# Increasing the Robustness of Deep Learning Models for Object Segmentation: A Framework for Blending Automatically Annotated Real and Synthetic Data

Artúr István Károly, *Member, IEEE*, Sebestyén Tirczka, Huijun Gao, *Fellow, IEEE*, Imre J. Rudas, *Life Fellow, IEEE*, and Péter Galambos, *Member, IEEE*

***Abstract*—Recent problems in robotics can sometimes only be tackled using machine learning technologies, particularly those that utilize deep learning (DL) with transfer learning. Transfer learning takes advantage of pretrained models, which are later fine-tuned using smaller task-specific datasets. The fine-tuned models must be robust against changes in environmental factors such as illumination since, often, there is no guarantee for them to be constant. Although synthetic data for pretraining has been shown to enhance DL model generalization, there is limited research on its application for fine-tuning. One limiting factor is that the generation and annotation of synthetic datasets can be cumbersome and impractical for the purpose of fine-tuning. To address this issue, we propose two methods for automatically generating annotated image datasets for object segmentation, one for real-world and another for synthetic images. We also introduce a novel domain adaptation approach called filling the reality gap (FTRG), which can blend elements from real-world and synthetic scenes in a single image to achieve domain adaptation. We demonstrate through experimentation on a representative robot application that FTRG outperforms other domain adaptation techniques, such as domain randomization or photorealistic synthetic images, in creating robust models. Furthermore, we evaluate the benefits of using synthetic data for fine-tuning in transfer learning and continual learning with experience replay using our proposed methods and FTRG. Our findings indicate that fine-tuning with synthetic data can produce superior results compared to solely using real-world data.**

***Index Terms*—Data annotation, deep learning (DL), digital twin, mixed reality, robotics, synthetic data.**

Manuscript received 11 December 2022; revised 7 April 2023; accepted 29 April 2023. This work was supported in part by the National Research, Development and Innovation Fund of Hungary, financed through the 2019-1.3.1-KK Funding Scheme under Project 2019-1.3.1-KK-2019-00007. The work of Péter Galambos was supported by the (Bolyai+) New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund under Grant UNKP-22-5. This article was recommended by Associate Editor H. Zhang. *(Corresponding author: Péter Galambos.)*

Artúr István Károly, Sebestyén Tirczka, Imre J. Rudas, and Péter Galambos are with the Antal Bejczy Center for Intelligent Robotics, Óbuda University, 1034 Budapest, Hungary (e-mail: artur.istvan.karoly@irob.uni.obuda.hu; sebestyen.tirczka@irob.uni.obuda.hu; imre.rudas@irob.uni.obuda.hu; peter.galambos@irob.uni.obuda.hu).

Huijun Gao is with the Research Institute of Intelligent Control and Systems, Harbin Institute of Technology, Harbin 150080, China (e-mail: hjgao@hit.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2023.3276485.

Digital Object Identifier 10.1109/TCYB.2023.3276485

## I. Introduction

MODERN robotic applications often need to function in dynamic environments, where certain aspects, such as illumination, clutter, and object occlusions, are constantly changing [1], [2]. Data-driven approaches, such as deep learning (DL), are commonly utilized in these scenarios. Robots can assess their environment based on visual data processing performed by DL models, providing them the required flexibility to operate under dynamic circumstances. The robustness of these models can be described as their capability to adapt/generalize to a range of settings with different environment factors [1]. Models which can generalize to a large variety of settings with different lighting, clutter, etc., conditions are considered more robust than models which are not. In this article, we inspect two ways to achieve good generalization. One is through the use of continual learning [2], [3], where generalization comes from accumulating knowledge over time in a dynamic environment while avoiding forgetting the previously acquired knowledge [3]. The other is through transfer learning [4], [5], which relies on training datasets that force the models trained on them to generalize well. We show that both approaches can benefit from using synthetic training data for increasing model robustness, with the help of representative experimental setups for object segmentation and image recognition for robotic tasks.

For both approaches, the training data has great significance as it directly influences the generalization capabilities (and thus the robustness) of the trained models. It has been shown that models trained on a dataset coming from a source domain adapt to a target domain better if the source and target domains are similar [5]. Variations in the data distribution of the training set can also help the trained model accommodate the difference in the data distributions between the source and target domains [6], [7]. This suggests that training datasets that contain samples from the same or a similar domain the models will be used in and which incorporate sufficient variations in the data distribution are likely to result in more robust trained models. However, data collection with a real-world robot setup can be highly time consuming and expensive due to the sheer amount of data needed to incorporate variations to obtain robust models [8]. Furthermore, since many DL-based

solutions use supervised learning techniques, training datasets often have to be annotated [9]. However, the manual annotation of the data is also very time consuming or practically impossible, especially in complex tasks, such as semantic scene segmentation or instance segmentation, which are often required in robotic applications. In addition, DL models are also known for their need for extensive training datasets. As a result, collecting and annotating the training data can be a massive roadblock to developing new DL-based solutions for robotics.

It is easy to see the great potential of using automated means for data collection/annotation and synthetic data for training DL models. Synthetic data can be generated rapidly and inexpensively with automated annotation, incorporating arbitrary variations in the data distribution. However, synthetic data for training DL models comes with a tradeoff by introducing a discrepancy between the domains from which the training data is generated and in which the models will be used. Consequently, models trained on synthetic datasets may not be able to adapt well to real-world data, even if they perform well on synthetic data. This phenomenon is known as the "reality gap." Overcoming this challenge (often referred to as "bridging the reality gap" [1], [6], [7]) is crucial for leveraging the benefits of synthetic datasets and ensuring that DL models trained on synthetic data are robust and generalize well to the real world.

There are two primary techniques for bridging the reality gap, regardless of the synthetic data generation approach. The first one is to create a synthetic dataset similar to a real-world dataset [10], [11], [12], [13], [14]. In the case of synthetic image data, this is achieved by creating photorealistic images. The other approach is domain randomization. This technique introduces unrealistic levels of variation in the synthetic dataset, which forces the models trained on such datasets to ignore the effects of the randomized factors and thus to generalize to the real-world data as well [6], [7], [13], [14]. Many approaches focus on what aspects and to what extent photorealism or domain randomization should be used and whether these two methods could be combined [13], [14], [15]. However, these approaches still try to bridge the reality gap, "building from only one side," by improving synthetic data generation. In this article, we propose a novel method, which we call filling the reality gap (FTRG), that takes advantage of an automated real-world dataset generation technique and a corresponding synthetic data generation pipeline and aims to overcome the reality gap by blending real-world and synthetic components inside images. With FTRG, we have complete control over which parts of an image come from the real-world data and which ones come from the corresponding synthetic images. Furthermore, real-world and synthetic parts can also be overlayed on top of each other by taking advantage of the alpha channel of the images. By continuously modifying the opacity, we can achieve a seamless transition between real-world and synthetic image components. In our experiments, we demonstrate how this approach can yield superior results when fine-tuning models for object segmentation over using datasets that contain images that are either completely synthetic or coming from the real-world dataset.

The benefits of using a synthetic dataset are often discussed from the aspect of transfer learning. It is well established that pretraining on a large synthetic dataset and then fine-tuning on real-world data can be superior to training only on real-world data [12], [16]. For example, Nowruzi et al. [16] explored the benefits of having a synthetic dataset in addition to real-world annotated data for object detection. In their experiments, they trained an SSD single-shot detector [17] with MobileNet as the backbone [18] from scratch on a dataset containing both synthetic and real-world data. They found that the additional synthetic data significantly reduced the need for training on real-world samples (10%, 5%, and 2.5% of the original real-world dataset was used). Additionally, when training the model from scratch, synthetic pretraining and fine-tuning on real-world data outperformed simply training the model on a mixed dataset (consisting of both real-world and synthetic samples).

Creating a synthetic dataset is often seen as an alternative when one lacks a large annotated real-world dataset for pretraining. This article focuses on the benefits of using synthetic data for fine-tuning when an automatic real-data labeling pipeline or an already annotated real dataset is available. We consider the typical case when a pretrained model is already available, and only the fine-tuning step has to be completed. We show how synthetic data can improve the fine-tuning steps of transfer learning approaches for robotic manipulation and the training of continuous learning methods using experience replay for image classification.

The contributions of this article include the following.

1) A real-data labeling procedure for robotic manipulation, which can automatically generate instance segmentation masks for images of real-world scenes.
2) A Blender-based annotation tool which can automatically generate instance segmentation masks for rendered synthetic images.
3) Experiments showing how synthetic rendered data can help boost the performance of continual learning methods using experience replay.
4) A novel way of creating a mixed reality dataset combining our automatic annotation techniques. This method is called "FTRG" as it combines synthetic and real-world data seamlessly.
5) Experimental results showing the benefits of using synthetic data in the fine-tuning step of transfer learning in instance segmentation, showcasing the potential of the FTRG method.

## II. Related Work

Novel techniques employing synthetic data have shown encouraging results in tasks that were previously deemed challenging, including transparent object detection, robotic cloth folding, and object rearrangement using visual data [19], [20], [21], [22]. One particularly successful approach, Dex-Net 2.0, leverages synthetic point cloud data and analytic grasp metrics to train a grasp quality convolutional neural network (GQCNN) for robotic grasp prediction [23].

