C.DOT - Convolutional Deep Object Tracker for Augmented Reality Based Purely on Synthetic Data

Kevin Kennard Thiel, Florian Naumann, Eduard Jundt, Stephan Günnemann, and Gudrun Klinker

Abstract—Augmented reality applications use object tracking to estimate the pose of a camera and to superimpose virtual content onto the observed object. Today, a number of tracking systems are available, ready to be used in industrial applications. However, such systems are hard to handle for a service maintenance engineer, due to obscure configuration procedures. In this article, we investigate options towards replacing the manual configuration process with a machine learning approach based on automatically synthesized data. We present an automated process of creating object tracker facilities exclusively from synthetic data. The data is highly enhanced to train a convolutional neural network, while still being able to receive reliable and robust results during real world applications only from simple RGB cameras. Comparison against related work using the LINEMOD dataset showed that we are able to outperform similar approaches. For our intended industrial applications with high accuracy demands, its performance is still lower than common object tracking methods with manual configuration. Yet, it can greatly support those as an add-on during initialization, due to its higher reliability.

Index Terms—Object tracking, deep learning, industry 4.0, neural network, synthetic data, augmented reality, computer vision

1 Introduction

Augmented Reality (AR) is a growing market. While, in 2015, the market value was $640.2M, it is estimated to be about $120B in 2020 [1] – an increase of more than 187 percent in 5 years.

One reason is the high potential within the field of industrial manufacturing, such as the automotive industry. For example, AR can be used throughout the entire life cycle of a car or other products [2], [3], [4]: as concept visualization support during the design segment [5], as training application providing instructions [6] or as actual / target comparison [7], [8] and joining process tool within the production segment [9], as maintenance and repair tool [10], [11], and as tool to feature show cases in the after sales segment [12]. Beyond production itself, AR also finds applications in plant planning and maintenance [13], [14], [15] as well as for logistics operations within the plants [16].

In each of these fields we find use cases which benefit strongly from congruent visualizations of virtual content, superimposed on real objects, allowing for natural interaction, intuitive information transfer and immersion [17].

- *Kevin Kennard Thiel is with the Volkswagen Group, 38440 Wolfsburg, Germany, and also with the Technical University of Munich, 80333 Munich, Germany. E-mail: kevin.thiel@volkswagen.de.*
- *Florian Naumann and Eduard Jundt are with the Volkswagen Group, 38440 Wolfsburg, Germany. E-mail: {florian.naumann, eduard.jundt}@volkswagen.de.*
- *Stephan Günnemann and Gudrun Klinker are with the Technical University of Munich, 80333 Munich, Germany. E-mail: {guennemann, klinker}@in.tum.de.*

Manuscript received 11 Feb. 2021; revised 18 May 2021; accepted 31 May 2021. Date of publication 14 June 2021; date of current version 27 Oct. 2022. (Corresponding author: Kevin Kennard Thiel.) Recommended for acceptance by K. Kiyokawa. Digital Object Identifier no. 10.1109/TVCG.2021.3089096

1.1 Pose Estimation for Industrial AR Applications

To create superimposed augmented reality experiences, the system needs to know the observer pose, relative to a point of interest on an observed real object. This observer pose is often associated with a virtual camera on a mobile device such as a tablet or a head-mounted display. It is given by at least six degrees of freedom (6-dof): the position (x, y, z) and the orientation (ϕ, θ, ψ).

Even though many physical sensing modalities can be employed [18], optical pose estimation is predominant, employing object tracking or object detection methods. Depending on the purpose, this may also be called object or camera localization.

1.1.1 Pitfalls of Optical Pose Estimation

Based on experience within several fields of automotive manufacturing [19], we found it very difficult to create accurate, fast and robust tracking or detection solutions in settings, e.g., in a car factory or a service station. Such scenarios often exhibit bright, uncontrolled illumination environments, and the objects that need to be tracked may have surfaces with dark coatings (car paints) or metallic components. They often exhibit strong specular reflections, and tend to be only minimally textured.

These conditions violate common feature detection and tracking assumptions. Such algorithms tend to expect an abundance of reproducible, stationary object points with clearly distinguishable photometric and geometric properties (textures and edges), seen under idealized illumination

conditions. In real environments, it is often the case that only few features can be tracked reliably.

This creates highly demanding scenarios compared to laboratory setups. Products such as the Microsoft HoloLens, the Magic Leap One or tools like Apple's ARKit or Google's ARCore cannot fulfill the accuracy demands in such dynamic manufacturing environments. They often do not work on reflective or dark surfaces and they are not able to track individual objects precisely.

1.1.2 Opportunities for Simplifications

Despite the high demands, industrial settings also offer some room for simplifications. Often setups can be arranged to specifically provide suitable empty work spaces, e.g., in applications using inspection and presentation areas, workshops, assembly lines and others. The object is typically placed on a rack or stands freely (like a car). Those existing scenarios are designed very strictly for isolated tasks, making occlusions unlikely. If occlusion does occur after all, object detection approaches recover as soon as enough features of the tracked object reappear.

1.1.3 Object Tracking Versus Object Detection

Even though object tracking and detection approaches for viewpoint localization operate on very different algorithmic principles, they are often referred to interchangeably as black-box *trackers* in AR applications. For example, GPS-based, cell-based and optical outside-in tracking systems[1] and even the AR Toolkit [20] are employing localization methods based on one-shot object detection approaches. On the other hand, inertial trackers and optical SLAM-based approaches [21], [22] use dynamic object tracking.

As the terms tracking and detection are used interchangeably and since tracking and detection methods based on optical features are both lacking sufficient image features with constant appearance in industrial AR applications (see Section 1.1.1) we adopt the generic use of the term *tracker* in this paper.

1.2 Tuning Trackers for Specific Automotive Scenarios

Several object trackers exist that have been designed specifically for the automotive industry [19], [23], [24]. They are based on geometrical features extracted from edges in a camera image. They have proven to create better results in industrial contexts than texture-based approaches thus far.

However, these trackers are quite difficult to deploy [25]. To achieve high-quality augmentations, they provide a large number of parameters such that they can be configured (i.e., tuned) specifically for every use case, depending on the observed object, the environmental illumination conditions and the hardware and sensors. Service engineers need to perform such parameter tuning manually. And they need to do it on a vast parameter space for algorithms that appear to them as blackboxes. Some parameter changes have been seen to exhibit strong, unexpected effects on the tracking result [25], [26].

1. https://ar-tracking.com & https://optitrack.com

To yield acceptable results, service engineers have to create test cases for every new object of interest, requiring costly data acquisition in terms of time and space. They have to do this on real objects or on not yet existing objects still under design, disregarding the variety of possible versions. In consequence, only rather general configurations are generated for several objects together. This may result in poor alignments or even in total loss of tracking in some of the targeted use cases.

To support this tedious task, we present a framework to automate and objectify the parameter tuning process. It uses an iterative analysis cycle based only on synthetic data and produces an optimized configuration [25].

1.3 Automatic Generation of Trackers for Application Scenarios

In this paper, we propose to completely move away from tuning parameters of a generic tracker. Rather, our new approach generates specific, one-of-a-kind trackers for every targeted model (like a car) based on its respective CAD (computer-aided design) data. Since it is specifically created based on data of a specific scenario, it will be ideal for the given tracking task. In theory, we hereby shift the focus from tuning the parameters of a hand-crafted algorithm to the elaboration of a generic parametric model.

We use a convolutional neural network (CNN) to output the camera pose in six dimensions when given RGB camera images of the real world object as an input. For that purpose the network is trained only on automatically generated synthetic data of the desired object. We named this approach C.DOT (Convolutional Deep Object Tracker). Fig. 1 shows estimation results using C.DOT on real image sequences.

Using a CNN yields several benefits, as a tracker can be created specifically for the object of interest and the use case, while still being able to consider variations, such as different object colors, holes, gaps or not yet assembled parts. Following a holistic approach - by processing not just individual feature points but the entire image and object - creates the possibility for higher robustness with respect to changes in the environmental conditions. As the training can be performed without considering the targeted mobile hardware (i.e., tablet or data glasses), the final network can become almost independent of it – except for a careful consideration of the focal length and distortion of the camera employed.

By using only synthetic data for training, cumbersome data acquisition from real objects becomes unnecessary and the process can be fully automated. This creates the possibility to have a system automatically ready even before a real object is produced. It is very suitable to applications involving the manufacturing of prototypes or general parallelization. The data for training the CNN can be derived from a digital model (CAD) and is already present within the digitalized manufacture. A specific CNN can be trained for every individual object – and with industry 4.0 and the internet-of-things in mind, this unique CNN can be stored within a database or even deployed within the object itself. This makes it possible for service engineers to acquire the necessary data for a tracker on demand without the necessity of openly sharing the geometrical data – often a consideration of confidentiality. Unfortunately, this approach is far from being trivial and brings challenges of its own.

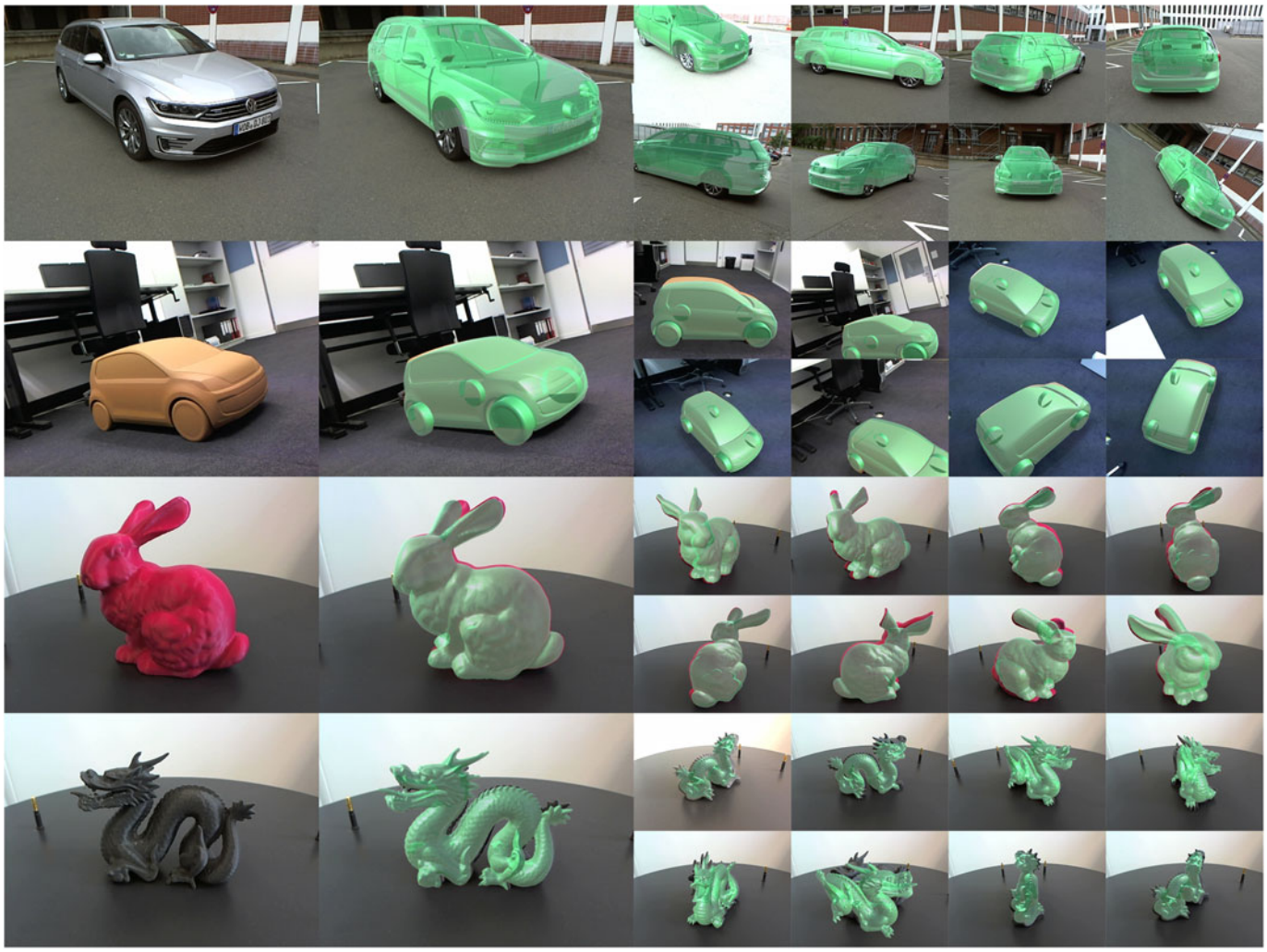

Fig. 1. Example images for estimation results on real image sequences for all four objects shown with the associated augmentation (a green transparent overlay). From top to bottom: VW Passat GTE, VW e-up!, Stanford Bunny, Stanford Dragon.

