# CAD2Real: Deep learning with domain randomization of CAD data for 3D pose estimation of electronic control unit housings

Simon Bäuerle[1,2], Jonas Barth[2], Elton Tavares de Menezes[2], Andreas Steimer[2], Ralf Mikut[1]

[1] Institute for Automation and Applied Informatics, Karlsruhe Institute of Technology, Karlsruhe, Germany
[2] Robert Bosch GmbH
E-Mail: {simon.baeuerle, ralf.mikut}@kit.edu

## Abstract

Electronic control units (ECUs) are essential for many automobile components, e.g. engine, anti-lock braking system (ABS), steering and airbags. For some products, the 3D pose of each single ECU needs to be determined during series production. Deep learning approaches can not easily be applied to this problem, because labeled training data is not available in sufficient numbers. Thus, we train state-of-the-art artificial neural networks (ANNs) on purely synthetic training data, which is automatically created from a single CAD file. By randomizing parameters during rendering of training images, we enable inference on RGB images of a real sample part. In contrast to classic image processing approaches, this data-driven approach poses only few requirements regarding the measurement setup and transfers to related use cases with little development effort.

# 1 Introduction

An exemplary use case for our approach is the 3D pose estimation of electronic control units (ECUs). The pose of each individual ECU needs to be detected robustly to enable an automated application of sealing materials. Generally, automated image processing is widely used in industrial series production [1, 2]. A commonly used method to detect ECU poses is by applying classic image processing. In some cases, these algorithms rely on fiducial marks imprinted on parts themselves. Setting up this image processing pipeline needs to be done by experts individually for each new product. Substituting classic image processing by fully automatically designed deep learning on our task can potentially save a significant amount of development effort for new product designs. Furthermore, ANNs generally impose much lower requirements regarding camera resolution and surrounding conditions, enabling simpler measurement setups. Adding fiducial markers in industrial applications “may be undesirable” [10]. Dropping the need for fiducial markers prevents changes on the product design, which would involve product engineers. However, deep state-of-the-art ANN architectures require a large number of labeled images, which are usually not available for ECUs. In contrast to real-world images, rendered CAD images are a widely available data source in industrial settings. A network trained solely on CAD data cannot be directly applied to real images though, since those are different with respect to pixel color values (see [3]). This domain gap is generally present on many different settings that involve ANNs. Techniques to overcome this domain gap are called domain adaptation and are subject to current research (e.g. [4, 3]). Instead of adapting the ANN to a different domain, an approach can also include adaptation of the domain itself: Images from the training domain can be randomized to such an extent, that the real-world domain is "just another variation" [5]. This method is called domain randomization and has been tested successfully on other use-cases such as indoor drone flight [6] or robotic grasping and manipulation [5, 7, 8, 9, 10]. Sundermeyer et al. are working on pose estimation by using a denoising autoencoder architecture [11, 12]. In contrast to Sundermeyer et al., we are using state-of-the-art ANN architectures and evaluate different randomization parameters. Tremblay et al., Khirodkar et al. and Hinterstoisser et al. are using domain randomization for object detection [13, 14, 15]. Khirodkar et al. focus

on the use case of detecting cars and also includes a pose estimation. Domain randomization is not limited to image data however. For example, Peng et al. have randomized the dynamic properties of their simulation model to transfer a robotic control algorithm trained by deep reinforcement learning to the real world [16].

In contrast to our setup, pose estimation approaches like BB8 [17], SSD-6D [18] or PoseCNN [19] employ further pose refinement to improve accuracy [20, 21]. Tekin et al. [20] and Do et al. [21] use standardized datasets, which does not target the domain gap that is widely present in real-world use cases. Kleeberger et al. also use domain randomization for pose estimation, but uses depth data instead of RGB images [22].

The generally very promising results on related use cases motivate the deployment of deep ANNs for pose estimation of ECUs. In Section 2 we outline our approach in a general way. This description is made on an abstract level without implementation details. It serves as a template, which can easily be applied to similar use cases. Subsequently, the experimental setup as described in Section 3 includes the details. In contrast to the previous section, we set out specific implementation aspects. A detailed overview is given e.g. over the parameters that we randomize and the way we set up the training datasets. Results for our use case are presented in Section 4. Performance during inference is listed for the different training datasets. We include an estimation of how much errors differ from their mean values. In Section 5 we discuss the results. We analyze the effects of different randomizations of training data. Furthermore, we compare this data-driven approach against classic image processing setups, with a focus on the possible impact on future manufacturing setups. Eventually, in Section 6 the most relevant aspects are summarized and a detailed outlook onto further research opportunities is given.