Liu et al. [24] proposed a synthetic dataset generation pipeline for robotic picking with a vacuum gripper based

on RGB-D data. They utilized Blender, an open-source 3-D computer graphics software [25], for generating photorealistic synthetic images and a learning-based (GAN) approach for overcoming the reality gap. They evaluated their approach against the popular Dex-Net 3.0 benchmark [26] and reported its performance on a real-world robotic picking setup. Their results revealed that the proposed approach can deal with multiobject picking and challenging novel objects (transparent, or small objects) robustly. Many recent scientific results utilize similar GAN-based image-to-image domain adaptation approaches to overcome the reality gap [27], [28], [29]. Contrary to these solutions, FTRG takes advantage of real-world images directly rather than relying on a learning-based approach to extract features that transfer well to the real-world domain.

Similarly to the solution of Liu et al., Denninger et al. [30] also utilized Blender for their synthetic dataset generator, BlenderProc. BlenderProc is entirely Blender-based and can generate photorealistic rendered images and corresponding segmentation, depth data, surface normals, etc. The potential of BlenderProc is demonstrated in indoor scene segmentation examples, but it could also be adapted to robotic tasks.

Greff et al. [31] introduced Kubric, a synthetic dataset generator utilizing PyBullet [32] and Blender. It is an extremely versatile framework that can generate photorealistic renders and various annotation types, such as segmentation masks, pose, optical flow, surface normals, etc. Several models trained using synthetic data generated from Kubric have been shown to achieve outstanding results in various fields, such as point tracking, semantic segmentation, salient object detection, pose estimation, etc. [31], which proves its effectiveness and flexibility.

Our synthetic data generation pipeline also utilizes Blender to create photorealistic renders and corresponding annotations. However, instead of creating an extensive framework, such as Kubric or BlenderProc, we aimed to keep our solution lightweight and implemented it as a Blender addon. This allows it to be used with no additional installation steps (only Blender is needed) and allows the pipeline to be used for potentially limitless scenarios (not limited to robotics solutions).

### A. Photorealistic Synthetic Data

Dvornik et al. [33] proposed the so-called copy–paste method for augmenting image datasets for object detection. Their approach utilizes object segmentation annotations to crop object instances from a set of real-world images and paste these cropped instances into other real-world images. They achieved significantly better model performance by utilizing the copy–paste strategy if the instances were placed considering the visual context. They used a CNN to model different object categories' visual context and used this network to guide the instance placement.

Li et al. [34] also utilized the copy–paste method. They proposed a sim-to-real framework for training object recognition and localization DL models for industrial robotic bin picking. They used photorealistic synthetic images, corresponding depth data, object segmentation masks, and 6-D object poses to train their models. They found that even though their generated synthetic data looked photorealistic, it still lacked the fine details that can be observed in real-world images, such as uneven illumination, object deformations, etc. As a result, they proposed a semi-synthetic dataset inspired by the copy–paste method. They manually cropped instances from real-world images and pasted them in real-captured backgrounds (hence the visual context was correct without the need for a context modeling technique). They showed that models trained on a mixed dataset containing images from their photorealistic synthetic and semi-synthetic datasets outperformed models trained using either only synthetic or semi-synthetic images. They showed that DL models trained with their framework could be directly applied to real-world samples without fine-tuning them on real-world data and achieved superior performance compared to state-of-the-art methods.

A similar approach can be seen in the winning solutions at the Amazon Robotics Challenge (ARC) 2017 [35], [36]. They both utilized semi-automated methods to generate segmentation masks for images of novel objects from multiple views, which were later used to construct a semi-synthetic dataset on which their models could be fine-tuned quickly.

FTRG takes this concept further by combining synthetic and real-world images. Instead of pasting the cropped real-world object instances on top of real-world backgrounds, with FTRG, we take advantage of a real-world and multiple corresponding synthetic images with the same camera–object alignments and camera setups but different textures and lighting. This allows us to mix and match different parts in a single image (e.g., real-world and synthetic objects in a single image with synthetic background). Instead of simply cropping and pasting different parts in the images, FTRG makes use of opacity, with which we can continuously control to what extent a certain part of an image is synthetic or real, thus creating a seamless transition between synthetic and real images.

Schwarz and Behnke [37] proposed Stillleben, a synthetic dataset generation pipeline for training DL models used for perception tasks in robotics, such as semantic segmentation, object detection, and pose estimation. Contrary to the copy–paste and similar approaches which facilitate 2-D synthesis, Stillleben utilizes a synthetic 3-D scene to generate high-quality rendered images for known objects with corresponding segmentation masks, depth data, surface normals, etc., in an online fashion. The authors emphasize that Stilleben can be used online, enabling its usage in life-long/continual learning. However, this property limits the quality of the rendered images. In order to overcome this challenge, Benedikt et al. utilized the GAN-based CUT approach (from Park et al. [38]), an image-to-image translation model, to adapt the synthetic images from Stilleben to the real-world domain [27]. They trained the RefineNet semantic segmentation model (proposed by Lin et al. [39]) from scratch. They showed that the model trained with the dataset utilizing their domain adaptation approach achieved higher intersection over union (IoU) scores with narrower distribution when compared to the model trained on images directly from Stillleben.

Our synthetic data generation pipeline also utilizes a 3-D synthetic scene. However, while Stillleben uses OpenGL to

generate rendered images quickly, our method utilizes Blender Cycles, a photorealistic renderer. This means that our approach can only generate offline synthetic datasets, but in return, it can provide better-quality rendered images. In our experiments, we demonstrate how using an offline-generated synthetic dataset can still be advantageous in the context of continual learning.

Martinez-Gonzalez et al. [10] and Garcia-Garcia et al. [11] introduced a photorealistic synthetic data generator for indoor semantic scene segmentation and robot manipulation using unreal engine and virtual reality (VR) technologies. The DL models trained on their data showed promising qualitative results in monocular depth estimation, 6-D pose estimation for synthetic samples, and 6-D pose estimation [10]. Their results suggest that VR technologies can help bring realism into synthetic scenes, such as realistic robot interactions. In the FTRG method, we aim to leverage the same principle. However, instead of "borrowing" real human actions to replace the synthetic robot interactions, we borrow parts of a real-world scene to replace parts of our synthetic scene in a mixed-reality setting.

Roberts et al. [12] created Hypersim, a photorealistic synthetic dataset for indoor scene understanding. They emphasize the use of publicly available 3-D assets. According to their research, most synthetic datasets only provide rendered images rather than 3-D assets, limiting their use cases' flexibility and potential for future development. They also highlight that their annotation pipeline is not tied to the rendering process, which makes it possible to generate or change the annotations of a scene without rerendering it. Our method for generating annotations for synthetic data is decoupled from the scene preparation and the rendering. It allows the approach to be used for any scene, not limited to robotic manipulation, and to change the annotations without rerendering. For our scenes, we only used publicly available free 3-D assets. In comparison, the preparation of Hypersim came with a cost of $57K, of which $6K was used to purchase the required 3-D assets, although their scenes are of much higher quality and the number of 3-D assets they used is much higher than ours as well.

Roberts et al. also conducted experiments to show the sim-to-real performance of models trained on Hypersim. Their experiments revealed that Hypersim pretraining could improve the semantic segmentation performance on NYUv2 [40] compared to pretraining on PBRS [41]. However, it did not improve performance compared to pretraining on SceneNet RGB-D [42]. They attribute these results to the fact that PBRS contains an order of magnitude more samples than Hypersim, while SceneNet RGB-D contains two orders of magnitude more samples. They imply that the reason why Hypersim is still able to yield comparable results to these datasets is because of its better photorealism. They also suggest that there could be a tradeoff between photorealism and the amount of data needed to achieve good sim-to-real performance. This means that datasets with less but more photorealistic data could be on par with more extensive but less photorealistic datasets. This is in accord with the findings of Huh et al. [43], who investigated what makes the ImageNet dataset [44] suitable for transfer learning. They showed that increasing the number of classes or the amount of pretraining data beyond a certain point did not bring significant benefits for transfer learning. These results indicate that an additional, small amount of well-chosen/good-quality data can improve transferability more than simply increasing the size of the dataset. Even though these findings are for the pretraining phase of transfer learning, they are also promising for the fine-tuning phase since the generalization capability should be preserved as much as possible during fine-tuning. In contrast, the size of the fine-tuning set should be kept as small as possible. In response to this, in our synthetic dataset generation pipeline, we aim to create high-quality photorealistic rendered images.

### B. Domain Randomization

Domain randomization introduces unrealistic levels of variation in the synthetic dataset, which forces the models trained on such datasets to ignore the randomized factors' effects and thus generalize to the real data as well [6], [7], [13], [14]. Nonphotorealistic synthetic images were shown to be useful for training object segmentation models for robotics in recent approaches [45], [46].

Tobin et al. [7] showed that a DL model trained on a large quantity of low-fidelity rendered images could be successfully deployed in a real-world scenario. They randomized camera and object positions and lighting conditions while they used unrealistic randomized environment textures. Their experiments demonstrated that a DL model, which was exclusively trained on their domain randomized synthetic data, was able to detect simple geometric objects in a real scene. Furthermore, the detections were also accurate and reliable enough to be used in a robotic grasping pipeline.