1.4 Problems Using Synthetic Data: The Reality Gap

The quality of the training data is essential when we want to teach a tracker the intended behavior by training a CNN. The training data needs to be representative of the use case. Otherwise the trained model will not perfectly match when applied during runtime to real data, due to missing information. Using real data of the use case is the easiest solution to yield best performance. Yet, for reasons discussed above, this is highly impractical.

Using synthetic data is problematic since the synthetic data is built only upon a model of the reality and thus is not a perfect imitation of reality. A lack of information cannot be avoided. This inherent difference is also known as the *reality gap* [27]. The term can be used to describe how well the simulation model fits the requirements of the reality. The better this model fits, the less loss of information is present while also keeping as much flexibility for environmental variations or sensory artifacts.

Bridging this reality gap is the process of finding a model which fits best. If this process is considered well, merely synthetic data may be sufficient to train a network, which generalizes well during real world usage, without ever having to consider real world data. That is why we focus on explaining in great detail our rendering aspects for the creation of realistic images which are induced with variance for generalization during training using a *Domain Randomization* [28] inspired data enhancement approach.

This paper presents our way of creating the C.DOT system, based on purely synthetic data. We present the results we achieved thus far and critically discuss both the system and the results.

2 State of the Art

Optical tracking is a well established topic in Augmented Reality. Good overviews can be found in [29], [30], [31]. These articles show the broad use across different fields such as computer vision, photogrammetry and odometry.

2.1 Traditional Approaches to Optical Tracking

Traditional tracking approaches have been based on the Perspective-n-Point (PnP) approach [32], [33], [34] to estimate the rotation R and translation t of an object in relation to the camera from three or more points, with extensions to include robust estimation (RANSAC) [35] and to match 3D point clouds to an object reference (ICP) [36]. Robust feature detection is commonly performed based on SIFT (Scale-Invariant Feature Transform) [37] or SURF (Speeded Up

Robust Feature) [38], as well as LINEMOD [39], which uses gradients and normals within the image to locate textureless objects in chaotic scenes.

With SLAM [21] and PTAM [22], it has become possible to track in an unknown environment, constructing a 3D model as part of the task. This is often used in the field of robotics for odometry [40]. Yet, for industrial applications, object data is often already present. Hence, no industrial tracking applications are known to rely solely on SLAM, PTAM or their derivatives. These are mostly integrated just to boost robustness of the main approach.

2.2 Hybrid Approaches to Optical Tracking

Recently, neural networks and deep learning [41] have started gaining importance. Hybrid approaches such as SegICP [42] and BB8 [43] are bridging the gap between traditional approaches and pure neural network driven approaches. A benefit of SegICP is that no pose for initialization is needed. BB8 [43] focuses on a pure neural network approach. It is supported by traditional approaches to estimate 6-dof even for occluded objects. It was validated on the LINEMOD dataset [44].

2.3 Approaches Based on Neural Networks

Tracking solutions that rely only on artificial neural networks are generally based on CNNs, due to the necessity of image processing. Many try to estimate the position and scale of objects in 2D images [45], [46], [47], [48], [49], [50]. They can only be utilized for simpler detection tasks.

Recently, a number of systems have emerged that are capable of full 6-dof tracking, based on combinations of synthetic and real data. PoseCNN [51] was trained on an own dataset with poses of 21 objects in 91 videos and on the LINEMOD dataset [44]. As for BB8, objects are localized and segmented in 2D RGB images.

The PoseNet system [52] estimates camera poses on 224×224 RGB images of rooms or outdoor environments with up to 50000 km^2 coverage. It operates in 5 ms with an accuracy of 0.5 m and 10° for indoor and 2 m and 6° for outdoor environments.

Wohlhart *et al.* [53] focus on highly distinct descriptors between different objects. For training they use real and synthetic images, with the synthetic data being created from 1241 positions derived by subdividing an icosahedron and by enhancing the data further with different noise patterns to simulate changing backgrounds. They tested depth, RGB and RGB-D images, with RGB-D yielding the better results.

Su *et al.* [54] estimate a 3-dof object-observer-relation. They use mostly synthetic data with structure-preserving object deformations resulting in 30000 object variations for 12 distinct classes. Further enhancements by random background images and different illumination setups create a dataset with more than 2 million images. These are then combined with 12000 real images of the VOC12 dataset [55]. Su *et al.* stated that using random images and a high number of training images for every object increases accuracy significantly.

Papon *et al.* [56] simulate 96×96 pixel RGB-D images enhanced with a small number of real images to recognize and estimate the pose of furniture.

Garon *et al.* [57] use image sequences to estimate full 6-dof poses with the focus on being robust against object occlusion. For training they use 250000 synthetic images, all (100 percent) of which are enhanced with background images of the SUN3D dataset [58], 95 percent with noise and 40 percent with motion blur. For training 180 real RGB-D images are added. The estimation is performed on an actual frame with 150×150 pixels and a rendered image from the perspective of the last estimated pose. This results in a necessary initialization of the algorithm during runtime from a predefined pose. The approach was able to estimate every 8 ms with a general accuracy of 4 mm and 2° for the tested objects.

Wang *et al.* [59] train a 6-dof pose and size estimator on *mixed-reality datasets* (real RGB-D images augmented with aligned virtual objects) which can even handle symmetric objects. Yet, following the observations by Manhardt *et al.* [26], the virtual objects are likely to stand out. Flat textures, different sensory artifacts and lightning can create strong obstructive descriptors for virtual objects, compared to real ones.

2.4 Approaches Using Only Synthetic Data

While the approaches presented in Section 2.3 may use synthetic data as a common practice, they also use real world data during training to some extent, creating mixed datasets. However, obtaining even small amounts of real data is hard and cost-intensive especially for industrial applications with many changing objects and parts. Hence, training only on synthetic data is necessary while maintaining our demand on high accuracy of less than 1 mm.

Approaches just using synthetical training data, like Tan *et al.* [60] or Garon *et al.* [61], show remarkable accuracies in their field. However, they rely on RGB-D images providing depth information, or use model data during runtime.

In this paper, we focus on RGB images only, due to technical limitations of RGB-D cameras in outdoor scenes and on specular objects (such as reflective car coatings or metal). Furthermore, these cameras are cheaper and more common. We also limit ourselves on pure synthetical images for training. Those can be derived from CAD by existing industrial processes allowing for full automation. Model data is treated highly confidential by the industry. Hence, we do not want to rely on any such data during runtime. Thus the only network input shall be images. This removes the need for any geometrical data during inference. All this limits our scope to initialization-free approaches using only synthetical RGB images.

To the best of our knowledge, only two methods satisfy these requirements. Those are by Kehl *et al.* [62] and by Rambach *et al.* [63], with the latter being an improvement of the former. Su *et al.* [64] present an application of the tracker of Rambach *et al.* to detect and regress the pose of an object in multiple states. Yet their evaluation lacks results for 6-dof pose estimation on real data due to missing ground truth. Thus, we will focus on Rambach *et al.* They use random poses around a unit sphere to generate 20000 to 30000 RGB images of 448×488 resolution, with a random indoor background being added to the images. To bridge the reality

gap, they utilize feature transformation as a domain adaptation strategy.

The training data is split into six groups. The following influences are added individually per group: Gaussian noise, random contrast, brightness adjustments, motion blur, speckle noise and a mixture of all of the previous. Afterwards, they apply a pencil filter to each image, transforming it to a single channel edge domain. This creates a representation that is invariant to illumination but loses the ability to handle possible color descriptors. For training, Rambach *et al*. use a PoseNet architecture and an ADD error [44].

The approach presented in this paper also works with purely synthetic data. It goes beyond Rambach *et al*.'s work in the following aspects: a) We use a deeper architecture, b) we define our problem space for indoor and outdoor industrial contexts, c) we consider rotations during training, and d) we use over two magnitudes more of training data. Furthermore, we utilize *Domain Randomization* instead of adaptation to bridge the reality gap (see Section 3.2). To this end, we focus on different aspects such as the illumination setup and reflections while retaining information for possible descriptors like color.

Our work presents an approach for industrial applications using only synthetic training data. By greatly enhancing the training data, robust full 6-dof estimations on real RGB camera images are achieved, exceeding the related work.

3 Concept

Our concept has to fulfill requirements for industrial applications which we will specify first. Thereafter we discuss how to bridge the reality gap related to our use of purely synthetic data.

3.1 Tracking Requirements of Industrial AR Applications

As stated in the introduction, it is our goal to automatically create trackers for automotive AR applications based on CNNs. The following requirements must be met when providing augmentation services in real industrial applications:

- *Robustness against erratic camera motion* - We consider an all-encompassing, wide range of different camera perspectives including perspectives where the object is cropped by the edges of the camera image plane.
- *Applicability to outdoor scenes and to scenes with specular objects* - We refrain from using RGB-D cameras. We only consider a single image data stream from an RGB camera. It is cheap and most common for mobile devices. However, this makes depth estimation more difficult than using RGB-D cameras.
- *Minimization of tracking loss induced by large inter-frame changes or by partially cropped object views* - We determine camera poses by object detection. This is highly beneficial for large objects such as cars or trucks.
- *Augmentation accuracy* - For actual/target comparisons and joining processes, we aim at a pose estimation accuracy of less than 1 percent of the objects diameter. For instruction or training use cases, we consider 10 percent of the objects diameter to be already sufficient.
- *Unit standard* - Standard unit metric in our automotive field is *millimeter* (mm). All our real and virtual objects and our training follow this standard. Yet we present most of our accuracy results in a *normalized unitless metric* for easier comparison (see Eqs. (2) and (3) on ADD^{*}).
- *Use on mobile devices* - The latency must be less than 50 ms for real time usage, while data glasses demand latencies of ≤ 16 ms.
- *Speed of off-line learning* - The training of the network is not time critical, as we can rely on parallelized development workflows. Its speed is not considered as a goal.

With this work we present a robust Tracking-by-Detection solution which is automatically created specifically for an arbitrary object based only on its digital geometry and material working on every standard RGB camera. Hence, using synthetical data only, bridging the reality gap becomes an open issue.

3.2 Bridging the Reality Gap

When creating synthetic data several properties of the reality, like sensory artifacts, specific object materials or lighting properties, are often only simulated weakly for efficiency reasons. This creates a *domain shift* or the *reality gap*, due to different data distribution among both domains, whereas the tasks between both stays the same. This particular transfer problem is addressed by the field of *Domain Adaptation (DA)* [65].

When utilizing synthetical RGB data for learning tracking methods, common DA solutions are mixed datasets, photorealistic renderings or feature transformation. Mixed and photorealistic data are not economical for industrial applications, for the high cost in real data retrieval (e.g., ground truth acquisition or measurement of real materials). In comparison feature transformation is more economical. It just uses a projection of the synthetic and real data into a lower dimensional, latent feature space such that the discrepancy between both distributions is decreased [65]. It can be interpreted as an intermediate domain in which both domains equal each other. The projection can be learned or directly applied to the data as a transformation (i.e., edge detection or Hough transformation). As the feature vector is reduced, information gets lost (i.e., color in the frequency domain). It is therefore difficult to find an intermediate domain, while still keeping crucial information (like color) for the descriptor. This approach was adapted by Rambach *et al*. [63], which we compare ourselves to in Section 5.4.

