# Measuring similarity of rendered and real image pairs using domain translation by employing Conditional Generative Adversarial Networks

Naveen Raj Datha[1] and Marcus Thiel[2]

[1] Fraunhofer Institute for Factory Operation and Automation IFF,
Sandtorstraße 22, 39106 Magdeburg
[2] Otto-von-Guericke-Universität Magdeburg,
Fakultät für Informatik, Data & Knowledge Engineering Group,
Universitätsplatz 2, 39106 Magdeburg

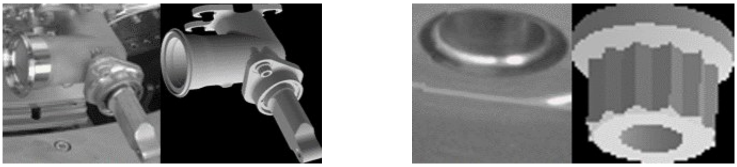
**Abstract** One way for the visual inspection of assemblies with many variants is to compare camera images with the corresponding rendered view of the CAD model. In this paper, we address the problem to decide whether there are significant differences between camera and rendered images, which signal an assembly error. Our approach uses a Conditional Generative Adversarial Network (CGAN) to translate the camera image to a rendered like one, followed by error detection by comparing the translated and rendered images.

**Keywords** Automated visual assembly inspection, CGAN, deep learning, quality assurance, human-machine systems

## 1 Introduction

This research is motivated by an inspection task in manual assembly. In order to ensure that a module is correctly assembled visual inspection is a frequent choice. When assembly errors may cause high costs in downstream processes or could lead to dangerous malfunction, an investment into a reliable automated inspection solution is of interest. For assemblies with many variants the following approach with cameras could be used. The cameras take images of

DOI: 10.58895/ksp/1000124383-22 erschienen in:
**Forum Bildverarbeitung 2020**
DOI: 10.5445/KSP/1000124383 | https://www.ksp.kit.edu/site/books/m/10.58895/ksp/1000124383/



(a) Correct assembly (b) Assembly error (missing part)

**Figure 1.1:** Pairs of camera and rendered images.

sufficient resolution of the parts to be inspected in such a way, that we know the position of the camera with respect to the assembly. This allows us to render the same view using the CAD model [1]. Then we compare the rendered view and the real camera image and to decide, whether there is an error. In the current work, we refer to the rendered view as the **CAD image** and the real camera image as **real-world image**.

The CAD model provides only geometrical information. A photo-realistic rendering is not possible. Yet, intensity changes can be expected, where surface normals change or neighbor pixels are on different objects or background. That is why the existing image processing approaches focus on the comparison of edges detected in CAD and real-world images [2]. As expected edges appear more or less distinct, and as there are further edges from texture and illumination, edge detection and comparison criteria need to be parametrized based on example images from the assembly. Whenever there are new parts, their finishing changes or lighting conditions are modified it may be necessary to adapt parameters and criteria again. Therefore, it is natural to ask whether there is a machine learning approach, which learns the classification of assembly errors and correctly mounted parts based on annotated example images with only few examples for assembly errors. In the current work, we introduce such a data-driven learning mechanism. Our approach can be easily adapted to new assembly products, parts or conditions, with minimal human expert involvement.

**Dataset:** The dataset we used consists of image pairs obtained from 42 real-world assemblies of 3 different assembly products. Figures 22.1(a) and 22.1(b) show such sample image pairs. From all the 42 experiments we have around 24333 inspection tasks, i. e. 24333 image pairs that can be used for training and testing. In the 24333 image pairs only 260 image pairs are assembly errors and all the other are correctly assembled samples. For our learning task, we leave out the samples from 9 experiments (3 from each assembly type) for testing and use the remaining samples for training. From here on, we refer to the correctly assembled samples as **negative samples** and the assembly error samples as **positive samples**.

We categorize the parts-of-interest in our inspection tasks into 10 different categories based on their visual appearance. Figure 1.2 shows one sample from each of these categories and their names. Also, we increase the dataset size by performing data-augmentation (discussed later in section 2). We adjust all the images to aspect ratio 1 : 1 for the sake of ease of training Convolutional Neural Networks.

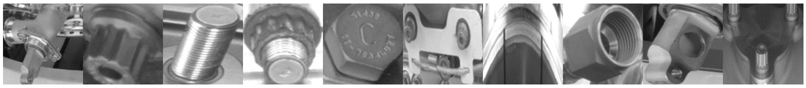


**Figure 1.2:** Sample images of 10 different categories. Names in order from left: Air-Adapters, Bolts-1, Bolts-2, Bolts-3, Bolts-4, Mounting-Plates, Stiffeners, Swivel-Nuts, Vent-Tubes, Miscellaneous categories.