Tremblay et al. [6] presented that the domain randomization technique can also be used for bridging the reality gap in more complex scenarios. They showed that DL models for vehicle detection on the real KITTI dataset [47], which were trained on their domain randomized data, could compete in performance with other models trained on the Virtual KITTI dataset [48]. In our experiments, we utilize domain randomization by assigning unrealistic textures to the objects in the synthetic scene.

Seeing the success of photorealism and domain randomization, a logical question arises: Can they be combined to get the best of both worlds? Tremblay et al. [13] proposed combining the two approaches for bridging the reality gap. They used photorealistic synthetic images in combination with domain-randomized ones. They showed that DL models exclusively trained on such a synthetic dataset could achieve state-of-the-art performance in robotic manipulation. Our experiments also inspect the effects of combining domain randomized and photorealistic data for fine-tuning.

Eversberg and Lambrecht [14] examined whether and to what extent it is worthwhile to implement photorealism versus domain randomization techniques in a synthetic dataset for an industrial object detection task. They used Blender to generate the synthetic dataset. Their experimental results suggest that domain randomization techniques outperformed higher realism for the background and clutter objects (which are not related to the object of interest). For the object of

interest, realistic textures and realistic lighting affecting the object resulted in better performance than domain randomization techniques. Similar to their solution, our synthetic data annotation pipeline also uses Blender, and we build upon their findings to determine how the blending of real-world and synthetic data should be done in our FTRG approach.

Prakash et al. [15] introduced structured domain randomization. They aimed to generate domain-randomized synthetic data but keep the structural context of the environment. For example, a conventional domain randomized image dataset for vehicle detection would place the vehicles, the camera, and other environmental objects in the scene according to a random distribution. In contrast, in a structured domain randomization dataset, the vehicles are placed on roads, so the structural context of the environment is preserved. They compared photorealistic approaches, such as the Virtual KITTI and the GTA V dataset [49], a domain randomized dataset [6], and their structured domain randomization approach. Their experiments showed that models trained on a dataset with structured domain randomization could outperform models trained on photorealistic or domain-randomized synthetic data. They suggest the advantage of structured domain randomization comes from the trained model having a better understanding of the scene context than models trained on conventional domain randomized data while also being exposed to a similarly large variation in the data distribution. Our FTRG method can be considered as a variation of structured domain randomization, where the environmental context is given by the arrangement of objects in the actual scene, and the randomization is applied to the texture and lighting of the objects.

Instead of trying to preserve environmental context, a more effective approach might be to do the inverse and identify and randomize the environmental factors that affect a DL model's performance, as proposed by She et al. [2] for continual learning in robotics.

### C. Continual Learning

Continual learning techniques aim to overcome the challenge of an ever-changing environment by continuously updating the prediction model as new data becomes available [3]. A great challenge for continual learning techniques is catastrophic forgetting. It happens when already acquired knowledge is lost (forgotten) due to training the model on new observations. There are multiple approaches to overcome catastrophic forgetting, such as introducing a regularization term, enforcing architectural changes in the network structure, or keeping a working memory of previous training data for experience replay during training [3]. In our experiments, we inspect the experience replay approach.

She et al. [2] aimed to identify the factors influencing the prediction accuracy of continual learning algorithms for robotics. They created a benchmark dataset, OpenLORIS Object, which explicitly contains quantitative information on environmental factors, such as lighting level, object pixel size, clutter, and occlusions. They used the train–test accuracy matrix to evaluate the trained models, from which they derived metrics, such as backward transfer (BWT) and forward transfer (FWT). BWT characterizes how well the model can solve previously seen tasks. This is a measure that continual learning techniques need to consider to avoid catastrophic forgetting. On the other hand, the FWT characterizes how well a model performs on yet-unseen future tasks. This is closely related to model robustness. Their results suggest that one reason for the poor performance of continual learning approaches is their struggle with transferring knowledge to new tasks and scenes. This is most apparent in the FWT measure since BWT is usually improved due to techniques that try to avoid catastrophic forgetting. We hypothesize that a small number of synthetic data could improve the FWT of certain continual learning approaches. The findings in identifying the environmental factors that affect DL performance are accounted for in our solutions, where we explicitly include randomization of said factors in our synthetic datasets.

Using synthetic data in continual learning is not unheard of. Synthetic data can be used during their evaluation process [50], meanwhile, generative models are often used to enrich available data based on past experiences and thus prevent catastrophic forgetting [51], [52]. However, to the best of our knowledge, a combination where synthetic data with implicit domain knowledge is used for experience replay instead of a generative model is yet to be seen. Such a system can automatically generate synthetic scenarios ahead of time. For example, suppose lighting conditions are expected to change during the operation, but the model did not encounter samples with different lighting yet. In that case, synthetic samples can be generated with different lighting conditions and added to the experiences encountered by the model. Our experiments show how such a system could improve the FWT of continual learning with experience replay and its effect on BWT and overall accuracy.

## III. Methods and Techniques

### A. Real Data Annotation Pipeline

Our real-data annotation pipeline can generate instance segmentation masks for real-world images automatically. We maintain a virtual counterpart of the significant elements of the real scene, such as the camera and the objects. The virtual scene is a digital twin of the real world, so the virtual camera and objects reflect the pose of their real counterparts. This setup enables us to compute the segmentation masks for the objects in the virtual scene and then associate these annotations with corresponding images taken by the camera in the actual scene.

The generation of instance segmentation masks relies on computing the perspective projection of 3-D points on the objects' surfaces onto the image plane. We use the formalism as described in [53], according to which, the perspective projection $\bar{\mathbf{x}} = (u, v, 1)^\top$ of a 3-D point (given in the world frame) ${}^w\mathbf{X} = ({}^wX, {}^wY, {}^wZ, 1)^\top$ is described as

$$\bar{\mathbf{x}} = \mathbf{K}\Pi\ {}^c\mathbf{T}_w\ {}^w\mathbf{X} \tag{1}$$

where $\mathbf{K}$ is the matrix containing the intrinsic parameters of the camera. These parameters can be determined by camera calibration. Our solution used the OpenCV library [54] for the camera calibration, with a printed A4-sized checkerboard

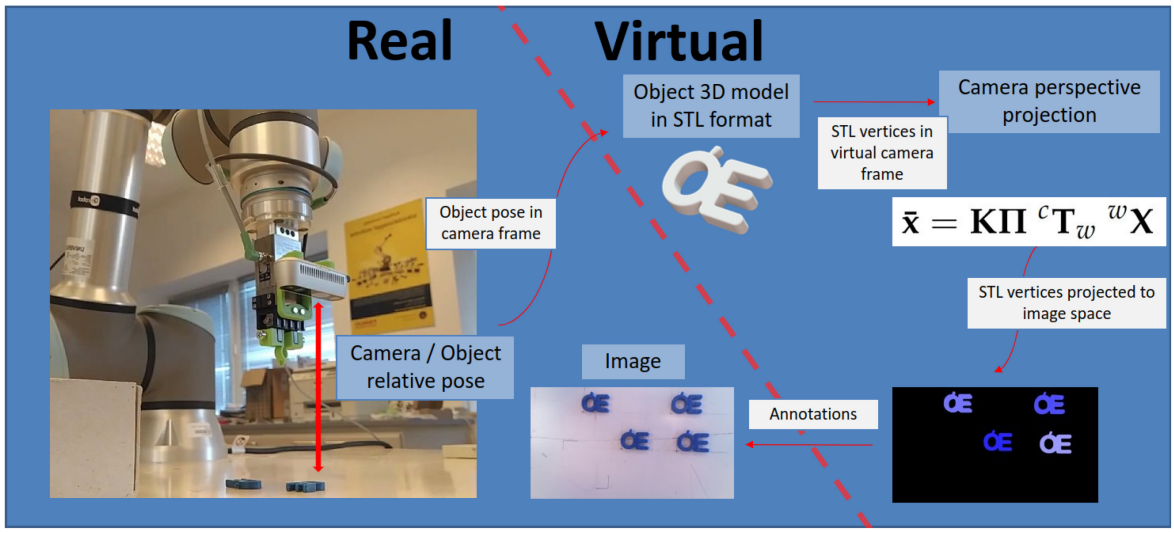



Fig. 1. Real data annotation setup.

pattern. Without the need for additional calibration steps, we were able to achieve sub-10-pixel average accuracy (in 1920×1080 resolution images) for projecting 3-D points onto the image plane, which resulted in human-level annotations. $\Pi$ is the projection matrix, which has the form of $[\mathbf{I}|\mathbf{0}]$, where $\mathbf{I}$ is a $3\times3$ identity matrix and $\mathbf{0}$ is a column vector of 3 zeros. ${}^{c}\mathbf{T}_{w}$ is the $4\times4$ homogeneous transformation matrix describing the transformation between the world and the camera frame.