## 2 Methods

Figure 1 gives an overview of our approach. To cope with the domain gap between simulated and real-world images we use domain randomization. Unlike domain adaptation methods, domain randomization does not adapt the ANN

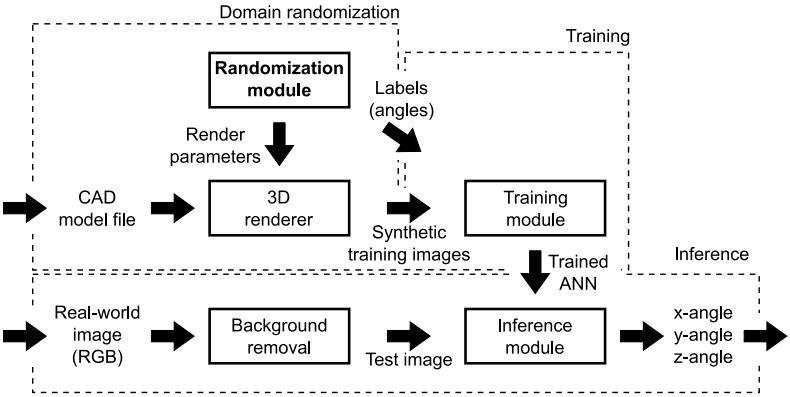



Figure 1: Overview of our approach

to a different domain. Instead, it rather adapts the training domain itself. Our training domain consists of CAD images, which are rendered with arbitrary settings. We generate different datasets of synthetic images automatically. We train state-of-the-art ANNs on synthetic datasets and perform inference on real-world images.

## 2.1 Domain randomization

We created a generic pipeline for applying domain randomization as depicted in the upper left area of Figure 1 (our specific implementation details are outlined in Section 3). CAD files are loaded into a 3D rendering software suite. A common exchange format is used for transferring CAD files. Geometric features are not changed in any way. Within the 3D renderer, several randomizations are applied. Specifically, we randomize shadows, translations and gray tones. Those are all parameters, which are not causally connected to the labels. For example, gray tones have no causal connection to rotation angles and must not affect inference. The pose of the model is set to a random angle configuration. An arbitrary number of angle configurations with individual randomizations is rendered and output to image files. Randomization parameters and rotation angles (=labels) are set via a scripting interface. Datasets with different randomization parameters can be created in any desired number automatically. This

kind of domain randomization is used to close the domain gap from training on CAD data to inference on real-world data (CAD2Real), saving the need for any labelled real-world images. Once finalized, this setup can be used for different products with minimum effort by replacing the CAD model and rerunning the script. Images for our baseline dataset *CAD_unchanged* are created with this method as well by simply omitting the randomizations.

### 2.2 ANN inference on real-world images

Real-world images are automatically preprocessed and then fed into the trained ANN. As shown in the lower part of Figure 1, the ANN infers all three rotation angles directly from real-world RGB images. We use a fully automatized preprocessing step to replace the background with a uniform gray tone.

## 3 Experimental Setup

We described our approach generically in Section 2. Here, we provide specific implementation details. We outline the creation of our different datasets used for training and during inference. Details regarding the ANN are also provided below.

### 3.1 Datasets

We work with two general types of datasets. On the one hand we have different training datasets. Those are solely based on CAD data and generated automatically. On the other hand we use experimental data for testing purposes. This experimental data is captured from real product samples in a laboratory setting.

Table 1: Parameters and their respective ranges

| Parameter | Range |
|---|---|
| X-/y-angle | [-15 $^o$, 15 $^o$] |
| Z-angle | [-45 $^o$, 45 $^o$] |
| Part translations | [-1.5 cm, 1.5 cm] |
| Camera translations | [-1.5 cm, 1.5 cm] |
| Gray tones | [0.05, 0.8] |
| Light positions | $x \in$ [-1 m, 1 m]<br>$y \in$ [-1 m, 1 m]<br>z = f(x, y) |