For our industrial setup we can assume to find the same geometrical features in each domain. All objects of interest are produced highly accurate on the basis of CAD data, which we also use for creation of the synthetical data. Thus, there exists at least a set of latent, homogeneous features among the synthetic and real domain. In contrast other features, like noise or environmental details, may vary or are not present between domains.

Domain Randomization (DR) [28] is a rather exceptional approach that uses a strong data enhancement strategy to create a broadly distributed source domain. The core idea is to

exploit the generalization ability of a network. When randomizing different aspects of the synthetic data, the network learns the recurring latent features of the data, while ignoring facets that exhibit variance. This can be done for all kinds of features that are different between the real and synthetic domain. It is strongly based on imperfection characteristics of a utilized sensor and unknown environmental conditions. Similar to feature transformation, DR limits the ability of the network to use such information as descriptors. Thus, modeling an intermediate domain by design is not required. The network is able to find a generalized feature domain by itself, specifically for the observed object, based on the training data which we model with possible variations of reality.

There do exist several approaches which already utilize DR. Montserrat *et al*. [66] use DR on synthetic images, which they combine with photorealistic renderings, to train a two-level 6-dof estimator. They rely on model data during inference, from which we refrain from for the reasons stated in Section 2.4. The works of Ren [67], Khirodkar [68] and Tremblay [69] focus on detection and 3-dof estimation. Yet they offer a great contribution showing the potential of DR in overcoming the reality gap, increasing overall accuracy. Tremblay *et al*. [69] even found out that DR can perform better than photorealistic renderings and showed that DR in conjunction with fine-tuning on mixed datasets creates a boost in performance. We have adopted the *Domain Randomization* approach for the C.DOT system.

4 Implementation

This section introduces the architecture and structure of our network. We describe the label creation and the rendering of the training images and how we enhance them to bridge the reality gap. We then define the loss metric used to train for superimposition. The section ends with a short description of a demonstrator.

4.1 Convolutional Neural Network Architecture

The CNN is the essential part of our solution. It is to be trained on synthetic image data of an object without any further consideration of the underlying architecture, parameters or hyperparameters.

Our implementation is based on Inception-ResNet-v2 [70]. It combines properties of an Inception net, such as sequential and parallel convolutional layers, with properties of Resnet such as residual connections. This way, the gradient can easily traverse the network, allowing for deep but trainable networks. Inception-ResNet-v2 was developed for classification purposes on 1000 different classes, resulting in even higher accuracy than the Inception-v4 network.

Adapting it for our regression problem, we only use the feature extractor of its architecture on top of our fully connected multilayer perceptron block (MLP) (see Fig. 2). For initialization we keep the weights which were present after training for object classification on the ImageNet dataset. This reduces our training duration noticeably, as not all filters have to be trained anew, while others can be changed completely to fully adapt to our problem. Yet one could try to freeze the feature extractor as recommended by Hinterstoisser *et al*. [71].

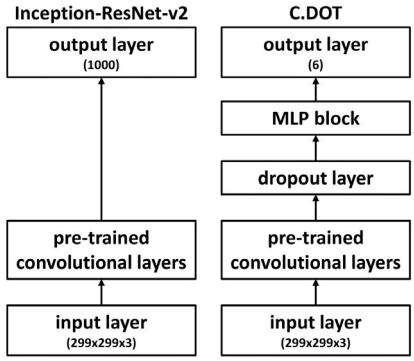

Fig. 2. Architecture of the entire network. The pre-trained convolutional layers for feature extraction represent the adapted inception-resnet-v2 architecture.

The feature extractor is followed by a dropout layer. During training this layer has a regular dropout rate of 0.25 (every 4th neuron will be deactivated per layer). The dropout layer is followed by five fully connected layers. These reduce the number of neurons from 1000 to 6 for 6-dof pose estimation (Section 4.2). The layers are batch normalized. We use a Parametric Rectified Linear Unit (PReLU) [72] as activation function. The weights were set by He-Normal, as proposed by He *et al*. [72]. Overall, this yields better and faster convergence during training compared to ReLU.[2]

The properties of the final network are listed in Table 1. The network is less complex than AlexNet (60 million parameters) and VGG16 (138 million parameters). Stored as HDF5 (Hierarchical Data Format), it occupies 654 MB disc space.

The training data (Section 4.3) is split into 70 percent training, 20 percent validation and 10 percent test data. A subset of the 10 percent test data are used for our evaluation on synthetic data (Section 5.2). For evaluations on real images complete new sets of data are used (Section 5.3).

The usage of the training data is organized by a generator. The generator loads random images and their associated labels defined by a batch size of 32 images per epoch. The image order for training is not defined. Yet, due to randomness, we can assume that it will be normally distributed for the duration of the training. Additionally, the generator performs normalization and data enhancement on all batch images (Section 4.4.2).

We use Tensorflow[3] in combination with KERAS[4] for development, implementation and training of the CNN. We use CUDA[5] for implementation on the GPU. All CNNs were trained and tested on the system listed in Table 2.

4.2 Pose Definition and Label Creation

Since large datasets require high creation times and memory consumption in general but in return prevent overfitting more effectively, a suitable size trade-off has to be found. We thus define our problem space as a hollow sphere S around the object $\mathcal{M}$ (see Fig. 3). The minimal distance for a camera position $c = (x, y, z)$ is the radius r_{min} of a bounding sphere, to get as closely as possible to the object without intersecting it. Accordingly, the center of S - the origin O of

2. $f(x) = x^+ = max(0, x)$
3. https://www.tensorflow.org/
4. https://keras.io/
5. https://developer.nvidia.com/cuda-zone

TABLE 1
Network Properties

Property	Amount
input dimension	[299, 299, 3]
output dimension	[6]
layers (total)	781
layers (convolutional)	244
layers (fully connected)	5
filters	76032
parameters (total)	56 986 462
parameters (trainable)	56 921 698
parameters (non-trainable)	64 764

our problem space - is not necessarily equal to the geometric center of the object. The maximum distance is defined to be $r_{max} = 3r_{min}$, as this is a typical range for observation in our industrial applications.

All camera positions c need to be evenly distributed around the center O. Therefore, the mean pose is zero, creating an equal probability for every dimension. A regular, convex polyhedron (i.e., icosahedron) is able to solve this problem. Hence, similar to [60], we use a polyhedron decomposition until we have vertices equal to the desired number of camera positions. Projecting those positions to our spheres S outer bounds fulfills our requirement for evenly distributed camera positions. To cover the entire volume of S, the distance of each point to the center O is randomly set within the range $[r_{min}, r_{max}]$.

The orientation of the camera could be represented as roll, pitch and yaw angles $(\phi_O, \theta_O, \psi_O)$ with regard to the reference frame of our problem space O. We call this *absolute* representation. As the rotation value around an angle $0° \leq \alpha \leq 360°$ jumps at the transition from $360°$ to $0°$, small changes in the orientation can result in large changes for estimation, afflicting the training. We therefore use a *relative* representation $(\phi_C, \theta_C, \psi_C)$ for the orientation.

Given the camera position c, we initially orient the camera to look at the sphere center O. We furthermore align the right vector of the camera to point in the direction of the cross product of the up-vector of the object and the viewing direction of the camera. This introduces a local reference frame C and an associated rotation matrix R_C for the initial camera orientation. We use this local reference frame C and our relative roll, pitch and yaw angles $(\phi_C, \theta_C, \psi_C)$ to define the final orientation of the camera. Thus the final rotation matrix is $R = R_C \cdot R_{(\phi_C, \theta_C, \psi_C)}$. We limit the roll angle ϕ_C to a range of $\pm 45°$, as we assume camera images of mostly upright angles in user-centric AR applications. To consider cropping of the camera field of view (FOV), we limit pitch θ_C and yaw angle ψ_C to be within $\pm$ half of the FOV. For a FOV of $60°$ this will result in θ_C, ψ_C being in the range $[-30°, +30°]$. This way, estimation will only consider situations when more than half of the object is visible in the image. The relative orientation $(\phi_C, \theta_C, \psi_C)$ can later be used together with R_c to calculate an absolute orientation.

TABLE 2
System Specification

Component	Specification
CPU	2x Intel Xeon E5-2667 v4 3.2 GHz
RAM	64 GB
GPU	Nvidia Quadro P6000
OS	Windows 7, 64 Bit

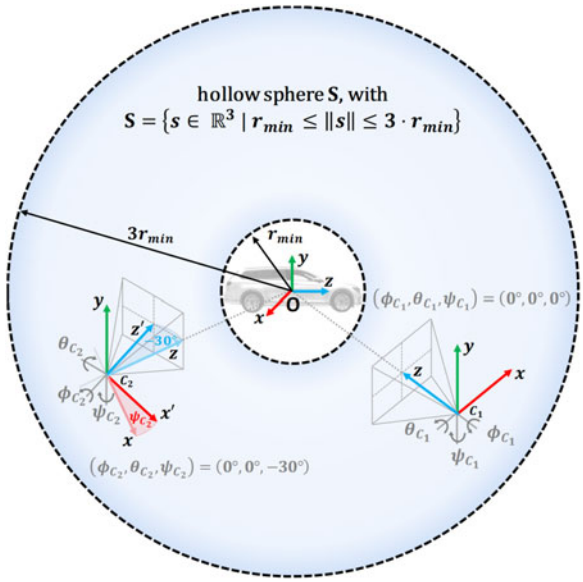

Fig. 3. Definition of the data space for the problem space.

Such relative representation has the downside effect of depending on the camera position. A correct estimation of the relative orientation will lead to a wrong absolute orientation for a bad estimation of the camera position. However, the benefit is that in our case only a smaller value range needs to be estimated by the network (e.g., $\theta_C, \psi_C = [-30°, +30°]$ and $\phi_C = [-45°, +45°]$). Every movement in this range yields continuous values for the object still being within the FOV.

We use the here specified data space to create poses to be used as labels for training, as well as camera poses for rendering of the synthetic training images. Hence, we fuse camera position c and *relative* orientation together to a pose $p = (x, y, z, \phi_C, \theta_C, \psi_C)$ inheriting all 6-dof. Such poses are limited to perspectives where the object is at least partially visible. All resulting poses P are stored as labels in a file.

4.3 Image Rendering and Creation of Training Data

We synthetically generate RGB images to be used as input data to train the network. An exemplary image is shown in column A of Fig. 4. We had to find a trade-off between image size and inherent information. Larger images yield a lot of information for estimation. But next to higher memory consumption and workload, they also contain details such as dust, dirt, scratches, production residues and more which are difficult to simulate in synthetic data and may limit the generalization of the network.

For image creation, we kept a common resolution of 299×299. With such resolution and a FOV of $\varphi = \varphi_h = \varphi_v = 60°$, it is mathematically possible to visualize a lateral displacement of 0,386 cm per pixel for 100 cm distance

$$\frac{2 \cdot d(O,c) \cdot tan(\varphi/2)}{w} = \frac{2 \cdot 100\text{cm} \cdot tan(30°)}{299px} = 0.386 \frac{\text{cm}}{px} \quad (1)$$

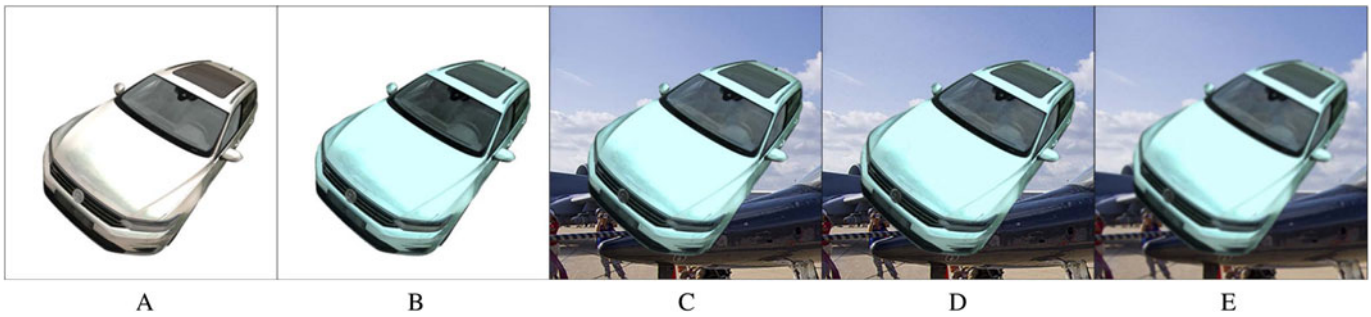

Fig. 4. Random example for complete enhancement procedure on one pose for VW Passat GTE. F.l.t.r.: image after rendering (A), color and brightness shifts (B), background (C), noise (D) and blur added (E). The final image is used for training.