Let $P({}^{w}\mathbf{X})$ denote the perspective projection of the point ${}^{w}\mathbf{X}$, $F = \{{}^{w}\mathbf{X}_1, {}^{w}\mathbf{X}_2, \ldots, {}^{w}\mathbf{X}_n\}$ a set of points to form a face, $R({}^{w}\mathbf{X})$ a ray coming from the origin of the camera frame and going through the point ${}^{w}\mathbf{X}$, and ${}^{w}\mathbf{X}^{\mathcal{O}}_{all}$ all the possible points on the surface of the object $\mathcal{O}$. For creating segmentation masks, each object has to be associated with a selected, finite set of points on their surface: ${}^{w}\mathbf{X}^{\mathcal{O}} = \{{}^{w}\mathbf{X} | {}^{w}\mathbf{X}$ on the surface of $\mathcal{O}\}$, ${}^{w}\mathbf{X}^{\mathcal{O}} \subseteq {}^{w}\mathbf{X}^{\mathcal{O}}_{all}$. The power set $\mathbb{P}({}^{w}\mathbf{X}^{\mathcal{O}})$ contains all the possible (not necessarily meaningful) faces for object $\mathcal{O}$, for a given set of surface points ${}^{w}\mathbf{X}^{\mathcal{O}}$. Polygons can be formed in the image plane by projecting each point of a face: $Poly^{F} = \{P({}^{w}\mathbf{X}_i)$ for ${}^{w}\mathbf{X}_i \in F\}$, and using the projections as the vertices of the polygon. A set of faces $F^{\mathcal{O}} \subseteq \mathbb{P}({}^{w}\mathbf{X}^{\mathcal{O}})$ have to be chosen for object $\mathcal{O}$, such that all the projections given by $P({}^{w}\mathbf{X}_j)$ for ${}^{w}\mathbf{X}_j \in {}^{w}\mathbf{X}^{\mathcal{O}}_{all}$ fall inside at least one polygon of $Poly^{F_k}$, for $F_k \in F^{\mathcal{O}}$, but projections $P({}^{w}\mathbf{X})$, where $R({}^{w}\mathbf{X})$ does not intersect the object, do not fall into any of the polygons from $Poly^{F_k}$, for $F_k \in F^{\mathcal{O}}$.

In the case of a simple cube, for example, the selected set of surface points should be the vertices (${}^{w}\mathbf{X}^{\mathcal{O}} =$ vertices), while the selected faces would naturally be the set of six faces of the cube ($F^{\mathcal{O}} =$ faces). When projecting the points of the faces, one would get six tetragons in the image plane. It is easy to see that for any point on the cube's surface, the projection would fall into at least one of these tetragons. Any other point which is not on the surface of the cube and also not between the camera and the cube or behind the cube (so the ray from the origin of the camera frame going through the point does not intersect the object) would get projected outside of all the tetragons. Thus, merging all the tetragons into a single polygon gives us the segmentation mask for the cube.

However, for objects with complex shapes, the manual selection of surface points and the manual definition of faces is not feasible. Luckily, a very common virtual representation for 3-D object models is the standard triangle language (STL) format. This representation defines a 3-D surface model of the object, given by an object mesh consisting of triangles formed by vertices. As a result, we can directly use the STL style representation of an object by defining ${}^{w}\mathbf{X}^{\mathcal{O}} = {}^{w}\mathbf{T}_{o}\ {}^{o}\mathbf{X}^{\mathcal{O}}$, where ${}^{o}\mathbf{X}^{\mathcal{O}}$ are the vertices in the STL format (they are defined in the object frame), and ${}^{w}\mathbf{T}_{o}$ is the $4\times4$ homogeneous transformation matrix describing the transformation between the object and world frames. $F^{\mathcal{O}}$ can be selected according to the triangles in the STL representation. In common robotics scenarios, it is usually assumed that a 3-D model of the objects is available or can be easily created. A photogrammetry application can also be used to acquire a 3-D mesh as such methods are becoming increasingly accessible with a mobile phone [55]. In our synthetic dataset generation pipeline, we utilized Qlone [56] for scanning clutter objects. Using the STL representation of the object mesh also has the advantage that one only needs to measure the pose of an object relative to the world frame, from which ${}^{w}\mathbf{T}_{o}$ can be determined, instead of measuring every individual point relative to the world frame ${}^{w}\mathbf{X}^{\mathcal{O}}$.

The transformation matrix ${}^{c}\mathbf{T}_{w}$ from (1) needs to be measured before the annotation procedure. In our setup, we utilize an industrial robot and attach the camera in a known, fixed pose to the robot's end effector. As a result, the transformation between the camera frame and the robot's tool center point (${}^{c}\mathbf{T}_{\text{TCP}}$) will be fixed regardless of the robot's pose. The transformation between the robot TCP and the world frame (${}^{\text{TCP}}\mathbf{T}_{w}$) can be obtained from the robot controller at all times. Thus, the transformation between the camera and the world frame can be written as ${}^{c}\mathbf{T}_{w} = {}^{c}\mathbf{T}_{\text{TCP}}\ {}^{\text{TCP}}\mathbf{T}_{w}$.

Fig. 1 shows our setup for the automatic annotation of real data. The preliminary steps needed for the annotations are as follows.

1) Acquiring object meshes in STL format (${}^{o}\mathbf{X}^{\mathcal{O}}$ for all objects).
2) Camera calibration (determining $\mathbf{K}$).
3) Attaching the camera to the robot (measuring ${}^{c}\mathbf{T}_{\text{TCP}}$).
4) Placing the objects in known poses (measuring ${}^{w}\mathbf{T}_{o}$).

After the preliminary steps, the data collection and automated annotation can be carried out. We move the camera with the robot to a set of pregenerated target poses scattered in a grid pattern on the surface of concentric spheres centered around the scene. At each pose, the robot stops, and the camera takes an image of the scene. The current pose of the robot TCP relative to the world frame is attached to the image as metadata. Based on this pose, ${}^{c}\mathbf{T}_{w}$ can be determined, and the object masks can be computed.

However, generating the segmentation masks for each object individually is not enough. Since multiple objects are present in the scene simultaneously, occlusions also have to be considered. In most cases, a simplistic approach, such as generating the segmentation masks for the objects in the order based on their distance from the camera and allowing these segmentations to overwrite each other, would be enough. On the other hand, such an approach would not be able to handle complex occlusions where objects could interlock. We propose an algorithmic solution that deals with such occlusions (Algorithm 1).

As seen from Algorithm 1, each object gets a unique color ID (RGB values). The algorithm starts with an empty segmentation mask $\mathbf{M}$, with five channels. The first three channels are used for color (RGB representation), while the other two channels store the information about which triangle of which object the given pixel belongs to. The algorithm goes through each object in order and projects the vertices of the triangles in the object's mesh. We use $\mathcal{O}$ and $\mathcal{T}$ to represent the object and triangle which are being projected. After projecting the vertices of a triangle $\mathcal{T}$, we check which pixels fall inside of the projection of $\mathcal{T}$. Then, for each of these pixels, a test is performed, which can have three possible outcomes. If the color of the pixel is black, it can be colored with the color ID of $\mathcal{O}$. If the color of the pixel is the same as the color ID of $\mathcal{O}$, it means that the pixel was already marked as one belonging to $\mathcal{O}$, so nothing needs to be done. If the pixel was already colored but with the color ID of a different object $\tilde{\mathcal{O}}$, it has to be decided which color the pixel should have. For this purpose, we can look up the triangle $\tilde{\mathcal{T}}$ of the object $\tilde{\mathcal{O}}$, based on the last two channels of the mask and call the *IsOccluded* function on the two triangles $\mathcal{T}$ and $\tilde{\mathcal{T}}$. This function returns true if $\mathcal{T}$ is occluded by $\tilde{\mathcal{T}}$, in which case nothing should be done. Otherwise, the pixel can be colored with the color ID of the object $\mathcal{O}$. The *IsOccluded* function determines whether there is an occlusion by first checking trivialities (all vertices of one triangle being closer to the camera than the other's) and, in nontrivial cases, using the *SignedVolume* function. The complete definition of the *IsOccluded* and *SignedVolume* functions is in Algorithms 2 and 3, respectively, in the Appendix.

This method has the limitations that occlusions with intersecting objects and occlusions involving objects with significant size differences (one having much larger triangles in their object mesh compared to the other) may not be handled properly. These situations, however, can be considered marginal concerning a system for automatically generating segmentation-type annotations.

**Algorithm 1:** Generate Masks With Occlusion

**input :** Image shape: $[w, h, 3]$, List of objects: $\mathbf{O} = [\mathcal{O}_1, \mathcal{O}_2, \dots]$

```
/* Init annotation as black image                    */
Init: M = zeros((w, h, 5));
for O ∈ O do
    for T ∈ O.triangles do
        /* Projection as in (1)                      */
        v1^i, v2^i, v3^i = Project(T.vertices);
        temp_img = zeros((w, h));
        /* Get internal pixels of the
           triangle                                  */
        P = Where(DrawTriangle(temp_img, (v1^i, v2^i, v3^i),
         color=1) == 1);
        for p ∈ P do
            if M[p][0 : 3] == [0, 0, 0] then
                /* It was background before
                   */
                M[p][0 : 3] = O.color_id;
                M[p][3] = O.id;
                M[p][4] = T.id;
            else if M[p][0 : 3] == O.color_id then
                /* It is the same object         */
                Pass;
            else
                T̃ = O.GetTriangle(M[p][3], M[p][4]);
                if IsOccluded(T, by = T̃) then
                    /* T̃ occludes T              */
                    Pass;
                else
                    M[p][0 : 3] = O.color_id;
                    M[p][3] = O.id;
                    M[p][4] = T.id;
```

Fig. 2 shows some example images demonstrating the segmentation masks generated by our automatic real-data annotation pipeline. The images belong to an instance segmentation dataset with different numbers and types of objects, clutter, and illumination. We use this dataset to train our baseline model and test all the models in our experiments on the benefits of using synthetic data for fine-tuning.