### 3.1.1 Training sets

For creation of our training datasets we use the 3D rendering software *Blender*. Blender is open-source software and freely available. It includes a powerful Python API that enables control via automatic scripts. The image generation pipeline is generally depicted in Figure 1. We import the CAD model as STEP-file, a data-format commonly used for CAD data exchange. The geometry itself is not modified. Rotations are applied around the x-/y- and z-axis within a range of -15 to 15 degrees for the x- and y-axis and a range of -45 to 45 degrees for the z-axis. The rotation angles serve as labels and are therefore stored for each image created. Random translations of the model along the x- and y-axis are optionally applied within a range of -1.5 cm and 1.5 cm. Random translations of the camera along the z-axis are optionally applied within a range of -1.5 cm to 1.5 cm. One light source is placed high above the model, emitting uniform lighting. Two additional light sources are optionally included as well. Those are placed randomly above the model, generating random shadows. The color of the model itself is randomized in uniform tones of gray. We are aware that the introduction of additional light sources also makes the part appear brighter. Therefore, we try to limit this effect by keeping the distance of both additional lights to the part on a constant value. For each randomly drawn x- and y-coordinate the z-coordinate is calculated so that the distance to the part

is always equal, effectively placing both lights on an imaginary sphere. All parameters with respective ranges are listed in Table 1.

We are using the training sets described below. During evaluation we compare our domain randomization approach against two datasets:

- *CAD_unchanged*: Images are labeled with rotation angles around x-/y- and z-axis. There are no further modifications made.
- *CAD_augmented*: We take the images from the set *CAD_unchanged* and apply random modifications. We modify brightness, translations and zoom. This data augmentation is applied to already rendered images. Data augmentation of this kind is usually used when not enough labeled training data samples are available.

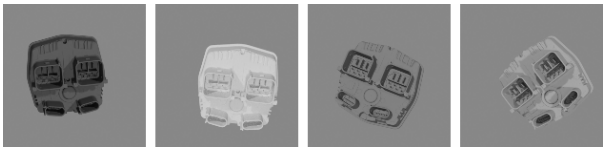
The following datasets include domain randomization. To test the influence of different parameters, we modify the extent of our randomization. Samples for each dataset are shown in Figure 2.

- Dataset *Rand_full* includes all randomizations described above.
- Dataset *Rand_noShadows* is the same as *Rand_full* except for the two additional light sources. Thus, no random shadows are included.
- Dataset *Rand_noTranslations* is the same as *Rand_full* except for the translations of the part and the camera. Part and camera remain at constant positions.
- Dataset *Rand_noGraytones* is the same as *Rand_full* except for randomizing the uniform gray tones. Part color is only affected by the position of both randomly placed lights.

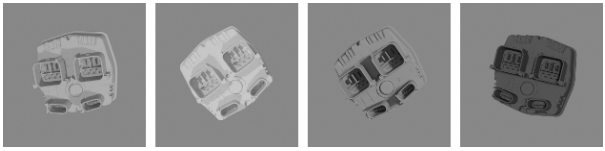
For each dataset described here, we have created 100 000+ training images.

(a) *CAD_unchanged*

(b) *CAD_augmented*

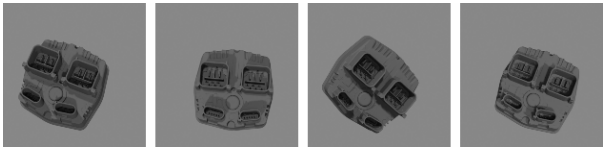


(c) *Rand_full*



(d) *Rand_noTranslations*

(e) *Rand_noShadows*



(f) *Rand_noGraytones*

Figure 2: Example images from training datasets

Figure 3: Example images from our real-world dataset with green background

#### 3.1.2 Real-world dataset

We outline the creation of our real-world dataset, which is used as test set during inference. We use a single sample part from an ECU that is already available during prototype phases (before starting series production). We use the following laboratory setup: The images are recorded with a common SLR camera, since there are no specific requirements regarding the camera. Image resolution is later scaled down during pre-processing to below 300x300 pixels. The camera is mounted onto a fixed frame with two lighting sources on either side. The sample part is placed 0.5 m below the camera on a green cardboard layer. To introduce rotations around the horizontal x- and y-axis we are using 3D-printed wedges. We use multiple wedges with slopes of 2.5, 5 and 10 degrees. Placing the wedges below our sample introduces the respective rotations. We recorded 20 different angle configurations, examples are shown in Figure 3.

