tobin J, Fong R, Ray A, Schneider J, Zaremba W, et al. Domain Randomization for Transferring Deep Neural Networks From Simulation to the Real World.2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada. 2017:23-30.

[7] `https://developer.nvidia.com/isaac-sim`.

[8] Greff K, Belletti F, Beyer L, Doersch C, Du Y, et al. A Scalable Dataset Generator. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA. 2022:3739-3751.

[9] Yoonwoo J, Seungjoo S, Junha L, Choy C, Anandkumar A, Cho M et al. Perfception: Perception Using Radiance Fields. 2022. Arxiv Preprint: https://arxiv.org/abs/2208.11537

[10] Crispell D, Biris O, Crosswhite N, Byrne J, Mundy JL. Dataset Augmentation for Pose and Lighting Invariant Face Recognition. arXiv preprint arXiv:1704.04326, 2017

[11] Lee K, Moloney D. Evaluation of Synthetic Data for Deep Learning Stereo Depth Algorithms on Embedded Platforms. In: 4th International Conference on Systems and Informatics (ICSAI). 2017; 2017;170-176

[12] Alhaija HA, Mustikovela SK, Mescheder L, Geiger A, Rother C. Augmented Reality Meets Computer Vision: Efficient Data Generation for Urban Driving Scenes. Int J Comput Vis. 2018;126:961-972

[13] Kaur P, Taghavi S, Tian Z, Shi W. A Survey on Simulators for Testing Self-Driving Cars. In: Fourth International Conference on Connected and Autonomous Driving (MetroCAD). 2021; 2021:62-70.

[14] https://unity.com/srp/High-Definition-Render-Pipeline.

[15] Tremblay J, Prakash A, Acuna D, Brophy M, Jampani V, et al. Training Deep Networks With Synthetic Data: Bridging the Reality Gap by Domain Randomization. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE Publications; 2018:1082-10828

[16] https://github.com/Unity-Technologies/MeasuredMaterialLibraryHDRP

[17] Mihaylov V. Rendering Metals and Worn or Weathered Metallic Objects. Master's thesis, Technical University of Denmark. 2013.

[18] https://www.researchgate.net/publication/281031452_Simplex_Noise_Demystified

[19] https://github.com/EmbarkStudios/texture-synthesis

[20] Vanherle B, Moonen S, Van Reeth F, Michiels N. Analysis of Training Object Detection Models With Synthetic Data. In: 33rd British Machine Vision Conference. London: BMVC. BMVA Press; 2022. Arxiv Preprint: https://arxiv.org/pdf/2211.16066.pdf

[21] Hodan T, Michel F, Brachmann E, Kehl W, GlentBuch A, et al. Bop: Benchmark for 6D Object Pose Estimation. In Proceedings of the European conference on computer vision (ECCV). 2018.

[22] De Roovere P, Moonen S, Michiels N, Wyffels F. Dataset of Industrial Metal Objects. 2022. Arxiv Preprint: https://arxiv.org/pdf/2208.04052.pdf

[23] Bochkovskiy A, Wang CY, Liao HYM. Yolov4: Optimal Speed and Accuracy of Object Detection. Clin Orthop Relat Res 2020. Arxiv Preprint: https://arxiv.org/pdf/2004.10934.pdf

[24] Peng S, Liu Y, Huang Q, Zhou X, Bao H. Pvnet: Pixel-Wise Voting Network for 6DOF Pose Estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019:4561-4570.

[25] Besl PJ, McKay ND. A Method for Registration of 3-D Shapes. IEEE Trans Pattern Anal Mach Intell. 1992;14:239-256.

[26] Kannala J, Brandt SS. A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses. IEEE Trans Pattern Anal Mach Intell. 2006;28:1335-1340.

[27] Ronneberger O, Fischer P, Brox T. U-net: Convolutional Networks for Biomedical Image Segmentation. Med Image Comput Comput Assist Interv MICCAI. 2015:234-241.

[28] Vanherle B, Put J, Michiels N, Van Reeth F. Detecting Tool Keypoints With Synthetic Training Data. In: Galambos P, Kayacan E, Madani K, editors. Robotics, computer vision and intelligent systems. Cham. Springer International Publishing; 2022;190-207.

[29] Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, Xiao J. Shapenet: An Information-Rich 3D Model Repository. 2015. arXiv preprint: https://arxiv.org/pdf/1512.03012.pdf

[30] Yang Y, Ramanan D. Articulated Human Detection With Flexible Mixtures of Parts. IEEE Trans Pattern Anal Mach Intell. 2013;35:2878-2890.

[31] Vanherle B, Put J, Michiels N, Van Reeth F. Real-Time Detection of 2D Tool Landmarks With Synthetic Training Data. In: Proceedings of the 2nd international conference on robotics, computer vision and intelligent systems – ROBOVIS. INSTICC, scite. Press; 2021:40-47.