### B. Synthetic Datasets and Automatic Annotation

Our synthetic data generation pipeline uses the open-source 3-D computer graphics suite Blender. For our experiments, we created two synthetic scenes. One is a tabletop environment containing objects available at our laboratory for testing the FTRG method. We refer to this scene as the OE scene. The other replicates a scene from the OpenLORIS

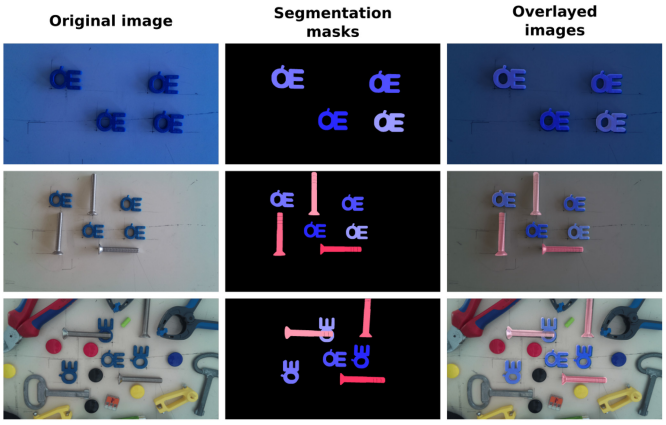



Fig. 2. Examples of automatically annotated real images.

Object dataset [2] for comparing continual learning model performance with and without synthetic data for experience replay. This scene is the synthetic counterpart of one of the real scenes in the OpenLORIS Object dataset, so we refer to it as SynLORIS. Both scenes were prepared with the scope of a fine-tuning dataset in mind. Although the prepared synthetic datasets demonstrate specific tasks, they are representative in the sense, that without additional complexity in the data generation pipeline, similar results could be achieved in other similar kinds of tasks as well.

*1) OE Scene:* The OE scene has two types of objects of interest, the 3-D printed "OE" logos (which the scene was named after) and standard DIN EN ISO 10642 M8x55 type bolts (later referred to as bolts). The scene also contains clutter objects of clamps and pliers, which were photo-scanned in our laboratory using the Qlone photogrammetry application. The scene's background is a planar surface serving as the tabletop. The arrangement of the OE logos, the bolts, the clutter objects, as well as the distance of the camera from the background plane were randomized.

We created photorealistic and randomized textures for our objects and the background and used realistic lighting. For the photorealistic shading of the tabletop, we used an image texture, the clutter objects used the textures acquired from the photo-scanning, and the OE logos and bolts used shaders created using Blender. The randomized textures were created to be unrealistic for the purpose of domain randomization.

From this scene, we generated 2500 images (OE synthetic dataset). Table I details the dataset composition. It contains images generated using five different settings, each with a training and validation split (400 and 100 images). The images from the first, second, and fourth settings utilize photorealistic textures, while those from the third and fifth settings use randomized textures. This means 60% of the images in the dataset use photorealistic and 40% use randomized textures. The images generated from the first three settings only show the OE logos, while the ones from the last two settings also contain bolts and clutter objects. Apart from images from the first setting, all other images were created with randomized lighting conditions, which make up 80% of the images in the dataset (light intensity and color were randomized, but the values were kept within realistic bounds). The number, position, and orientation of the OE logos, the bolt and the clutter objects, as well as the camera's distance from the table, were randomized for each image, resulting in a unique arrangement of the scene.

TABLE I
COMPOSITION OF THE OE SYNTHETIC DATASET

| Image ID | training (t) validation (v) | textures | objects | lighting |
|---|---|---|---|---|
| 0-399 | t | photoreal. | OE | static |
| 400-499 | v | photoreal. | OE | static |
| 500-899 | t | photoreal. | OE | rand. |
| 900-999 | v | photoreal. | OE | rand. |
| 1000-1399 | t | rand. | OE | rand. |
| 1400-1499 | v | rand. | OE | rand. |
| 1500-1899 | t | photoreal. | OE, bolt | rand. |
| 1900-1999 | v | photoreal. | OE, bolt | rand. |
| 2000-2399 | t | rand. | OE, bolt | rand. |
| 2400-2499 | v | rand. | OE, bolt | rand. |

Fig. 3. Example rendered frames from the OE synthetic dataset.

Some example rendered frames of the OE synthetic dataset can be seen in Fig. 3. It displays rendered images with either photorealistic or randomized textures for the OE logos, bolts clutter objects and the background, different lighting conditions, and randomized object and camera placement.

*2) SynLORIS:* The SynLORIS scene was modeled after a real scene from the OpenLORIS Object dataset. We attempted to reconstruct a similar environment by replicating the placement of the desk and background elements, as well as the direction of the lighting. We also aimed to use 3-D assets that resemble the real objects and textures, but we limited our selection to freely available 3-D assets from BlenderKit's library. For this scene, we did not aim to create very high-quality photorealistic renders and the perfect recreation of the objects. As for the OpenLORIS Object benchmark, the images are resized to $50 \times 50$ pixels, so the fine details would have been lost anyway.

In the SynLORIS scene, we introduced variation in two factors mentioned in [2]: 1) illumination and 2) object pixel size. There are three sources of illumination in the scene: 1) an HDRI; 2) an area light representing the light coming in through the window; and 3) a point light source above the table. In order to change the illumination, both the power of the lamps and the strength of the lighting from the HDRI were

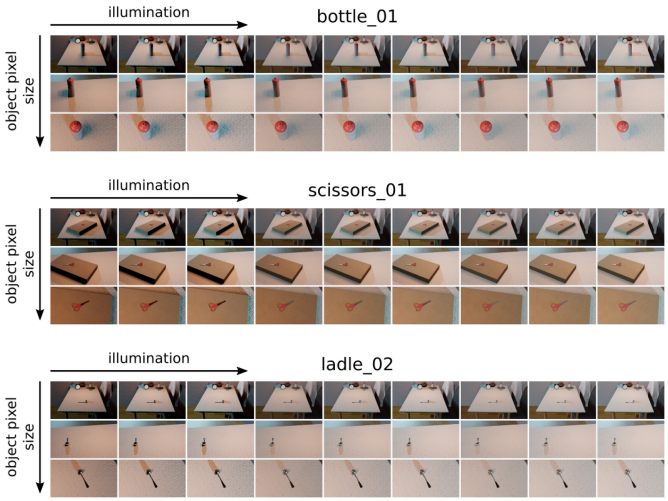



Fig. 4. Synthetic images rendered from the SynLORIS scene.

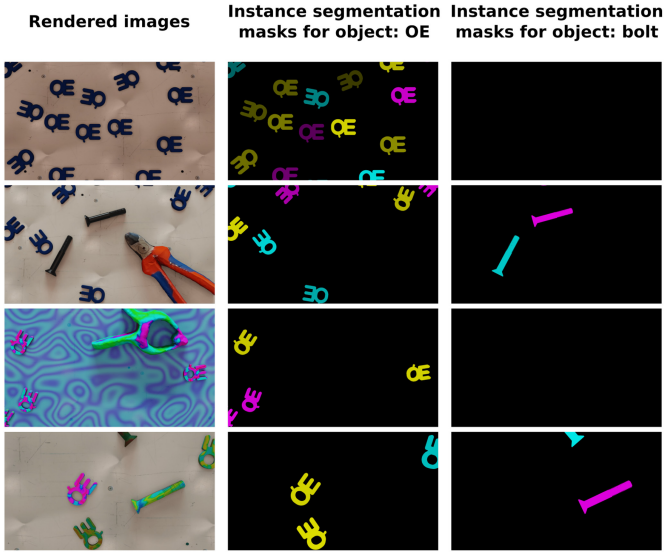



Fig. 5. Example BAT annotations for the OE synthetic dataset.

modified. The variations in the object pixel size were emulated by moving the camera closer and further from the object along a manually defined 3-D spline.

Similarly to the single-factor experiments in OpenLORIS Object, we also created nine tasks for the illumination factor, each with a different illumination level. For each task, we considered seven different objects (a small subset of the objects in the OpenLORIS Object dataset, which all use the same scene). We rendered 30 synthetic images for each object in each task. The camera path was the same regardless of the object or the task. In total, the generated dataset contains 1890 rendered images (9 tasks, 7 objects per task, and 30 images per object). Fig. 4 shows some of the rendered frames. We use the SynLORIS dataset in our experiments to show how an image classification model using experience replay can benefit from synthetic data.

*3) Blender Annotation Tool:* For the generation of the segmentation-type annotations in our OE synthetic dataset, we created a blender annotation tool (BAT), a Blender addon.[1] BAT can be used via a simple user interface (referred to as BAT panel), located in its own tab of the "n-panel" of the 3-D Viewport. The class for the background is added by default with black color. The BAT panel can be used to create, delete, or rename classes, change the color ID of a class or the collection of objects associated with it, and toggle whether the object collection should be treated as a collection of instances or not.