### 3.2 Artificial Neural Network: Training and inference

We use the state-of-the-art architecture *InceptionV3* [23], with weights pre-trained on ImageNet. The architecture including its weights can be imported within Keras [24] in two lines of code. To adjust to our number of labels we append a dense layer with three neurons. Pose estimation is sometimes implemented as classification, with a binning of rotation angle intervals. Binning limits the resolution of angle values to discrete intervals. To avoid this, we

opt for using regression as proposed by e.g. Mahendran et al. [25] instead. We use a linear activation function within the last layer, directly outputting continuous rotation angles around the x-/y- and z-axis. Each ANN is trained on a dataset as described above. We train on each dataset with 10 000 randomly drawn samples for 50 epochs. First runs have shown slightly differing results when re-drawing the samples and re-starting training. Therefore, we execute 30 independent runs on each dataset. For final evaluation we use error metrics as described below. The lower area of Figure 1 shows our setup for inference with the trained ANN. We use preprocessing of the real-world images for removing the background. Since the images are taken on green background, preprocessing can be done automatically without much effort.

## 3.3 Evaluation metrics

In this section we describe the metrics used for evaluation later on. First, we focus on the evaluation of the training set characteristics (to get an impression of "which training set is best"). We then also estimate the confidence of statements that are made on our limited number of real-world images. For a single test or validation image, an ANN outputs three distinct angle values $\hat{y}_i, i \in \{1,2,3\}$. These are compared against the corresponding ground truth angle values $y_i, i \in \{1,2,3\}$ by calculating error

$$e_i = \| \hat{y}_i - y_i \|_1 \, . \tag{1}$$

Each single ANN is evaluated on validation and test data. We merge all angle values ($\hat{y}_i$ and $y_i$) into one vector per dataset ($\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$). This is done on validation and test data separately for each ANN. Errors are then averaged over all angles and the respective number of images, yielding an mean error per ANN of

$$\overline{e}_{ANN} = \frac{1}{3N} \| \hat{\boldsymbol{y}} - \boldsymbol{y} \|_1 \, . \tag{2}$$

$N$ is the number of images in either the validation ($N = 500$) or test set ($N = 20$). Since we assess the fitness of the training set characteristics, we calculate the mean error per training set with

$$\overline{e} = \sum_{j=1}^{M} \overline{e}_{ANN,j}, \tag{3}$$

with $M = 30$ ANNs for each training dataset. We further calculate the standard deviation per training set

$$s = \sqrt{\frac{1}{M-1} \sum_{j=1}^{M} (\overline{e}_{ANN,j} - \overline{e})^2}. \tag{4}$$

This yields a standard error of

$$s_{\overline{e}} = \frac{s}{\sqrt{M}}. \tag{5}$$

We further calculate a margin of error for $\overline{e}$. Since our sample size is limited, we use the Student's t-distribution instead of the normal distribution. For our sample size of $M = 30$ and a confidence level of 99% we calculate the margin of error for $\overline{e}$ as

$$MOE = \pm t_{M-1} s_{\overline{e}}, \tag{6}$$

with $t_{M-1} = 2.76$. When working with a normal distribution instead, $t_{M-1}$ would be replaced by $z_c$ with a value of 2.58 for the 99% confidence level ($t_{M-1}$ converges towards $z_c$ for very large sample sizes). The value of $t_{M-1}$ is retrieved from a table for the Student's t-distribution (see e.g. [26], p.206) and depends on the sample size and the confidence level. For our sample size of $M = 30$, we need a $t$-value that corresponds to $M$ - 1 = 29 degrees of freedom. The values of the distribution function in the table can be used directly when working with a one-sided interval. Our two-sided interval gives an estimation into both directions (upper and lower bound). Therefore, we record $t$ for a value of 0.995 for the distribution function to account for the two-sided interval. Up to now, we have mainly looked at how results differ when re-drawing samples from the training set and retraining the ANN. The limited number of images within the test set is another factor that might affect our results. For our limited