The images are rendered with Unity3D.[6] We read the data file with all labels P and set the camera iteratively to every defined pose p. The virtual camera's field of view is set to match the focal length of our real camera used to capture real images (see Section 5.3.1). This is done to strongly improve depth estimation on real images by the final networks. For the rendering we consider steady data enhancements (Section 4.4.1). The light is placed normally distributed. Specular reflections are taken into account. The background of each image is kept transparent to be filled during training as a temporary data enhancement step (see Section 4.4.2). Every image is rendered in 16-fold target resolution (four times per image dimension [$\mathrm{width} \times \mathrm{height}$]) and shrunk with the Lanczos-Filter. This is done for anti-aliasing purposes without tampering with the image content (known as Ordered Grid Supersampling Anti Aliasing [73]). In the end an image for every label is rendered and stored on hard disk.

To handle the increased data demand for the steady data enhancements (Section 4.4.1), the icosaeder decomposition is performed with 400 divisions per edge. Hence, the final dataset consists of $n \approx 1.6 \cdot 10^6$ images. We use PNG as a loss-free compression format, resulting in an overall memory consumption of 40-80 GB, depending on the scenario.

4.4 Data Enhancement

We use *Domain Randomization* as our data enhancement strategy to increase the dataset for better representation of the problem space, for reduction of overfitting and to compensate for the reality gap. Based on our experience in computer vision we focus on sensory artifacts such as noise, blur and white balance, as well as environmental conditions like changing backgrounds, illumination geometry, luminosity, reflectance and illumination color for simulation. The rendering and image processing strategies for these enhancements are presented in detail subsequently. They can be seen in Fig. 4.

4.4.1 Steady Enhancements

Both illumination and reflection are hard baked into the object image prior to training. They are thus considered as a steady data enhancement (Fig. 4 column A).

- *Illumination* - The luminosity and the direction of light alter the appearance of an object, according to self-shadowing and reflecting properties of surfaces.

  To generalize this, both luminosity and light direction are randomly varied for different renderings of the object. In our industrial context, we can assume that light is always shining from a top-view perspective onto the object. Therefore, we apply a normally distributed random vector for the light source, with the mean pose being at the top central view downwards onto the object.
- *Reflection* - To simulate different reflections, the intensity value of the reflection probes in Unity3D are randomly set. The material properties (metallic & smoothness) are fixed as defined by our digitalized material database. Additionally, a cubic environment map is created by randomly orienting a randomly picked real image from the COCO dataset [74] which will be projected on the reflection probes. Using environment mapping [75], this creates an artificial reflection without any relation to a background (a different arbitrary background is induced later during training).

4.4.2 Temporary Enhancements

All of the following enhancements are added only temporarily during training by the generator (Section 4.1, Figs. 4B, 4C, 4D, and 4E).

- *Object color*- Imitating real material colors is difficult and even impossible, if objects are not yet produced. Color of the environmental illumination, as well as the white-balance of the camera make it even harder to render exactly what a real camera might sense. We therefore refrain from an accurate color model for synthesizing. Instead, for each image, we individually modify the color channels I_λ of the entire image by $I'_\lambda = min(I_\lambda \cdot s_\lambda \cdot i, 255)$. Here $\lambda \in \{r, g, b\}$ are the primary RGB color channels. $s_\lambda \in [0.75, 1.25]$ is a randomly ranged saturation modification per color channel. $i \in [0.5, 1.5]$ is a randomly ranged intensity modification over all channels. s_λ and i are uniformly distributed. These ranges are freely chosen and not yet investigated further. As we will know the color of the automotive paint or our mock-ups, we can assume camera white-balance and color temperature of light sources as the main factors influencing the perceived object color. Thus, we take care not to modify the color substantially, thereby still keeping the option for an object color descriptor. The enhancement result for object color can be seen in Fig. 4 B.

6. https://unity3d.com/

- *Background* - Adding an artificial background helps to be independent of prior knowledge of the environment. With constant color, color variations or noise as a background creating strong descriptors for segmentation, those must be avoided. As a common practice, we use photos as they have realistic textures, edges, noise and color. In principle, they cannot be considered synthetic. Yet, as they are largely available in different datasets in the internet, they are as easily to obtain as synthetic data. We use the COCO dataset [74] and increase the number of images. We shift color and brightness of the background images and randomly choose from different $90°$ orientations, mirroring on the principal axis and sub-sections of the input-image size. A result of adding background can be seen in Fig. 4 C.
- *Noise* - To increase robustness of the CNN, we simulate camera noise with additive white Gaussian noise (AWGN): $I'_\lambda = max(min(I_\lambda + z, 255), 0)$. Here I_λ is the intensity of a pixel for a primary RGB color channel $\lambda \in \{r, g, b\}$ with $0 \leq I_\lambda \leq 255$. z is set per pixel as a random number, normally distributed by the parameterized probability density function. Quantization noise is already present due to the nature of the rendered images. Other noise types are not considered as AWGN already covers the most relevant noise type for digital cameras. The result of adding noise can be seen in Fig. 4 D.
- *Blur* - During runtime we blur the real camera images, to compensate for real noise. To mimic this, all training images are blurred to compensate for the induced noise on the training data. Furthermore, we noticed sharp edges between rendered object and the added background. A similar observation was made by Manhardt *et al.* [26]. Such edges create a strong obstructive descriptor if not compensated. They are reduced by the blur as well. We apply a Gaussian blur of radius $r = 1$ using python package Pillow.[7] This blur can be seen in Fig. 4 E. It marks the final step before an image is used for training.

During training, all described data enhancements come together. The final image is normalized within a value range between $[-1, 1]$ and transferred to the CNN for training. Results of the full enhancement process with all its steps are shown in Fig. 4 for one random example on one pose.

4.5 Loss Metric

For Augmented Reality the superimposition of virtual and real objects is crucial. A strong error on camera position can result in a marginal error within this superimposition when observed on the image plane. As we encounter a regression problem, a mean square error is preferable over the usage of cross entropy. However, in such cases, position t and orientation R have to be weighted. R is a 3×3 rotation matrix and t the 3×1 translation vector. An error metric which represents the error on the object itself (like an augmentation) is able to weigh position t and orientation R equally as a pose.

7. https://pillow.readthedocs.io

The ADD error by Hinterstoisser *et al.* [44] is such an error model. It describes the error as an offset between the pose of the real object and the estimated pose of the virtual augmented counterpart. Here all vertices v, from a set of vertices V defining the digital 3D model $\mathcal{M}$, are transformed from the object coordinate system in O to the specific camera coordinate system C. This is done once using the pose of the ground truth camera $T_{gt} = (R_{gt}|t_{gt})$ defined by our labels and once by the camera pose $T_e = (R_e|t_e)$ estimated by our CNN. This creates two sets of vertices per pose sharing the same coordinate frame in which both can be compared. The offset between these sets is calculated by the mean euclidean difference representing the difference between the estimated and the real object pose. The ADD error is therefore defined as:

$$\mathrm{ADD} = \frac{1}{|\mathbf{V}|} \sum_{v \in \mathbf{V}} ||(R_{gt}v + t_{gt}) - (R_e v + t_e)|| \tag{2}$$

While the ADD error is used during training, for presentation and comparison we normalize it by the objects diameter d

$$\mathrm{ADD}^* = \frac{\mathrm{ADD}}{d} \tag{3}$$

Digital 3D models for industrial usage are often very detailed and therefore composed of huge vertex sets. The computation time of the ADD error increases with higher vertex counts. Using a subset of vertices could conserve time. But as parts of an industrial model have very high vertex density this creates imbalanced vertex distributions. For now we avoided the issue of subset creation, increasing the network's training time noticeably.

4.6 Demonstrator

We have developed a demonstrator for live execution of our approach on webcam images (see Fig. 1). For each scenario (represented by the respective CAD model), our CNN is automatically trained individually to become a unique tracker per object model, ready to determine camera poses for webcam images. The estimated poses are used to overlay virtual object models on the webcam images.

We used Unity3D as our rendering engine. Using Python, we implemented a local server to run the final CNN. The webcam images are sent to the server and the trained network estimates the pose. The resulting 6-dof pose is sent back and received by Unity3D and rendered accordingly, creating a live superimposed augmentation on the real object footage. This is handled in real time with 0.05s per image (20 FPS). A different setup could further improve this performance.

5 Evaluation

To identify how well our solution works, we pose three questions:

- How accurate is the estimated camera pose, considering different influencing factors and their magnitudes?

- How accurate is the estimated camera pose, considering the application on real world data – and are we able to overcome the reality gap?
- How well performs C.DOT compared to the works of Kehl *et al.* [62] and Rambach *et al.* [63].

5.1 Method

Here we highlight overall details of our evaluation method. We introduce the objects used for evaluation. Further do we state how we trained and selected each network.

5.1.1 Test Objects

We tested our approach on four objects: 1) one of our manufactured vehicles (VW Passat GTE), 2) a real mock-up used in our design process (VW e-up!), 3) a popular 3D print of a computer graphics model (Stanford Bunny) and 4) another 3D print of a computer graphics model (Stanford Dragon) (see Fig. 1). The first two objects are provided from real use cases (quality control and design). While the VW e-up! is based on a milling model data, the VW Passat GTE is manufactured based on its highly confidential construction data. For simple presentation a generalized, less complex model is used. All are available to us due to a digitalized development process. The last two objects are publicly available by the Stanford Computer Graphics Laboratory[8] and are chosen because of their universality. In Section 5.4 we compare our approach against the related work of Kehl *et al.* [62] and Rambach *et al.* [63] using further test objects and validation data of the LINEMOD dataset [44].

5.1.2 Training and Network Selections

We trained each network on the ADD error (Eq. (2)). Each training session took 4 days, resulting in $\sim$ 1000 epochs with $\sim$ 100 steps per epoch and 32 batches for each step. This end condition is based on experience rather than theoretical analysis. Due to our focus on synthetical data, we lack real world data for validation: During training, such real world data thus could not be considered, preventing an objective validation to identify when the CNN will perform best on real images. Shorter sessions led to bad results and longer ones to overfitting, regarding the usage on real world data. However, overfitting never occurred when applied on synthetic data, even for longer training sessions ($\sim$ 7 days). As an end condition a network which performed best during the 4 days of training was selected as the final *C.DOT* network. Best performance was measured by the lowest validation loss (ValLoss) when tested on the 20 percent synthetic validation data.

5.2 Accuracy on Synthetic Images

The evaluation data is a random disjoint set of 1002 synthetic images out of the 10 percent test data. The data enhancement is performed as described in Section 4.4 equivalent to the training and validation data. All images were not considered during training nor during validation ($M_{Training} \cap M_{Test} = \emptyset$). The same is true for the 442 background images injected. Each network per object is trained on the ADD error (Eq. (2)).

8. http://graphics.stanford.edu/data/3Dscanrep/

TABLE 3
Accuracy on Synthetic Data

	Object	VW Passat GTE	VW e-up!	Stanford Bunny	Stanford Dragon
ADD*	mean	0.047	0.138	0.049	0.053
	median	0.043	0.056	0.036	0.034
	StdDev	0.025	1.156	0.121	0.129
	10% [%]	97.30	85.93	96.91	95.71
	30% [%]	99.99	95.21	99.30	98.40
	d [mm]	5422.55	971.16	205.99	173.49

All accuracy results are given as ADD*, by estimating the ADD error for each image individually using the trained networks, normalized by the objects diameter (Eq. (3)). We have calculated the mean, median and standard deviation (StdDev) across all images. Rows 4 and 5 present the metric of Hinterstoisser *et al.* [44]: The percentage of correctly estimated poses is counted across all images. A pose is correct if

$$\text{ADD} \leq tr \cdot d, \tag{4}$$

where $tr \in \{0.1, 0.3\}$ is a 10 or 30 percent range threshold for the diameter $d = 2 \cdot r_{min}$ of the object $\mathcal{M}$ [44]. This metric was also used by Rambach *et al.* [63] in their work. The results are listed in Table 3. A value of 0.01 represents our desired goal of a pose estimation accuracy of 1 percent of the objects diameter.