BAT uses the viewport renderer OpenGL to generate the segmentation masks. As a result, BAT only works if a GUI of Blender is open. It cannot run in background mode. However, the generation of the annotations takes significantly less time and resources (two orders of magnitude in our experience), and the rendering of the synthetic images can be completely decoupled from the generation of the annotations. This allows us to render the dataset on a powerful headless server while, in the meantime, generating the corresponding annotations on a separate system with limited resource usage. A few examples of the generated annotations can be seen in Fig. 5.

[1]https://github.com/ABC-iRobotics/blender_annotation_tool

TABLE II
TRAIN–TEST ACCURACY MATRIX $R$ FROM [2]; $Tr$ REPRESENTS TRAINING DATA, $Te$ REPRESENTS TESTING DATA, $R_{i,j}$ IS THE ACCURACY OF THE MODEL TRAINED ON $Tr_i$ AND EVALUATED ON $Te_j$, AND $N$ IS THE NUMBER OF TASKS

| R | $Te_1$ | $Te_2$ | ... | $Te_N$ |
|---|---|---|---|---|
| $Tr_1$ | $R_{11}$ | $R_{12}$ | ... | $R_{1N}$ |
| $Tr_2$ | $R_{21}$ | $R_{22}$ | ... | $R_{2N}$ |
| ... | ... | ... | ... | ... |
| $Tr_N$ | $R_{N1}$ | $R_{N2}$ | ... | $R_{NN}$ |

## C. Continual Learning Experiments

We conducted our experiments on continual learning using the OpenLORIS Object benchmark by She et al. [2]. Their paper introduced an evaluation method for continual learning techniques based on the train–test accuracy matrix, which can be described by Table II. They introduced the metrics FWT which is the average accuracy calculated for the upper triangle of the train–test accuracy matrix (marked in blue in the table), and BWT, which is the average accuracy for the lower triangle of the train–test accuracy matrix (marked in red in the table). BWT characterizes how well a model remembers previous tasks, while FWT describes how well a model can adjust to new tasks after training on the preceding ones.

The results of She et al. showed that FWT is the most challenging to maximize for many continual learning approaches. In our interpretation, FWT measures how well the trained model can generalize to new tasks. Our experiments aim to test our hypothesis that synthetic data, in the form of rendered images, can help improve the FWT of certain continual learning models. For this purpose, we use continual learning with experience replay [57] and evaluate it on the single-factor benchmarks introduced by She et al.

For our experiments, we use a selected set of seven objects from the OpenLORIS Object dataset (bottle_01, bowl_01,

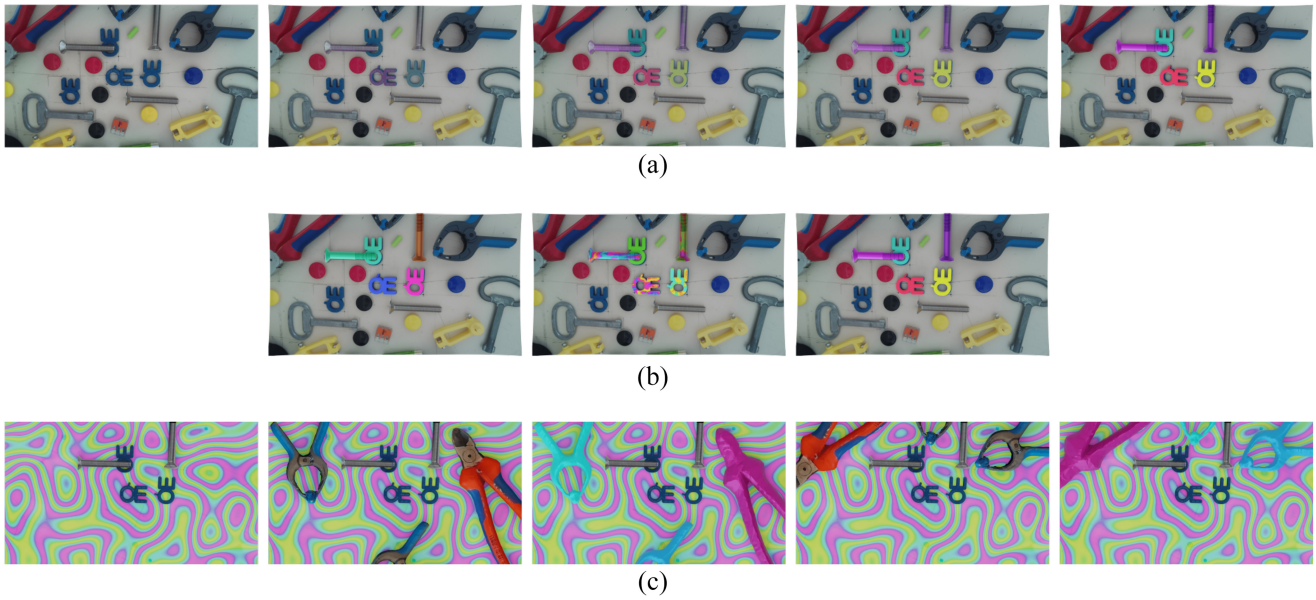


Fig. 6. Samples from our FTRG dataset. (a) Seamless transition from real to synthetic textures on a selected subset of objects. (b) Random texture for a selected subset of objects of interest with real background and clutter. (c) Real objects in a synthetic scene with synthetic clutter.

cup_02, cup_04, ladle_02, paper_cutter_04, and scisssors_01) as these are the objects for which we generated our synthetic dataset SynLORIS as well. There are four factors in total: 1) illumination; 2) occlusion; 3) clutter; and 4) object pixel size. For each factor, there are nine tasks with different levels of the corresponding factor. We evaluate two models for each factor. One of the models is only trained with data from the original OpenLORIS Object training set, while the other is trained with data from both the OpenLORIS Object and the corresponding SynLORIS data. Both models were given the same memory budget of 2000 and were trained for 100 iterations on each task. We use the validation set of the OpenLORIS Object dataset for the corresponding seven objects for testing across all factors and tasks. We evaluate each model for all four factors according to the same metrics, which were used by She et al. We report our findings in Section IV-A.

### D. FTRG Method

FTRG combines our automatic real-data segmentation pipeline with our synthetic data generation and annotation method to create rendered counterparts to real-world images. This allows us to blend different elements of the real and synthetic scenes (such as the objects of interest, the background, or clutter objects) in a single image. By controlling opacity, we can influence how much the virtual scene is blended with the real one, creating a seamless transition between reality and the synthetic environment. According to its purpose, the name of our method is FTRG.

FTRG starts with creating a real dataset with our automatic real-data annotation pipeline. Then, a synthetic counterpart of the scene is built in Blender, and based on the images from the real dataset, the camera pose, its motion, and its internal parameters are determined by the motion tracking module of Blender. After rendering, the real and synthetic images can be seamlessly blended using Blender's compositor workspace by layering them on top of each other and continuously adjusting the opacity of different parts of these layers based on the automatically generated segmentation masks. For our experiments, we created an FTRG dataset by combining different elements of the real and synthetic images (e.g., real background with synthetic objects and synthetic background with real objects). The labels for the FTRG dataset are "inherited" either from the real or the synthetic scene (using BAT). Fig. 6 shows possible ways to blend synthetic and real data in the FTRG dataset, showcasing the seamless transition between real-world and synthetic image components.

In order to show the effectiveness of this method, we compare the detection results of Mask-RCNN [58] networks which were fine-tuned using the FTRG method, and others which were using photorealistic synthetic data, domain randomized synthetic data, or real images for fine-tuning.

The FTRG method is currently limited by the automated real-world dataset collection approach because it requires that the pose of each object in the real-world scene is known. In simple tabletop scenarios, this can be ensured, but in a more complex setting, such as robotic bin-picking, the exact object poses are very hard to determine. In the future, an improved real-world data collection pipeline and advanced camera tracking could solve this issue.

## IV. Results

### A. Continual Learning Experiments

In our continual learning experiments, we compare the performance of models using experience replay on the OpenLORIS Object benchmark's single-factor experiments. One of the models only had access to formerly seen samples for experience replay, while the other had access to

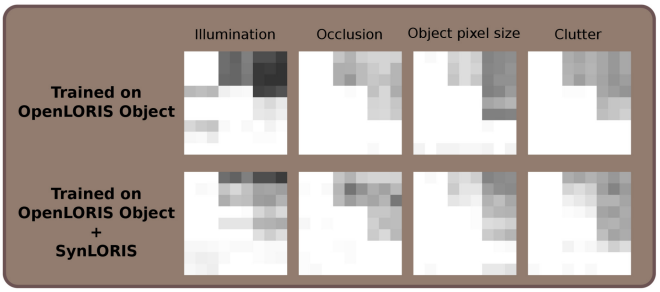



Fig. 7. Train–test accuracy matrices of image classification models, using experience replay and data from the OpenLORIS Object and the SynLORIS datasets; as evaluated by the OpenLORIS Object benchmark on all four factors (brighter color means greater accuracy).