Table 2: Mean error on validation data and real-world data

| Training dataset | Validation data $\overline{e}$ [$^o$] | Real-world data $\overline{e}$ [$^o$] |
|---|---|---|
| CAD_unchanged | $0.4 \pm 0.04$ | $11.7 \pm 2.4$ |
| CAD_augmented | $0.4 \pm 0.05$ | $2.5 \pm 0.6$ |
| Rand_full | $0.5 \pm 0.06$ | $\mathbf{1.5 \pm 0.2}$ |
| Rand_noTranslations | $0.5 \pm 0.07$ | $7.7 \pm 2.8$ |
| Rand_noShadows | $0.5 \pm 0.1$ | $3.6 \pm 0.8$ |
| Rand_noGraytones | $0.4 \pm 0.09$ | $\mathbf{1.2 \pm 0.2}$ |

amount of real-world samples we take a similar approach as above, but now on a more isolated scope. For the first ANN trained on dataset *Rand_full*, we evaluate the mean error $\overline{e}_{test}$ for angles around the x-axis on the test set containing 20 images. This is the same calculation as presented in Equation (2), but discarding angles around y- and z-axis. For those 20 error values $e_i$ we also calculate the standard deviation

$$s_{test} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(e_i - \overline{e}_{test})^2} \tag{7}$$

and the margin of error

$$MOE_{test} = \pm t_{N-1}\frac{s_{test}}{\sqrt{(N)}} \tag{8}$$

with $t_{N-1} = 2.86$ for $N = 20$ and a 99% confidence level.

## 4 Results

We train 30 ANNs on each of the datasets described in Section 3.1. We evaluate the mean absolute error and include a margin of error as described in Section 3.3 for test and validation data on each dataset. Results are shown in Table 2. The outer left column indicates the dataset used during training. We then eva-

luate the performance on validation data and real-world data. Validation data images are from the same domain as the images used during training. Real-world data is taken from product samples and therefore substantially different. This is our target domain used for testing. ANNs trained on *CAD_unchanged* exhibit the lowest error on the validation set, but insufficient performance on real-world test images. For *CAD_augmented* the mean absolute error on real-world images is lower by a factor of approximately five. Another improvement by another factor of almost two over *CAD_augmented* is gained by using *Rand_full*: Angles around all axes are inferred with an mean error of 1.5 degrees. *Rand_full* has full randomization applied. For *Rand_noTranslations* and *Rand_noShadows* we note an error-score inbetween *CAD_unchanged* and *CAD_augmented*. Dropping the randomization of translations affects performance worse than dropping randomization of shadows. Training on *Rand_noGraytones* gives slightly better results than on *Rand_full*, but only by a small margin. Errors on the validation set increase from *CAD_unchanged* to *CAD_augmented* and further rise for the randomized datasets. All randomized datasets show similar errors on validation data. We now take an isolated look onto the limited number of real-world samples as described in Section 3.3. We evaluate the error for rotations around the x-axis only and look at a single ANN trained on *Rand_full*. For our 20 real-world samples the ANN inference has a mean error of $\mathbf{1.6 \pm 0.4}$ **degrees**. This margin of error is calculated for a confidence level of 99%. Our experimental setup as described in Section 3.1 has measurement errors which affect the ground truth labels. All results presented above are naturally limited to measurement tolerances.

# 5 Discussion

The results presented in Section 4 show a consistent advantage by training on randomized datasets for our application of pose estimation of ECUs. In the later part of this section we also discuss the impact of this research direction on image processing setups in related product applications. But first we look more closely at the effects of how the datasets were set up. First of all, the insufficient performance with the dataset *CAD_unchanged* is not surprising. In this case, the training images differ a lot from the real-world images. This

can be interpreted as a "wide" domain gap, leading to poor transferability from source domain to target domain. A significantly improved performance on real-world images is achieved by applying state-of-the-art data augmentation to the training set. Data augmentation is commonly used to expand the training set size. This is especially useful when dealing with a limited amount of labeled training data. We believe that there is a second benefit of data augmentation. Augmenting training data with changing brightness or translations also increases the diversity within the training set. Increased diversity of features not relevant for inference favors transferability from source to target domain. This effect is exactly the underlying idea of domain randomization. Data augmentation therefore can be seen as a "light version" of domain randomization. With full domain randomization applied, inference quality is further increased by another factor of approximately two. In comparison to data augmentation, domain randomization introduces even more diversity to the training set. This time, the introduced diversity goes beyond simply adjusting images. Modifications of this kind cannot be easily applied to raw images. This is especially clear for the randomization of shadows. Calculating the position and intensity of shadows is an integral part during rendering and not easily possible when working on two-dimensional image data only. Also, translations made within the renderer lead to different outcomes compared to augmentation by translations as well. Translations applied during state-of-the-art data augmentation will not change camera perspective. In contrast, inside the renderer not only the part position changes, but also the perspective view of the part changes. Translations in data augmentation therefore are different from those in domain randomization. However, we believe that the major error reduction is achieved simply by the fact that additional translations are introduced, no matter whether perspective changes or not. The poor performance with the dataset *Rand_-noTranslations* especially motivates the introduction of translations. To get a better impression of the different aspects of domain randomizations, in addition to dropping the added translations in *Rand_noTransla- tions* we dropped the shadow randomization in *Rand_noShadows* and the gray tone randomization in *Rand_noGraytones*. The performance with *Rand_noShadows* is worse compared to *Rand_full* and *CAD_augmented*. It seems that shadows and translations are both relevant factors when dealing with the present domain gap. However, dropping the randomization of gray tones in *Rand_noGraytones* has not caused