The tests have produced considerably good results on all four objects. The VW e-up!'s network performs worst. The reason might be less contours and color differences. Despite its highly reflective surface and low color saturation, the VW Passat GTE's accuracy is as good as that of the Stanford Bunny and VW e-up!. The Stanford Dragon is slightly less accurate than the Stanford Bunny. The difference between it's mean and median indicate outliers, which are also represented by a higher standard deviation compared to the Stanford Bunny. An in depth analysis of the networks performances can be found in the upcoming sections.

Even though the network was tested on synthetic data, yet no result was sufficient to satisfy our high accuracy goals thus far. We emphasize that training intentionally for the application on synthetic data only would likely result in higher accuracies. But as our goal is the application on real world data, the possible performance on synthetic data is limited compared to a training designed specifically for such case.

5.2.1 Spatial Dependencies of the Error

In addition to the results shown in Table 3, we visualized the distribution of the ADD* error within a two dimensional heat map (see Fig. 5). The magnitude of the error for each tested image is mapped on a color scale and placed locally at the true image pose in relation to the object. The resulting point clouds are projected onto 2D, while points which would be projected onto the same pixel are shown as a

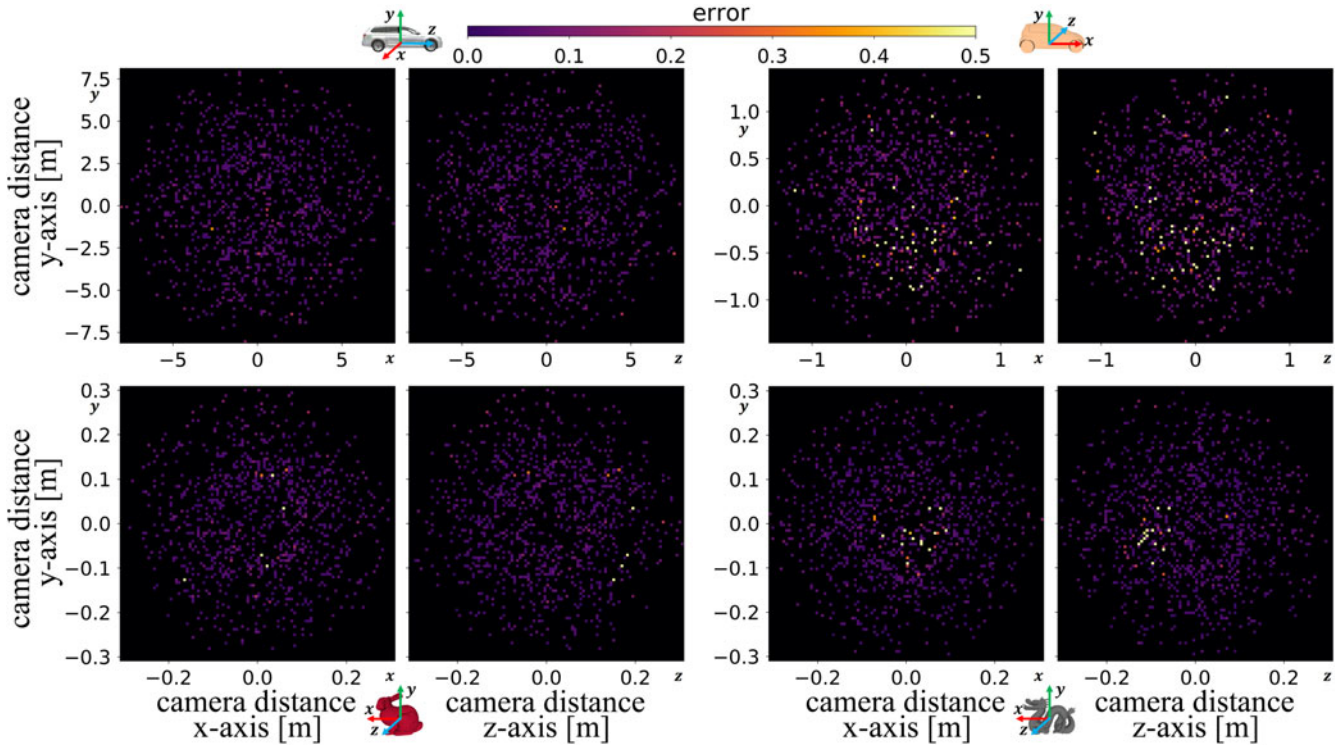

Fig. 5. Projected 3D distributions (x-y and z-y colormaps) of the accuracy of estimated camera poses for all four test objects.

mean value. This allows for investigation of spatial dependencies of the error.

For all objects, Fig. 5 shows an error between 0 and 0.1 homogeneously distributed over the whole space. High errors are more likely on poses very close or distant to an object. Close images include the object only partially, containing less information for an accurate estimation. Distant poses can result in stronger rotational errors and consolidate features in fewer pixels. The distribution for the VW Passat GTE shows almost no relevant outliers, giving reason to its very good performance in Table 3. In contrast, the VW e-up! indicates larger estimation errors compared to the others. Longer training sessions might reduce this effect. Yet strong outliers originate mostly from poses of its underbody, which possesses less features and is almost point-symmetric. This indicates topological influences, explaining its lower accuracy in Table 3. Of unknown cause are the strong outliers for the Stanford Dragon, representing close poses to its right side. We assume also topological influences to be the reason, as seen for the VW e-up!.

5.2.2 Investigating Influencing Factors

To evaluate the accuracy of the tracker under variations of influencing factors, we have run several robustness experiments. We use the networks and synthetic test data, as explained in the beginning of this section, with the same enhancement procedure. Yet, to produce the different influences, we manipulate the test data on various magnitudes as part of the respective enhancement step. The resulting error is presented as the mean over all test images. Those experiments create valuable insights, fostering explainability on the otherwise hidden functioning of each individual network.

Blur and noise - Blur, as a low-pass filter, can be considered as a direct countermeasure to noise. We therefore evaluated different magnitudes of blur and noise together (see Section 4.4 for details). The results are shown in Fig. 6. The figure indicates that the trained blur of filter radius $r = 1$ performs best. Lower values allow for unrealistically sharp edges and details. For higher values, too much of the relevant information is filtered out until the performance is limited at $r \geq 3$. Noise only has a strong negative impact for low blur values ($r < 1$).

Hue and brightness - As both factors target the same problem space of object color, we evaluate both together (see Fig. 7). We transform the color space from RGB to HSV in order to be able to manipulate hue and brightness (value) individually before presenting it as an RGB color image to the network for estimation purpose. All objects show individual behavior on the equally induced influences. This points out that each network adapts itself specifically to the training data. The network for the VW e-up! shows a prominent singular noisy behavior. In the supplementary material, which can be found on the Computer Society Digital Library,[9] we show that a strong correlation exists to the randomly injected background, which becomes apparent when excluding this injection step. This indicates a deficiency of the networks capability to segment the object from its background. The network's accuracy becomes highly sensitive to the random background, resulting in such noisy behavior. For the other objects a strong reduction of accuracy is given at low brightness, as features are less prominent. An increased brightness shows a much weaker impact instead. Those objects also show that a hue modification has at least a slight impact, which is especially strong for the very saturated Stanford Bunny. As intended the network

9. http://doi.ieeecomputersociety.org/10.1109/TVCG.2021.3089096

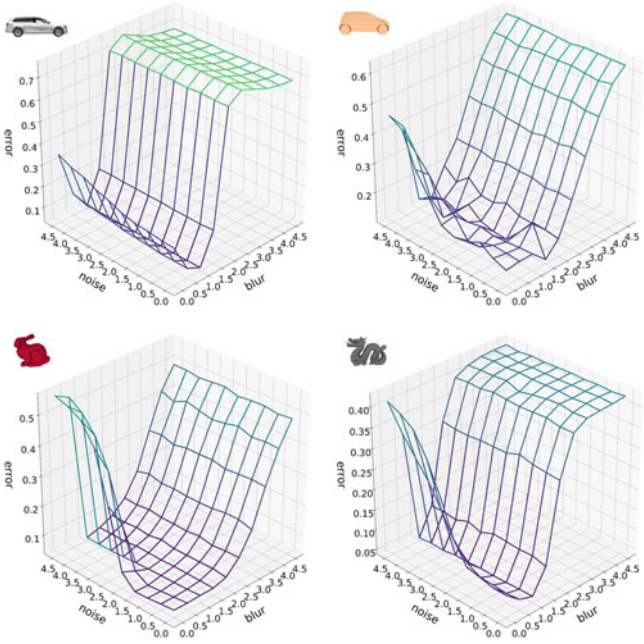

Fig. 6. Blur and noise influence for all four objects (by icons).

uses color as a descriptor for the Stanford Bunny, but does not have to rely on it as seen for the others. If desired the dependence on color could be further reduced by allowing full range color variations during data enhancement. Yet we varied the RGB color only within a limited range during training (see Section 4.4.2). In consequence a good performance is still given for the considered range, as noticeable for the Stanford Dragon and VW Passat GTE.

Occlusion - Even though occlusion is not highly relevant for our use cases, it may still occur. We thus evaluate how robust the network is against it. A partial occlusion should not just be a lateral slide-in, as for some of our images the object would not even be visible at all. As a result we create a systematic,

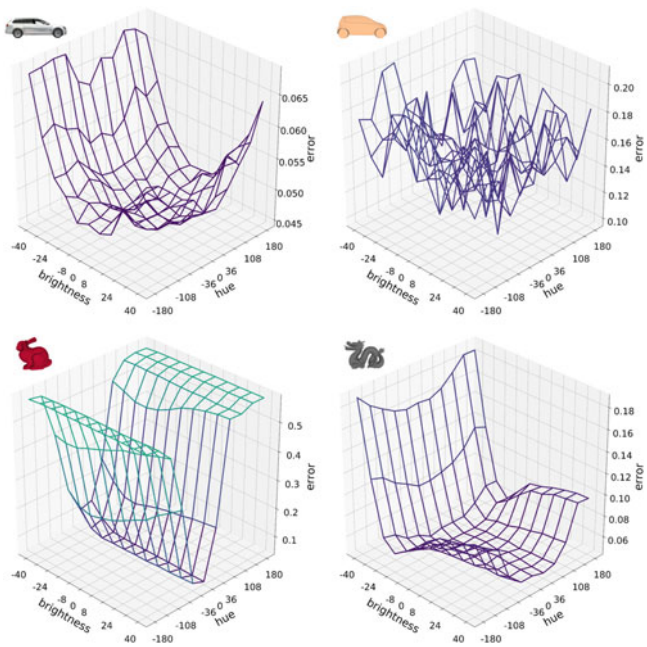

Fig. 7. Hue and brightness influence for all four objects (by icons).