TABLE III
QUANTITATIVE RESULTS FROM OUR CONTINUAL LEARNING EXPERIMENTS. VALUES PER CELL FROM TOP TO BOTTOM: ACCURACY, BWT, FWT, AND OVERALL ACCURACY (AS DESCRIBED IN [2]))

| | Illumination | Occlusion | Object pixel size | Clutter |
|---|---|---|---|---|
| m1 | 0.954 | 0.999 | 0.991 | 1.0 |
| | 0.945 | 0.999 | 0.991 | 1.0 |
| | 0.539 | 0.782 | 0.683 | 0.704 |
| | 0.77 | 0.903 | 0.854 | 0.868 |
| m2 | 0.979 | 0.996 | 0.991 | 0.993 |
| | 0.979 | 0.995 | 0.993 | 0.994 |
| | **0.685** | 0.74 | **0.768** | 0.716 |
| | 0.848 | 0.882 | 0.892 | 0.869 |

samples from the SynLORIS dataset. Fig. 7 visualizes the train–test accuracy matrices of the models for all four factors for qualitative assessment. The train–test accuracy matrices are represented as images, where the intensity of a pixel reflects the value of an element of the matrix, with 0 being black and 1 being white.

It is important to mention that the SynLORIS dataset only includes variations in two factors: 1) illumination and 2) object pixel size. The effects of this are visible in Fig. 7, where significant differences in the train–test accuracy matrices can only be observed in the illumination and object pixel size factors. In these cases, the model which had access to synthetic samples outperformed the model which was only trained on real samples. Table III shows our continual learning experiments' quantitative results, using the metrics She et al. introduced in [2]. The first row (m1) shows the results of training models exclusively on the OpenLORIS Object dataset. The second row (m2) shows the performance of models which were trained on data from both the OpenLORIS Object and the SynLORIS datasets. Notice how the models trained on real and synthetic samples had significantly better FWT for the illumination and object pixel size factors than models that only used real data.

### B. Synthetic Data for Fine-Tuning and FTRG

Our experiments demonstrate the effects of synthetic data for the fine-tuning phase of training a deep neural network. We also highlight the benefit of using the FTRG method compared to using only photorealistic synthetic images and/or domain randomization.

TABLE IV
PERFORMANCE OF MODELS (MAP @ IOU≥ 0.5) EVALUATED ON FIVE RANDOMLY SELECTED SUBSETS OF OUR TESTING SET

| | subset1 | subset2 | subset3 | subset4 | subset5 |
|---|---|---|---|---|---|
| *MRCNN-R* | 0.8889 | 0.8983 | 0.9029 | 0.9077 | 0.8674 |
| *MRCNN-P* | 0.8763 | 0.8599 | 0.8505 | 0.8296 | 0.8395 |
| *MRCNN-DR* | 0.8654 | 0.8307 | 0.8437 | 0.8132 | 0.7949 |
| *MRCNN-DR-R* | 0.9955 | 0.9817 | 0.9831 | 0.9774 | 0.9651 |
| *MRCNN-DR-P-R* | 0.9984 | 0.9887 | 0.9891 | **0.9812** | 0.9709 |
| *MRCNN-FTRG* | **1.0** | **0.99** | **0.9895** | 0.98 | **0.9715** |

We trained multiple Mask-RCNN [59] models for instance segmentation, using different datasets and evaluated them on the same test dataset.[2] All the models had the same network architecture and were initialized with the same pretrained weights (pretrained on the COCO dataset [60]). We used the same hyperparameters and a fixed number of training steps to fine-tune all the models. We trained five models in total and named each after the type of data used for their training.

1) *MRCNN-R:* This model was fine-tuned using only real data. The training dataset was annotated by our automated real dataset annotation method.
2) *MRCNN-P:* This model was fine-tuned using only photorealistic samples from the OE synthetic dataset.
3) *MRCNN-DR:* This model was fine-tuned using only synthetic images with unrealistic textures from the OE synthetic dataset.
4) *MRCNN-DR-R:* This model was fine-tuned using both domain-randomized synthetic samples from the OE synthetic dataset and samples from the real dataset.
5) *MRCNN-DR-P-R:* This model was fine-tuned using synthetic samples from the OE synthetic dataset (both photorealistic and domain randomized) and real samples.
6) *MRCNN-FTRG:* This model was fine-tuned using the FTRG dataset, which combines the real dataset with the domain-randomized synthetic dataset using the FTRG method.

For the test data, we collected real images using a variety of scenes (different backgrounds and arrangements of objects), clutter (level of clutter as well as types of clutter objects), and illumination conditions. This test set was also annotated with our automated real data annotation method. In Table IV, we report the mean average precision (mAP) at IoU greater than or equal to 0.5 for all five models over five randomly selected subsets of the test dataset.

The results show that models fine-tuned exclusively on synthetic data performed worse than the model trained exclusively on real samples. However, models trained on datasets consisting of both synthetic and real samples could significantly outperform the model trained using only real data. This suggests that using synthetic data in the fine-tuning phase of training deep neural networks can yield superior performance compared to training only on real data.

[2]Mask-RCNN implementation from: https://github.com/matterport/Mask_RCNN.

**Algorithm 2:** IsOccluded Function

```
input : Triangle: 𝒯, Other triangle: 𝒯̃
def IsOccluded(𝒯, by = 𝒯̃):
    if all(𝒯̃.vertices.z < 𝒯.vertices.z) then
        /* 𝒯̃ is closer to the camera */
        return: True
    else if all(𝒯.vertices.z < 𝒯̃.vertices.z) then
        /* 𝒯 is closer to the camera */
        return: False
    else
        /* There is an overlap in z */
        occluded = False;
        orig = [0,0,0];
        for ṽ1, ṽ2 ∈ Pairs(𝒯̃.vertices) do
            for v1, v2 ∈ Pairs(𝒯.vertices) do
                /* Check if the sides of 𝒯̃
                   intersect the 𝒯-orig
                   tetrahedron */
                if sign(SignedVolume(ṽ1, v1, v2, orig)) ==
                 sign(SignedVolume(ṽ2, v1, v2, orig)) then
                    Pass;
                else if sign(SignedVolume(ṽ1, ṽ2, v1, v2)) ==
                 sign(SignedVolume(ṽ1, ṽ2, v2, orig)) ==
                 sign(SignedVolume(ṽ1, ṽ2, orig, v1)) then
                    occluded = True;
                    break;
                else
                    Pass;
            if occluded == True then
                break;
        return: occluded
```

**Algorithm 3:** SignedVolume Function

```
input : 3D points: a, b, c, d
def SignedVolume(a, b, c, d):
    return: dot(cross(b − a, c − a), d − a)
```

The model trained on our FTRG dataset could outperform all other models in four out of five cases. We believe the slight improvement compared to the *MRCNN-DR-P-R* model is because of the fact that our FTRG dataset contains samples with blended real and synthetic components, which helps the *MRCNN-FTRG* model adapt to different real-life domains easier.

## V. Conclusion

Our results suggest that not only pretraining but fine-tuning can also benefit from synthetic data for increasing the robustness of models for object segmentation and image recognition in robotic tasks, especially if quick and easy-to-use automated methods are available for annotation and dataset generation. This statement also extends to methodologies that utilize a more dynamic weight-update strategy instead of transfer learning, such as continual learning with experience replay.

We showed that a model trained on t.he FTRG dataset could outperform models that only use entirely real or synthetic samples. As a result, investigating the effect of using the FTRG method for pretraining is an interesting direction for future research.

## Appendix

See Algorithms 2 and 3.

## Acknowledgment

The code, datasets, and results are available at https://github.com/ABC-iRobotics/Automatic-Dataset-Generation. Péter Galambos is a Bolyai Fellow of the Hungarian Academy of Sciences.

## References

[1] N. Sünderhauf et al., "The limits and potentials of deep learning for robotics," *Int. J. Robot. Res.*, vol. 37, nos. 4–5, pp. 405–420, 2018.

[2] Q. She et al., "OpenLORIS-object: A robotic vision dataset and benchmark for lifelong deep learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 4767–4773.

[3] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Netw.*, vol. 113, pp. 54–71, May 2019.

[4] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[5] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–14.

[6] J. Tremblay et al., "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 969–977.

[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2017, pp. 23–30.

[8] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," 2016, *arXiv:1603.02199*.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[10] P. Martinez-Gonzalez, S. Oprea, A. Garcia-Garcia, A. Jover-Alvarez, S. Orts-Escolano, and J. Garcia-Rodriguez, "UnrealROX: An extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation," *Virtual Real.*, vol. 24, no. 2, pp. 271–288, 2020.

[11] A. Garcia-Garcia et al., "The RobotriX: An extremely photorealistic and very-large-scale indoor dataset of sequences with robot trajectories and interactions," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2018, pp. 6790–6797.

[12] M. Roberts et al., "Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 10912–10922.

[13] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," 2018, *arXiv:1809.10790*.

[14] L. Eversberg and J. Lambrecht, "Generating images with physics-based rendering for an industrial object detection task: Realism versus domain randomization," *Sensors*, vol. 21, no. 23, p. 7901, 2021.

[15] A. Prakash et al., "Structured domain randomization: Bridging the reality gap by context-aware synthetic data," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 7249–7255.

[16] F. E. Nowruzi, P. Kapoor, D. Kolhatkar, F. A. Hassanat, R. Laganiere, and J. Rebut, "How much real data do we actually need: Analyzing object detection performance using synthetic and real data," 2019, *arXiv:1907.07061*.

[17] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[18] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.

[19] M. Byambaa, G. Koutaki, and L. Choimaa, "6D pose estimation of transparent objects using synthetic data," in *Proc. 28th Int. Workshop IW-FCV Front. Comput. Vis.*, 2022, pp. 3–17.