deteriorating performance, but even shows slight improvements compared to *Rand_full*. Since we trained 30 different ANNs we do not believe this effect is caused by chance. A possible explanation is that by randomization of gray tones many gray tones are outside of a usable scope (e.g. too light or too dark). This leads to a smaller part of training set being useful for inference, since some images are "too far off". Also, we want to mention that the effects of introducing random shadows and random gray tones overlap in some sense. Both affect the color of the part at a certain position and are not entirely independent of each other. In our opinion, the unexpected behavior on the dataset *Rand_noGraytones* does not hurt the idea of domain randomization in general. It rather motivates further studies on the unique effects of different randomization types. Instead of randomizing gray tones only, e.g. textures could be introduced as well.

Also, the varying error on the validation set from CAD datasets to randomized datasets motivates the analysis of different hyperparameters, mainly training set size and the number of epochs. The most efficient hyperparameters are generally likely to be different depending on the randomization type and extent. With domain randomization we desire to cover all aspects during rendering that make the real-world images different from the CAD images. This does not mean representing reality in simulation exactly however. For example when dealing with differing textures of the part in the real-world, applying textures with random noise to the simulation might be sufficient. The successful application of domain randomization on this use case shows high potential for future setups of image processing pipelines in series production of ECUs. The pipeline that we used can easily be transferred to other products. Other products may be manufactured on different production lines. Subsequent processing of pose information varies between products or production lines. To standardize e.g. the naming of parameters on these interfaces and during further processing steps, an ontology-based approach is useful. Zhou et al. [27] and Svetashova et al. [28] have applied ontologies to other production processes successfully. This approach not only helps during technical setup, but also enables a common understanding of process-specific details across all involved persons [27, 28]. In contrast to many algorithms of classic image processing, our pose estimation approach is not bound to specific product

features. Further improvements aimed at improving inference accuracy can also maintain this product-independent aspect. This advantage is based on the fact, that the features needed for inference are learned by the ANN during the training process and not manually tuned. This data-driven approach has the advantage that for a different problem setting only the problem-specific training data needs to be supplied. We use only data sources that are available without major effort. With our approach, the problem-specific training data can be created automatically from a single CAD model file. Once again we emphasize that training is done on purely synthetic data. Not a single real-world image is needed during training. We see a high potential for a significant reduction of development effort in future image processing setups.

## 6 Summary and Outlook

We have set out to analyze the applicability of domain randomization to our use case of pose estimation of ECUs. Our goal was to minimize the domain gap and to deploy an ANN trained solely on synthetic data to real-world images. We have shown that applying domain randomization exceeds the effect of data augmentation by a factor of around two. The mean error for inferred rotations around all three axes is only 1.2 degrees on real-world images. The entire pipeline for creating randomized training datasets and training the ANN is fully automatized. The only input needed for creation of all training data is a single CAD model file, which is readily available for all ECUs. We use only a state-of-the-art ANN architecture with a minor adjustment regarding the output dimensionality. Training is done end-to-end, we infer all rotation angles directly from RGB images. No further depth data is needed. We have analyzed the application to our use case and motivated further research directions. The following aspects could make this approach fit for application in production of ECUs:

- We focused on detecting part rotations. For application in series production a post-processing step to determine translational degrees of freedom needs to be appended. Including the translations directly into the labels as well might also be a feasible approach for our use case (directly inferring 6D pose information).
- Our pre-processing currently requires a green-colored background. Randomizing the backgrounds as done in other use cases [5, 10, 13, 11] could make our approach feasible for a background containing workpiece carriers. This would drop the need for using any pre-processing at all.
- We have provided insights into the effect of different randomizations. To further improve accuracy, these influences need to be examined in more detail. Ideally, this simultaneously includes adjustment of hyperparameters for training the ANN as well.