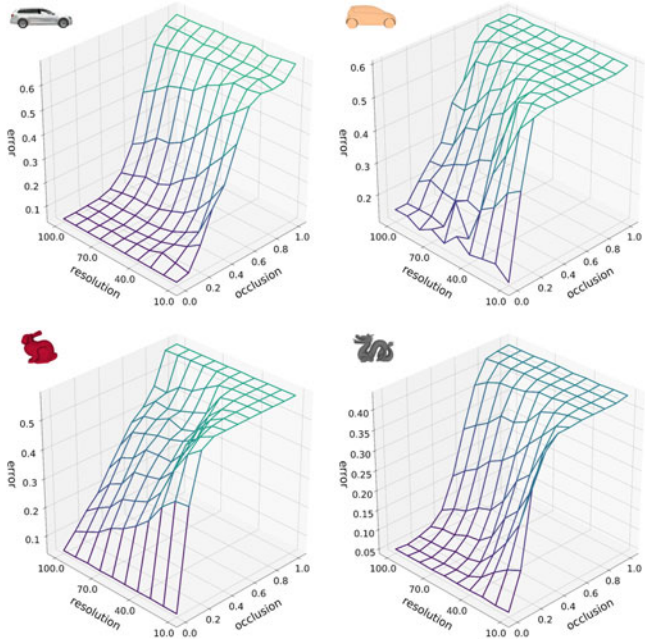

Fig. 8. Occlusion influence for all objects (by icons). Resolution is the potential size of an occlusion area in pixels. Occlusion is a percentage of the defined occlusion area which will be filled by a background image. See also Fig. 9.

but abstract test, where an arbitrary number of object features is occluded. We use a uniform grid of quads with variable size and resolution where each quad is textured by an image from our background dataset (see concept in Fig. 9). We evaluate the error regarding occlusion size and resolution of this grid. As expected, the accuracy is reduced with stronger occlusions in general. However, the accuracy does not cut off immediately (like for other estimators) but rather reduces gradually (see Fig. 8). The network is able to estimate continuously even for fragmented images.

5.3 Accuracy on Real Camera Image Sequences

To evaluate our approach on real data, we captured ground truth and applied it to our four networks per object trained on the ADD error (Eq. (2)) as described in Section 5.1.2.

5.3.1 *Image Capture and Ground Truth Acquisition*

We captured image sequences for each of the four trained objects using a Logitech webcam. The VW Passat GTE was captured outdoors, the other three objects were captured indoors. For the smaller 3D printed objects, we used a turntable with constant speed and a tripod for the camera. The

Fig. 9. Concept for the grid variations in two dimensions, resolution and occlusion size, which we use to evaluate occlusions. Eventually the white spaces are filled by a background similar to Fig. 4C.

TABLE 4
Accuracy on Real Camera Image Sequences for All of Our Four Objects Given as ADD^* (Eq. (3))

	Object	VW Passat GTE	VW e-up!	Stanford Bunny	Stanford Dragon
ADD^*	mean	0.048	0.068	0.048	0.093
	median	0.045	0.065	0.045	0.063
	StdDev	0.027	0.040	0.022	0.089
	10% [%]	96.33	94.56	97.37	68.00
	30% [%]	100.00	99.78	100.00	94.50
	diameter d [mm]	5422.55	971.16	205.99	173.49
	ValLoss [mm]	250	95.79	9.65	11.40
	epochs	740	685	405	850

Rows 10% and 30% represent the percentage of correct poses (Eq. (4)) for diameter d. ValLoss describes the validation loss of the final network, trained on the ADD error (Eq. (2)).

TABLE 5
Comparison Between Two Networks for the VW Passat GTE Trained (A) *With* and (B) *Without* Data Enhancement Steps, When Applied to the Same Ground Truth

	Data Enhancement	(A) with	(B) without
ADD^*	mean	0.090	0.141
	median	0.080	0.086
	StdDev	0.051	0.128
	10% [%]	79.18	55.30
	30% [%]	98.14	87.55
	ValLoss [mm]	350	280
	epochs	470	460

For (B) a network with the smallest validation loss (ValLoss) was chosen to be compared with a network (A) with a similar number of epochs and closest ValLoss.

other two were filmed by hand. For practical reasons, we did not aim for obtaining uniformly distributed images from all around an object but rather a single movement sequence. Therefore, the results are not comprehensive when compared to tests on uniformly distributed synthetic images (as done in Section 5.2). The automatic white balance and auto focus were always active, dynamically changing color and focus. For the indoor scenes, the light flickers due to frequency mashups of the room light and the camera framerate.

Each captured image was processed to compensate for distortion. The camera was calibrated once to retrieve distortion coefficients and to approximate the focal length for acquisition of ground truth data. The focal length also defines the FOV of the virtual camera during rendering of the training images to specifically imitate the utilized real camera (see also Section 4.3).

For all four image sequences we created ground truth measurements using the VisionLib[10] object tracking system. This tracker is one of the hand-crafted blackbox systems we use for AR purposes (Section 1.2). It was carefully tuned before being used here as a ground truth reference. As its accuracy turned out to be more than a magnitude higher than that of our estimator, it is validly applicable. Our ground truth dataset is made publicly available as part of this publication [79].

5.3.2 Comparison Against the Ground Truth

For each image sequence, we let our corresponding networks estimate the pose of the camera with respect to the object and compared the estimations to the ground truth. The results are shown in Table 4. Rows 1–3 show statistics (mean, median and standard deviation) based on the ADD^* error (Eq. (3)). Rows 4 and 5 show the percentage of correctly estimated poses across all images (Eq. (4)). We visualized all results using our demonstrator fed with all image sequences as an offline input source. The estimated poses are rendered as superimposed augmentations. A selection of those images are shown in Fig. 1. The full sets are published as supplementary material,[9] available online.

10. https://visionlib.com/

For all objects the hit rate for the 30 percent and even for the 10 percent range is always very close to 100 percent. Only exception is the Stanford Dragon which performs worst in both cases, also indicated by a higher standard deviation. Considering the fact that the listed accuracies are all achieved using only synthetical RGB images for training to automatically create an individual tracker, this is a remarkably good and outstanding result. The results of all objects are often similar or even better than the results from tests on synthetical data (see Section 5.2). This can be argued by the *Domain Randomization* lowering the in general achievable performance on synthetical data. The reason for the Stanford Dragon performing worse than the others may be due to an almost indifferent appearance to the background. Another reason may be strong focus shifts in the image sequence. Such shifts alter the *depth of field* or allow for *focus breathing* which can have a slight impact on focal length. These effects can lower the ground truth acquisition and the network estimation capability likewise. To compensate for that further research may consider focus variations or variations in the depth of field or focal length as further data enhancement strategies during training.

The network for the VW Passat GTE achieved the second best result of all. Being our most relevant, but also most complex object (due to strong reflections and specular lights), the results are also very good. However, considering that the objects diameter is much larger than the others, the absolute error is also much larger. A different strategy for reflections, another loss metric or a better end condition for training may improve these results.

That said, we do not yet achieve our initial set goal of ≤ 0.01 accuracy for any of the objects. Thus, our proposed system is not yet ready for most industrial AR use cases. Yet, it will already be sufficient for use cases demanding ≥ 0.1 accuracy or for supporting our existing solutions (see also Section 6.3).

5.3.3 Importance of Data Enhancement for Real Applications

To determine whether the *Domain Randomization* has a positive effect on bridging the reality gap, we trained two networks for the VW Passat GTE, one with and one without data

TABLE 6
Comparison of Our *C.DOT* Approach to Related Work of Kehl *et al.* [62] and the Variants by Rambach *et al.* [63] of Using Gray-Scaled Pencil Filter Images (Pen) or RGB Color Images (RGB)

	Object	Ape	Benchv.	Cam	Can	Cat	Driller	Duck	Eggbox	Glue	Holepu.	Iron	Lamp	Phone	ϕ
10%	[62]	0	0.18	0.41	1.35	0.51	2.58	0	8.9	0	0.3	8.86	8.20	0.18	2.42
[%]	[63] (pen)	**4.37**	21.74	**1.25**	2.09	**2.54**	12.46	**4.78**	1.43	**7.38**	**3.88**	**39.18**	**27.35**	5.39	10.29
	[63] (RGB)	0	5.76	0.16	0	0	**12.97**	1.83	0	0.49	2.26	**39.18**	0.32	0	4.84
	C.DOT	0.56	**38.84**	0.40	**20.48**	1.35	0.51	2.71	**15.96**	3.19	1.85	29.69	3.34	**22.84**	**10.90**
	ABC	1.05	**40.75**	**1.58**	**32.94**	1.10	2.19	**6.46**	**15.96**	**8.36**	2.91	**46.26**	7.49	**22.84**	**14.61**
30%	[62]	5.62	2.07	34.52	61.43	**36.87**	**56.01**	5.56	24.61	14.18	18.23	59.26	57.64	35.55	31.65
[%]	[63] (pen)	27.93	61.36	7.58	27.78	24.53	49.87	**29.05**	13.57	41.92	24.19	87.66	71.66	30.67	38.29
	[63] (RGB)	0.08	36.73	1.5	0.33	1.87	54.33	17.63	0	7.05	24.19	**89.48**	5.95	3.94	18.69
	C.DOT	**40.48**	**89.95**	**39.80**	**73.49**	25.61	53.20	24.40	**74.47**	**62.04**	**29.99**	87.76	**72.21**	**62.83**	**56.63**
	ABC	**46.76**	**93.99**	**60.12**	**86.03**	33.84	**70.12**	**54.63**	**74.47**	**73.03**	**38.39**	**97.04**	**74.98**	**62.83**	**66.63**
C.DOT	mean	0.52	0.17	0.46	0.25	0.53	0.33	0.82	0.23	0.30	0.58	0.18	0.27	0.36	0.38
	median	0.35	0.12	0.34	0.19	0.42	0.29	0.80	0.19	0.26	0.46	0.13	0.24	0.21	0.31
	StdDev	0.48	0.17	0.35	0.20	0.41	0.17	0.57	0.16	0.22	0.44	0.17	0.15	0.36	0.30
	ValLoss [mm]	35.97	26.11	22.13	30.87	44.73	32.02	28.10	34.90	31.13	41.84	30.81	34.18	24.74	32.12
ABC	mean	0.35	0.14	0.37	0.18	0.39	0.27	0.46	0.23	0.24	0.54	0.12	0.24	0.36	0.30
	median	0.31	0.12	0.26	0.13	0.37	0.26	0.27	0.19	0.22	0.38	0.11	0.21	0.21	0.23
	StdDev	0.23	0.10	0.30	0.15	0.20	0.11	0.40	0.16	0.14	0.67	0.08	0.17	0.36	0.24
	ValLoss [mm]	60.16	29.69	75.35	42.51	45.35	46.03	90.09	34.90	34.83	212.03	32.46	49.13	24.74	59.79
	diameter d [mm]	142.1	330.9	316.6	284.2	185.0	318.8	155.7	197.4	193.1	196.1	317.2	316.6	254.3	246.8

The comparison is done for 13 objects of the LINEMOD dataset [44]. Rows 10% *and* 30% *represent the percentage of correct poses (Eq. (4)) for the individual diameter d of each object. All our networks are trained on the ADD error (Eq. (2)). Orange highlights the worst result of a column and* ***blue*** *the best. The ϕ column shows the average of each row.* Approximated Best Case (ABC) *shows what C.DOT could achieve in an approximated case of selecting the best possible network. It is marked with* ***bold*** *for cases where it exceeds the other results. To show the difference between C.DOT and its ABC-Analysis, we added the* ADD* *error's mean, median, standard deviation (StdDev) and the validation loss (ValLoss) in millimeters.*

enhancement and compared them to each other. We trained one network (B) just using our realistic looking model (see Fig. 4 A), *without* adding any of the data enhancement steps of Section 4.4. We compare it to our network (A) trained with all data enhancement steps, but chosen by a similar number of epochs and validation loss. The results are shown in Table 5. When applied to our real ground truth the network (A) performs much better than network (B). Without *Domain Randomization* the network (B) is not able to manage variations in the input data, resulting in much higher standard deviation than (A). The lower validation loss of (B) also indicates possible overfitting to the synthetic data, which is more likely if data enhancement is not utilized. This positive effect of *Domain Randomization* is further substantiated when comparing network (B) with the network of the VW Passat GTE of Table 4. Even though both have a similar low validation loss, the one using data enhancements is much better suited for real input resulting in higher accuracy and lower standard deviation.

In conclusion, we are able to bridge the reality gap on real data merely using *strongly enhanced* synthetic data. This results in accuracies high enough for simple superimpositions – yet, insufficient for the high demands on accuracy as posed by many industrial use cases.