[20] T. Kollar, M. Laskey, K. Stone, B. Thananjeyan, and M. Tjersland, "SimNet: Enabling robust unknown object manipulation from pure synthetic data via stereo," in *Proc. Conf. Robot Learn.*, 2022, pp. 938–948.
[21] T. Lips, V.-L. De Gusseme, and F. Wyffels, "Learning keypoints from synthetic data for robotic cloth folding," 2022, *arXiv:2205.06714*.
[22] A. Goyal et al., "IFOR: Iterative flow minimization for robotic object rearrangement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 14787–14797.
[23] J. Mahler et al., "DEX-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," 2017, *arXiv:1703.09312*.
[24] W. Liu et al., "Robotic picking in dense clutter via domain invariant learning from synthetic dense cluttered rendering," *Robot. Auton. Syst.*, vol. 147, Jan. 2022, Art. no. 103901.
[25] (Blender Found., Stichting Blender Found., Amsterdam, The Netherlands). *Blender—A 3D Modelling and Rendering Package*. (2018). [Online]. Available: http://www.blender.org
[26] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-Net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 5620–5627.
[27] B. T. Imbusch, M. Schwarz, and S. Behnke, "Synthetic-to-real domain adaptation using contrastive unpaired translation," in *Proc. IEEE 18th Int. Conf. Autom. Sci. Eng. (CASE)*, 2022, pp. 595–602.
[28] D. Duplevska, M. Ivanovs, J. Arents, and R. Kadikis, "Sim2Real image translation to improve a synthetic dataset for a bin picking task," in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, 2022, pp. 1–7.
[29] X. Yang, X. Fan, J. Wang, and K. Lee, "Image translation based synthetic data generation for industrial object detection and pose estimation," *IEEE Robot. Autom. Lett.*, vol. 7, no. 3, pp. 7201–7208, Jul. 2022.
[30] M. Denninger et al., "BlenderProc," 2019, *arXiv:1911.01911*.
[31] K. Greff et al., "Kubric: A scalable dataset generator," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 3749–3761.
[32] E. Coumans and Y. Bai. "PyBullet, a python module for physics simulation for games, robotics and machine learning." 2019. [Online]. Available: http://pybullet.org
[33] N. Dvornik, J. Mairal, and C. Schmid, "Modeling visual context is key to augmenting object detection datasets," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 364–380.
[34] X. Li et al., "A sim-to-real object recognition and localization framework for industrial robotic bin picking," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 3961–3968, Apr. 2022.
[35] D. Morrison et al., "Cartman: The low-cost cartesian manipulator that won the Amazon robotics challenge," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 7757–7764.
[36] M. Schwarz et al., "Fast object learning and dual-arm coordination for cluttered stowing, picking, and packing," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 3347–3354.
[37] M. Schwarz and S. Behnke, "Stillleben: Realistic scene synthesis for deep learning in robotics," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 10502–10508.
[38] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *Proc. 16th Eur. Conf.*, 2020, pp. 319–345.
[39] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1925–1934.
[40] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 746–760.
[41] Y. Zhang et al., "Physically-based rendering for indoor scene understanding using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 5287–5295.
[42] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: Can 5m synthetic images beat generic ImageNet pre-training on indoor segmentation?" in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2678–2687.
[43] M. Huh, P. Agrawal, and A. A. Efros, "What makes ImageNet good for transfer learning?" 2016, *arXiv:1608.08614*.
[44] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[45] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "The best of both modes: Separately leveraging RGB and depth for unseen object instance segmentation," in *Proc. Conf. Robot Learn.*, 2020, pp. 1369–1378.
[46] C. Xie, Y. Xiang, A. Mousavian, and D. Fox, "Unseen object instance segmentation for robotic environments," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1343–1359, Oct. 2021.
[47] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[48] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual worlds as proxy for multi-object tracking analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4340–4349.
[49] M. Johnson-Roberson, C. Barto, R. Mehta, S. N. Sridhar, K. Rosaen, and R. Vasudevan, "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" 2016, *arXiv:1610.01983*.
[50] N. Duczek, M. Kerzel, and S. Wermter, "Continual learning from synthetic data for a humanoid exercise robot," 2021, *arXiv:2102.10034*.
[51] W. Masarczyk and I. Tautkute, "Reducing catastrophic forgetting with learning on synthetic data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2020, pp. 252–253.
[52] G. M. Van de Ven and A. S. Tolias, "Generative replay with feedback connections as a general strategy for continual learning," 2018, *arXiv:1809.10635*.
[53] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: A hands-on survey," *IEEE Trans. Visualization Comput. Graph.*, vol. 22, no. 12, pp. 2633–2651, Dec. 2016.
[54] G. Bradski and A. Kaehler, "The OpenCV library," *Dr. Dobb's J. Softw. Tools*, vol. 25, no. 11, p. 120–125, 2000.
[55] J. Fraley, A. Imeri, I. Fidan, and M. Chandramouli, "A comparative study on affordable photogrammetry tools," in *Proc. ASEE Annu. Conf.*, 2018, pp. 1–8.
[56] M. E. Gurses et al., "Qlone®: A simple method to create 360-degree photogrammetry-based 3-dimensional model of cadaveric specimens," *Oper. Neurosurg.*, vol. 21, no. 6, pp. E488–E493, 2021.
[57] D. Rolnick, A. Ahuja, J. Schwarz, T. Lillicrap, and G. Wayne, "Experience replay for continual learning," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/fa7cdfad1a5aaf8370ebeda47a1ff1c3-Paper.pdf
[58] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.
[59] W. Abdulla. "Mask R-CNN for object detection and instance segmentation on keras and TensorFlow." 2017. [Online]. Available: https://github.com/matterport/Mask_RCNN
[60] T.-Y. Lin et al., "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

**Artúr István Károly** (Member, IEEE) received the B.Sc. and M.Sc. degrees in mechatronic engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree with the Doctoral School of Applied Informatics and Applied Mathematics and the Antal Bejczy Center for Intelligent Robotics, Óbuda University, Budapest.

His research focuses on developing novel methods for deep learning in robotics that enable the use of robust deep learning models with limited and automatically annotated data, as well as unsupervised or semi-supervised learning techniques.

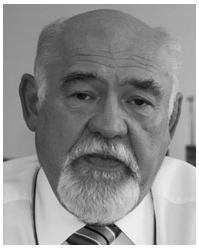
**Sebestyén Tirczka** received the B.Sc. and M.Sc. degrees in electrical engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 2019 and 2021, respectively.

He is currently a Research Engineer with the Antal Bejczy Center for Intelligent Robotics, Óbuda University, Budapest. His research interests include applying classical image processing and deep learning in visual informatics and robotics, focusing on laboratory automation and agrifood applications.

**Huijun Gao** (Fellow, IEEE) received the Ph.D. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2005.

From 2005 to 2007, he was a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB, Canada. Since 2004, he has been with the Harbin Institute of Technology, where he is currently a Chair Professor and the Director of the Research Institute of Intelligent Control and Systems. His research interests include intelligent and robust control, robotics, mechatronics, and their engineering applications.

Dr. Gao serves/served as the Editor-in-Chief for the IEEE/ASME TRANSACTIONS ON MECHATRONICS, the Co-Editor-in-Chief for the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and an Associate Editor for *Automatica*, the IEEE TRANSACTIONS ON CYBERNETICS, and the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS. He is a Vice President of the IEEE Industrial Electronics Society and a Council Member of the International Federation of Automatic Control. He is a member of Academia Europaea and a Distinguished Lecturer of the IEEE Systems, Man, and Cybernetics Society.



**Imre J. Rudas** (Life Fellow, IEEE) graduated in mechanical engineering from Bánki Donát Polytechnic, Budapest, Hungary, in 1971. He received the master's degree in mathematics from Eötvös Loránd University, Budapest, in 1977, and the Ph.D. degree in robotics and the Doctor of Science degree from the Hungarian Academy of Sciences, Budapest, in 1987 and 2004, respectively.

He is a Rector Emeritus and a Professor Emeritus with Óbuda University, Budapest. He has published six books, 12 university books, more than 850 papers in various journals and international conference proceedings, and received more than 7500 citations. His present areas of research activities are computational cybernetics, robotics, and computational intelligence.

Dr. Rudas is a Junior Past President of the IEEE Systems, Man, and Cybernetics Society.

**Péter Galambos** (Member, IEEE) received the M.Sc. and Ph.D. degrees in mechanical engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 2006 and 2013, respectively.

He was a Research Intern with Toshiba Corporate Research and Development Center, Kawasaki, Japan, from 2007 to 2008, then joined the Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA-SZTAKI), Budapest, where he held a "Young Researcher" Scholarship from 2010 to 2012. From 2011 to 2015, he served as a Team Leader with MTA SZTAKI and coordinated the development of the VirCA VR system and its research applications. He joined Óbuda University, Budapest, in 2013, participating in robotics-related research and development activities and education. He is currently the Director of the Antal Bejczy Center for Intelligent Robotics and the Deputy Director of the University Research and Innovation Center, Óbuda University. His research interests include advanced industrial robotics and control systems, cyber–physical systems, and virtual reality.