The successful execution of the steps outlined above can reduce the entire pipeline for 6D pose estimation to solely a state-of-the-art ANN architecture. These architectures are conveniently available within the Keras library. This would provide a fully automated pipeline for pose estimation of new ECUs and similar products.

## References


[1] A. Korodi, D. Anitei, A. Boitor, and I. Silea, "Image-processing-based low-cost fault detection solution for end-of-line ECUs in automotive manufacturing," *Sensors*, vol. 20, no. 12, p. 3520, 2020.

[2] S.-H. Huang and Y.-C. Pan, "Automated visual inspection in the semiconductor industry: A survey," *Computers in Industry*, vol. 66, pp. 1–10, 2015.

[3] E. R. Tavares de Menezes, "Machine learning and classic image processing - extraction of 3D information from real-world 2D images

of electronic part housings," Masters thesis, TH Köln University for Applied Sciences, 2019.

[4] M. Böhland, T. Scherr, A. Bartschat, R. Mikut, and M. Reischl, "Influence of synthetic label image object properties on GAN supported segmentation pipelines," in *Proceedings 29th Workshop Computational Intelligence*, pp. 289–305, KIT Scientific Publishing, 2019.

[5] 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 *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, IEEE, 2017.

[6] F. Sadeghi and S. Levine, "Cad2RL: Real single-image flight without a single real image," arXiv preprint 1611.04201, 2016.

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

[8] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, and others, "Solving rubik's cube with a robot hand," arXiv preprint 1910.07113, 2019.

[9] S. Grün, S. Höninger, P. M. Scheikl, B. Hein, and T. Kröger, "Evaluation of domain randomization techniques for transfer learning," in *2019 19th International Conference on Advanced Robotics (ICAR)*, pp. 481–486, IEEE, 2019.

[10] X. Ren, J. Luo, E. Solowjow, J. A. Ojea, A. Gupta, A. Tamar, and P. Abbeel, "Domain randomization for active pose estimation," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019 pp. 7228–7234, IEEE, 2019.

[11] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3D orientation learning for 6D object detection from RGB images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 699–715, 2018.

[12] M. Sundermeyer, M. Durner, E. Y. Puang, Z.-C. Marton, N. Vaskevicius, K. O. Arras, and R. Triebel, "Multi-path learning for object pose estimation across domains," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13916–13925, 2020.

[13] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 969–977, 2018.

[14] R. Khirodkar, D. Yoo, and K. Kitani, "Domain randomization for scene-specific car detection and pose estimation," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1932–1940, IEEE, 2019.

[15] S. Hinterstoisser, O. Pauly, H. Heibel, M. Martina, and M. Bokeloh, "An annotation saved is an annotation earned: Using fully synthetic training for object detection," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 0–0, 2019.

[16] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[17] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3828–3836, 2017.

[18] 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 *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1521–1529, 2017.

[19] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes," arXiv preprint 1711.00199, 2017.

[20] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6D object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 292–301, 2018.

[21] T.-T. Do, M. Cai, T. Pham, and I. Reid, "Deep-6Dpose: Recovering 6D object pose from a single RGB image," arXiv preprint 1802.10367, 2018.

[22] K. Kleeberger and M. F. Huber, "Single shot 6D object pose estimation," arXiv preprint 2004.12729, 2020.

[23] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[24] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[25] S. Mahendran, H. Ali, and R. Vidal, "3D pose regression using convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 2174–2182, 2017.

[26] J. Puhani, *Statistik: Einführung mit praktischen Beispielen*. Springer Fachmedien Wiesbaden, 2020.

[27] B. Zhou, Y. Svetashova, S. Byeon, T. Pychynski, R. Mikut, and E. Kharlamov, "Predicting quality of automated welding with machine learning and semantics: A Bosch case study," in *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, p. 8, 2020.

[28] Y. Svetashova, B. Zhou, T. Pychynski, S. Schmidt, Y. Sure-Vetter, R. Mikut, and E. Kharlamov, "Ontology-enhanced machine learning pipeline: A Bosch use case of welding quality monitoring," in *The Semantic Web - ISWC 2020*, 2020.