5.4 Comparison Against Related Work

We compared C.DOT against the work by Rambach *et al.* [63] (two variants) and the work by Kehl *et al.* [62], based on the LINEMOD dataset of Hinterstoisser *et al.* [44]. We trained 13 different networks based on 13 digital objects of the dataset: Ape, Benchvise, Cam, Can, Cat, Driller, Duck, Eggbox, Glue, Holepuncher, Iron, Lamp and Phone. We slightly adjusted our data space (see Section 4.2) to work with the dataset. Each network is trained on the ADD error (Eq. (2)). For every epoch, the trained network is saved and tested on our 20 percent synthetical validation data. We selected a network which had the smallest validation loss (ValLoss).

All results are provided in Table 6 and are given as a hit rate in percentage for the 10 and 30 percent range of the diameter for each object (Eq. (4)). Additionally we present the estimation mean, median and standard deviation for each image sequence per object given by ADD* (Eq. (3)), as well as the validation loss for the finally selected network. In general our approach shows remarkably good results, when compared to the best related work of each column: $\Delta = \text{C.DOT} - \text{best}$. For the 30 percent range we outperform the related work in 8 of 13 cases: $\Delta(Cam) : +5.22\%$ up to $\Delta(Eggbox) : +49.86\%$ more accuracy. For 4 cases (Driller Duck, Iron, Lamp) C.DOT performs on a similar level (between $\Delta(Lamp) : +0.55\%$ and $\Delta(Duck) : -4.65\%$), whereas $\Delta(Cat) : -11.26\%$ performs the worst. The 10 percent range performance is more diverse. Here we outperform the related work in 4 of 13 cases, but in those cases noticeably strong: $\Delta(Eggbox) : +14.53\%$ up to $\Delta(Can) : +18.39\%$. For 6 cases C.DOT performs on a similar level: $\Delta(Cam) : -0.85\%$ up to $\Delta(Glue) : -4.19\%$ less accuracy. Driller, Iron and Lamp are exceptions, where the related work of Rambach *et al.* [63] performs much stronger ($\Delta(Driller) : -12.46\%$ up to $\Delta(Lamp) : -24.01\%$). This makes C.DOT at average $\Delta(\phi) : +18.34\%$ better for the 30 percent range, but only $\Delta(\phi) : +0.61\%$ better in the 10 percent range.

The reason for bad performances in the 10 percent range on objects like Ape, Cam, Cat, Duck, Glue and Holepuncher for *all* the listed approaches can only be assumed so far. They might be due to strong discrepancies between the digitally reconstructed objects in the dataset compared to their real appearance in the images. Another reason, particularly limiting the estimation capability of C.DOT might be the fact that those objects span only a few pixels within our input images for the defined resolution of 299×299 pixels (see Eq. (1)). This is primarily true for the larger viewing distances that are used by the LINEMOD dataset. This may also explain why tests on our own ground truth (see Section 5.3) performed much better: our objects are larger and the viewing distance was defined for a much closer perspective, providing more information for estimation within the input images. Another explanation is that our own digital models are of much higher visual fidelity (very close to the real objects) than the LINEMOD models.

Approximated Best Case - Finding an end condition for training when using only synthetical data is difficult. The reason is that no real data is present in the training, validation and test sets. The validation loss gives no clear indication whether a network will be best suited for real data. For further investigation we use an approximated best case analysis to find a possible upper bound of performance. We use this to show a clear representation of the problem and for opening up the discussion. While validating a network on 20 percent synthetical validation data every epoch, we additionally validate every fifth on the LINEMOD data too. This creates a pure theoretical and intentionally biased end condition, which we use to select a second network as an approximated best case (ABC). It is an experimental case were the network is performing best possible on the test data. Note that for ABC the validation and test data are *purposely not two disjoint sets*. Yet training and validation data are still disjoint, as one is synthetical and the other is real. We want to emphasize that this process is for additional insights only and is by no means an alternative to our regular C.DOT approach presented earlier.

The results for ABC are shown in Table 6 to be compared with C.DOT. The C.DOT networks show what can actually be achieved in a regular practical usage. The ABC networks indicate a possible upper limit, an overall potential of what C.DOT might be able to achieve at best, in case of finding a well suited end condition or if real world validation data would be available. Even though C.DOT is often close to ABC's performance, there is still potential for improvement. If we would be able to push it to its limits, we would likely outperform the related work more often (indicated in **bold**). But even then, this would not apply to all cases, like in the 10 percent range. Comparing ABC's performance to C.DOT gives us a metric of how close C.DOT is to such approximated best case. The discrepancy between end conditions becomes especially apparent when looking at the ValLoss in Table 6. Against expectations the ValLoss is always much higher for ABC than for C.DOT, whereas ABC is always much better suited for usage on real data (e.g., as strongly noticeable in the "Cam" or "Duck" cases). All this indicates that C.DOT is overfitting to the synthetical data, and that a minimum ValLoss is a barely sufficient end condition for the training on synthetical data.

6 Conclusion & Future Work

We have demonstrated that an object tracker for AR can be realized using a CNN that is trained only on synthetic RGB data, thereby creating a purely data driven approach without the necessity of manually configuring obscure parameters. Following a Tracking-by-Detection approach, our tracker estimates on each image separately. Processing always the full image creates a holistic approach.

Our C.DOT approach is based on a well defined network architecture and data structure for training with strong data enhancement strategies (*Domain Randomization*). For this reason we focused on different rendering aspects (i.e., reflections) and their randomization. This allows bridging the reality gap more easily since the network will handle real features as just another variation of the synthetic domain. As a result, the estimation is quite independent of sensor artifacts, making it feasible to handle automatic corrections such as white balancing or auto-focus during tracking. To encode our 6-dof labels and to define our poses for rendering, we use a rather uncommon relative description of the camera orientation. This facilitates the training of the network and its estimation capabilities.

We performed several robustness tests to assess our approach: We tested for spatial dependencies, behavior on induced sensory artifacts and occlusion. We also evaluated how well the tracker performs on real data and compared it to related work. The results show, that despite using only synthetic training data, we are able to outperform similar approaches.

6.1 Critical Discussion

For the estimation method, we utilized the ADD error metric (Eq. (2)), which performs remarkably great. However it lacks comparability between objects of distinct size, creating disadvantages for larger objects like cars. Besides, the metric was found to be very computational demanding for industrial models, opening the topic of finding reasonable subsets of vertices.

Furthermore, potential dependencies between estimation accuracy, network architecture, number of training images and utilized strategies for bridging the reality gap are not yet unraveled. We tested only the practical effect of our *Domain Randomization*. One of the other dependencies could have had an even stronger positive impact on the outcome of our approach when compared to the related work. Thus, further investigations are needed.

A strong constraint of using merely synthetic data is the lack of an end condition during training. Such end condition has a strong influence on the estimation accuracy, due to the occurance of fluctuations in performance between epochs and the danger of overfitting to the synthetic data. However, in many industrial applications real data is not available and the prospective performance on real data is thus unknown during training. This leaves only experience as an end condition or improved realistic rendering as a solution against overfitting. But if ground truth is present, it will likely boost performance considerably just by providing a better end condition.

We have tested our approach on our own synthetic and real data and compared it to the related work, showing great performance and indicating that the reality gap could

indeed be overcome by outperforming other approaches so far. While a perfect superimposition satisfying industrial requirements cannot be realized yet, the results are very promising and can already be used for some simpler AR use cases. An alternative to AR applications would be the support of robotic use cases, like identification of the heading of an object (as part of sensor fusion) or as assistance on course grabbing tasks like vacuum gripping.

6.2 Future Work

We have proposed and implemented a CNN-based system, showing that the proposed deeper architecture constitutes a viable approach. Further research may focus on temporal use of the tracking data by utilizing recurrent networks or sequential inputs as done by Garon *et al*. [57]. A tracking over time on consecutive images could easily be built on top of our tracker. This would allow for pose filtering or smoothing over time, reducing the jitter. One could improve the accuracy of our tracker by using pose refinement methods proposed by Manhardt *et al.* [26]. Moreover, it is possible to improve the result by creating a specialized network architecture layout for pose estimation or by optimizing the hyperparameters of the network.

Future work could focus on testing a unitless, distance independent metric. Such metric could account better for different superimposition requirements among different use cases and allow for standardized comparison between approaches.

We currently focus on tracking solutions for mobile devices. Not limiting our approach to such devices with limited capabilities, general industrial setups become possible as well (e.g., estimations on conveyor belts). In such cases, fusion of multiple estimations from different cameras or perspectives can create a further boost to object pose estimation accuracy.

Bridging the reality gap to increase accuracy even more still remains an open question requiring further research. Other ways to overcome the gap may be the use of Style-transfer concepts [76] as a domain adaptation strategy or Generative Adversarial Networks [77] to generate more realistic images.

More importantly, further research needs to investigate data deployment strategies to enable good end conditions when working with synthetic data. It could be a viable option to rely on our proposed *Domain Randomization* during training, whereas for validation a set of highly photorealistic renderings could be used.

6.3 Conclusion

At this stage of development, the strongest argument in favor of our C.DOT approach is that it provides an improved alternative for the initialization procedure for classically engineered trackers that are currently used in industrial applications (e.g., [23], [78]). This is especially beneficial as our approach works as a Tracking-by-Detection system, estimating the camera pose separately on every image of a sequence without considering previous frames. Furthermore, our approach is robust against environmental or sensory influences while still yielding continuous results for such interferences. As a downside, a huge time investment for rendering and training has to be considered, always specializing the overall usage to each particular object. This however, works well in predictable, planned industrial use cases which we aim for. Most importantly, the process can be completely automated, getting rid of laborious calibration and tuning efforts.

References

[1] M. Y.-C. Yim, S.-C. Chu, and P. L. Sauer, "Is augmented reality technology an effective tool for e-commerce? An interactivity and vividness perspective," *J. Interactive Marketing*, vol. 39, pp. 89–103, 2017.

[2] W. Huber, *Industrie 4.0 in Der Automobilproduktion: Ein Praxisbuch*. Wiesbaden, Germany: Springer, 2016.

[3] W. Schreiber and P. Zimmermann, *Virtuelle Techniken im Industriellen Umfeld: Das Verbundprojekt AVILUS – Technologien und Anwendungen*. Berlin, Germany: Springer, 2011.

[4] W. Schreiber, "ARVIDA – BmBF Project Angewandte Referenzarchitektur für virtuelle Dienste und Anwendungen," arvida.de/en. [Online]. Available: https://www.ARVIDA.de/

[5] G. Klinker, A. H. Dutoit, M. Bauer, J. Bayer, V. Novak, and D. Matzke, "Fata Morgana - A presentation system for product design," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality*, 2002, pp. 76–85.

[6] D. Reiners, D. Stricker, G. Klinker, and S. Müller, "Augmented reality for construction tasks: Doorlock assembly," in *Proc. IEEE/ACM Int. Workshop Augmented Reality*, 1998, pp. 31–46.

[7] S. Nölle and G. Klinker, "Augmented reality as a comparison tool in automotive industry," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality.*, 2006, pp. 249–250.

[8] O. Wasenmüller, M. Meyer, and D. Stricker, "Augmented reality 3D discrepancy check in industrial applications," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2016, pp. 125–134.

[9] F. Echtler *et al.*, "The intelligent welding gun: Augmented reality for experimental vehicle construction," in *Virtual and Augmented Reality Applications in Manufacturing*, S. Ong and A. Nee, Eds. Berlin, Germany: Springer Verlag, 2003, ch. 17.

[10] G. Klinker, H. Najafi, T. Sielhorst, and F. Sturm, "FixIt: An approach towards assisting workers in diagnosing machine malfunctions," in *Proc. IUI-CADUI'04 Workshop Exploring Des. Eng. Mixed Reality Syst.*, vol. 91, 2004. [Online]. Available: http://ceur-ws.org/Vol-91/paperE1.pdf

[11] T. Engelke, J. Keil, P. Rojtberg, F. Wientapper, M. Schmitt, and U. Bockholt, "Content first: A concept for industrial augmented reality maintenance applications using mobile devices," in *Proc. ACM Multimedia Syst. Conf.*, 2015, pp. 105–111.

[12] C. Resch, P. Keitler, and G. Klinker, "Sticky projections-A model-based approach to interactive shader lamps tracking," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 3, pp. 1291–1301, Mar. 2016.

[13] G. Klinker, O. Creighton, A. H. Dutoit, R. Kobylinski, C. Vilsmeier, and B. Brügge, "Augmented maintenance of powerplants: A prototyping case study of a mobile AR system," in *Proc. Int. Symp. Augmented Reality*, 2001, pp. 124–136.

[14] K. Pentenrieder, C. Bade, F. Doil, and P. Meier, "Augmented reality-based factory planning - An application tailored to industrial needs," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality*, 2007, pp. 31–42.

[15] M. Olbrich, H. Wuest, P. Riess, and U. Bockholt, "Augmented reality pipe layout planning in the shipbuilding industry," in *Proc. IEEE Int. Symp. Mixed Augmented Reality.*, 2011, pp. 269–270.

[16] B. Schwerdtfeger and G. Klinker, "Supporting order picking with augmented reality," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality.*, 2008, pp. 91–94.

[17] M. Esengün and G. Ince, "*The role of augmented reality in the age of industry 4.0*," *Industry 4.0: Managing The Digital Transformation*, Cham, Germany: Springer International Publishing, 2018, pp. 201–215, doi: 10.1007/978-3-319-57870-5_12.

[18] G. Welch and E. Foxlin, "Motion tracking: No silver bullet, but a respectable arsenal," *IEEE Comput. Graphics Appl.*, vol. 22, no. 6, pp. 24–38, Nov./Dec. 2002.

[19] D. Stanimirovic, N. Damasky, S. Webel, D. Koriath, A. Spillner, and D. Kurz, "[Poster] a mobile augmented reality system to assist auto mechanics," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality.*, 2014, pp. 305–306.

[20] H. Kato and M. Billinghurst, "Marker tracking and HMD calibration for a video-based augmented reality conferencing system," in *Proc. IEEE/ACM Int. Workshop Augmented Reality.*, 1999, pp. 85–94.

[21] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

[22] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality.*, 2007, pp. 1–10.

[23] H. Wuest and D. Stricker, "Tracking of industrial objects by using CAD models," *J. Virt. Reality Broadcast.*, vol. 4, no. 1, 2007. [Online]. Available: http://nbn-resolving.de/urn:nbn:de:0009-6-11595, doi: 10.20385/1860-2037/4.2007.1.

[24] G. Gay-Bellile, S. Bourgeois, M. Tamaazousti, S. Naudet-Collette, and S. Knodel, "A mobile markerless augmented reality system for the automotive field," in *Proc. IEEE ISMAR Workshop Tracking Methods Appl.*, 2012.

[25] K. K. Thiel, E. Jundt, and G. Klinker, "[Poster] automated evaluation and configuration of object tracking for augmented reality," in *Proc. IEEE Int. Symp. Mixed Augmented Reality*, 2017, pp. 132–134.

[26] F. Manhardt, W. Kehl, N. Navab, and F. Tombari, "Deep model-based 6D pose refinement in RGB," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 833–849.

[27] N. Jacobi, P. Husbands, and I. Harvey, "Noise and the reality gap: The use of simulation in evolutionary robotics," in *Proc. Eur. Conf. Adv. Artif. Life*, 1995, pp. 704–720.

[28] 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.*, 2017, pp. 23–30.

[29] F. Zhou, H. B.-L. Duh, and M. Billinghurst, "Trends in augmented reality tracking, interaction and display: A review of ten years of ismar," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality*, 2008, pp. 193–202.

[30] K. Kim, M. Billinghurst, G. Bruder, H. B.-L. Duh, and G. F. Welch, "Revisiting trends in augmented reality research: A review of the 2nd decade of ISMAR (2008–2017)," *IEEE Trans. Vis. Comput. Graphics*, vol. 24, no. 11, pp. 2947–2962, Nov. 2018.

[31] E. Marchand, H. Uchiyama, and F. Spindler, "Pose estimation for augmented reality: A hands-on survey," *IEEE Trans. Vis. Comput. Graphics*, vol. 22, no. 12, pp. 2633–2651, Dec. 2016.

[32] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 8, pp. 930–943, Aug. 2003.

[33] L. Quan and Z. Lan, "Linear N-point camera pose determination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 8, pp. 774–780, Aug. 1999.

[34] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPnP: An accurate O (n) solution to the PnP problem," *Int. J. Comput. Vis.*, vol. 81, no. 2, pp. 155–166, 2009.

[35] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[36] P. J. Besl and N. D. McKay, "A method for registration of 3D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992.

[37] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. Int. Conf. Comput. Vis.*, 1999, pp. 1150–1157.

[38] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vis. Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[39] S. Hinterstoisser *et al.*, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Proc. Int. Conf. Comput. Vis.*, 2011, pp. 858–865.

[40] A. Mallios, P. Ridao, D. Ribas, and E. Hernández, "Scan matching SLAM in underwater environments," *Auton. Robots*, vol. 36, no. 3, pp. 181–198, 2014.

[41] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[42] J. M. Wong *et al.*, "SegICP: Integrated deep semantic segmentation and pose estimation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5784–5789.

[43] M. Rad and V. Lepetit, "BB8: A scalable, accurate, robust to partial occlusion method for predicting the 3D poses of challenging objects without using depth," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3848–3856.

[44] S. Hinterstoisser *et al.*, "Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes," in *Proc. Asian Conf. Comput. Vis.*, 2013, pp. 548–562.

[45] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[46] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[47] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[48] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[49] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[50] P. Ondrúška and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 3361–3367.

[51] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," in *Proc. Robot. Sci. Syst.*, 2018, doi: 10.15607/RSS.2018.XIV.019.

[52] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A convolutional network for real-time 6-DoF camera relocalization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2938–2946.

[53] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3D pose estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3109–3118.

[54] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2686–2694.

[55] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes challenge 2012 (VOC2012) results." 2012. [Online]. Available: http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html

[56] J. Papon and M. Schoeler, "Semantic pose using deep networks trained on synthetic RGB-D," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 774–782.

[57] M. Garon and J. Lalonde, "Deep 6-DOF tracking," *IEEE Trans. Vis. Comput. Graphics*, vol. 23, no. 11, pp. 2410–2418, Nov. 2017.

[58] J. Xiao, A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using SfM and object labels," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1625–1632.

[59] H. Wang, S. Sridhar, J. Huang, J. P. C. Valentin, S. Song, and L. Guibas, "Normalized object coordinate space for category-level 6D object pose and size estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2637–2646.

[60] D. J. Tan, F. Tombari, S. Ilic, and N. Navab, "A versatile learning-based 3D temporal tracker: Scalable, robust, online," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 693–701.

[61] M. Garon, D. Laurendeau, and J.-F. Lalonde, "A framework for evaluating 6-DoF object trackers," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 608–623.

[62] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-based 3D detection and 6D pose estimation great again," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1530–1538.

[63] J. Rambach, C. Deng, A. Pagani, and D. Stricker, "Learning 6DoF object poses from synthetic single channel images," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality.*, 2018, pp. 164–169.

[64] Y. Su, J. R. Rambach, N. Minaskan, P. Lesur, A. Pagani, and D. Stricker, "Deep multi-state object pose estimation for augmented reality assembly," in *Proc. IEEE/ACM Int. Symp. Mixed Augmented Reality.*, 2019, pp. 222–227.

[65] G. Csurka, *A Comprehensive Survey on Domain Adaptation for Visual Applications*. Berlin, Germany: Springer, 2017.

[66] D. M. Montserrat, J. Chen, Q. Lin, J. P. Allebach, and E. J. Delp, "Multi-view matching network for 6D pose estimation," *CoRR*, vol. abs/1911.12330, 2019.

[67] X. Ren *et al.*, "Domain randomization for active pose estimation," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 7228–7234.
[68] R. Khirodkar, D. Yoo, and K. M. Kitani, "Domain randomization for scene-specific car detection and pose estimation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2019, pp. 1932–1940.
[69] 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. 1082–10828.
[70] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 4278–4284.
[71] S. Hinterstoißer, V. Lepetit, P. Wohlhart, and K. Konolige, "On pre-trained image features and synthetic images for deep learning," in *Proc. Eur. Conf. Comput. Vis. Workshops*, 2019, pp. 682–697.
[72] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.
[73] D. Carreker, *The Game Developer's Dictionary: A Multidisciplinary Lexicon for Professionals and Students*, 1st ed. Boston, MA, USA: Course Technology Press, 2012.
[74] T. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.
[75] N. Greene, "Environment mapping and other applications of world projections," *IEEE Comput. Graphics Appl.*, vol. 6, no. 11, pp. 21–29, Nov. 1986.
[76] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *Journal Vision*, 2016, doi: 10.1167/16.12.326.
[77] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *Proc. Int. Conf. Learn. Representations.*, 2018. [Online]. Available: https://openreview.net/forum?id=Hk99zCeAb
[78] J. Rambach, A. Pagani, M. Schneider, O. Artemenko, and D. Stricker, "6DoF object tracking based on 3D Scans for augmented reality remote live support," *Computers*, vol. 7, no. 1, 2018, Art. no. 6.
[79] K. K. Thiel, "C.DOT ground truth dataset" *Technical Univ. Munich*, 2021. [Online]. Available: https://mediatum.ub.tum.de/1614575, http://doi.ieeecomputersociety.org/10.1109/TVCG.2021.3089096, doi: 10.14459/2021mp1614575.001.

Kevin Kennard Thiel received the MSc degree in visual computing from the Otto-von-Guericke University Magdeburg, Magdeburg, Germany, in 2014 for utilizing image-based rendering for spatial augmented reality. He is currently working toward the external PhD degree with the Technical University of Munich, Munich, Germany researching on automated tracking generation, evaluation and configuration for industrial applications. He is employed with the Volkswagen Group in Wolfsburg as an expert for creating novel augmented & virtual reality systems as well as a data analyst fostering groupwide digitalization.

Florian Naumann received the MSc degree in computer science from the University of Applied Sciences of Central Hesse, Giessen, Germany, in 2018. He is currently working toward the PhD degree and data scientist at Volkswagen Group, Wolfsburg, Germany. His research interests focus on data analysis and machine learning in the area of image processing, natural language processing, and time series processing.

Eduard Jundt received the degree in mathematics and computer science from the University of Hanover, Hanover, Germany, in 2001. Afterwards, he worked 18 years with the Research Department, Volkswagen Group in Wolfsburg. His research interests focuses on visualization, perception, virtual reality, augmented reality, and especially on spatial augmented reality. Since 2019, he leads a team within R&D at Volkswagen Commercial Vehicles in Wolfsburg focusing on virtual technologies.

Stephan Günnemann received the PhD degree in computer science from the RWTH Aachen University, Aachen, Germany, in 2012. Afterwards, he worked with Carnegie Mellon University first as a postdoctoral fellow and as a senior researcher later on. He researched for the Simon Fraser University in Canada as well as for the Research & Technology Center of the Siemens AG. In 2015, he founded a DFG Emmy-Noether Research Group, Computer Science Department, Technical University of Munich (TUM). In 2016, he became professor with TUM where he is leading the Data Analytics and Machine Learning Group. In 2020, he was appointed as director of the Munich Data Science Institute. His research focuses on robust machine learning approaches, ensuring a reliable use of AI techniques in science and industry.

Gudrun Klinker received the Diploma in informatics from Hamburg University, Hamburg, Germany, 1982, and the PhD degree in computer science from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1988. She worked with the Cambridge Research Lab of Digital Equipment Corporation, the European Computer-Industry Research Center and the Fraunhofer Institute for Computer Graphics on various aspects of scientific visualization and augmented reality. Since 2000, she is a professor with the Technical University of Munich. She is a member of the steering committees of IEEE ISMAR and VR. She has been an associate editor of the *IEEE Transactions on Visualization and Computer Graphics* and the *Computers & Graphics*. She has served on numerous program committees such as VRST, 3DUI, and UIST. Her research interests focus on developing approaches to ubiquitous augmented reality that lend themselves to realistic industrial and consumer applications including games.