## 2 Method

Suppose that the real image could be translated from the domain of real textured images to the domain of rendered like images, then an image comparison could be used for classification. Similar images would represent the correctly mounted parts and differences in the images would signalize assembly errors. The intention is to simplify the classification to asking whether a similarity measure for two images is above or below some threshold. The learning comes in with domain translation. The advantage is that for learning the domain translation we only need negative samples.

Figure 2.1 shows the conceptual idea behind our methodology, the data flows from left to right in the pipeline. The first stage is about pre-processing data. We generate a mask from the CAD image, where the pixels of part of interest are assigned value 0 and the background pixels are assigned a value 1. We then extract the background from the real-world image by simply multiplying the real-world image with the mask image. The extracted background is then added to the CAD image resulting in a hybrid image. We use this hybrid image as our ground-truth for training and also for classification of assembly errors. Figure 2.2 shows an example of these data pre-processing steps. In section 3 we also discuss results of experiments, where we omitted this pre-processing and kept the black background.

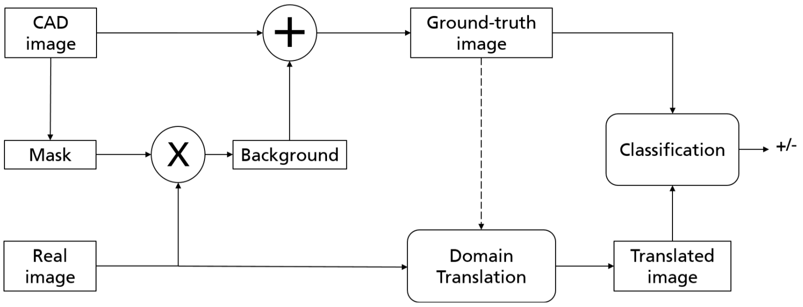


**Figure 2.1:** Conceptual diagram

The second stage is image domain translation. For domain translation, we choose a deep learning approach: Conditional Generative Adversarial Networks (CGAN) [3]. CGANs are a special case of Generative Adversarial Networks (GAN) [4] which are state-of-the-art image generation models. A CGAN consists of two blocks, generator and discriminator, both these blocks are made up of Convolutional Neural Networks. During training, the generator of CGAN learns to translate input real-world image to CAD domain. The discriminator on the other hand learns to distinguish between the generator's output and ground-truth CAD image, given the real-world image. For the generator network, we experimented with two different architec-

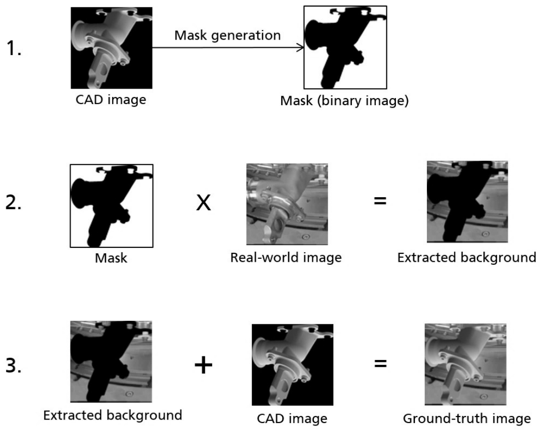


**Figure 2.2:** 3 step data pre-processing to obtain ground-truth images

tures from the state-of-the-art, the pix2pix [5] and U-Net [6] architectures. For the discriminator, we use the architecture proposed in pix2pix [5]. Also, to make domain translation invariant to Euclidean transformations or small changes of intensities, we apply data augmentation techniques such as Flipping, rotating, translating, small random increase/decrease of pixel values on the training data set. The third stage of the conceptual design consists of a classification block, where we compare the translated and ground-truth images to detect assembly errors. Note that, though we use the terminology of classification here, no learning process happens in this block. The actual learning process happens only in the domain translation block.

In our approach, we need image comparison metrics for two purposes. One for evaluating the quality of image translation, the other for comparing the translated and ground-truth images in the classification block. For the purpose of measuring the image translation quality we use the Structural SIMilarity (SSIM) Index [7]. Given a pair of perceptually similar images, SSIM gives a measure of similarity in the structural information of the images. A good domain translation model, while translating the domain of an input image, should not affect/degrade the structural information present in it. Thus, comparing the structural information in the translated images with the structural information in the ground-truth images forms a good basis for testing the translation quality. The SSIM metric serves this purpose here. Note that, for training and testing the CGAN performance, we only need the negative samples from the dataset, we do not need the positive samples here.

For comparing the translated and ground-truth images in the classification block, we use the Mean squared error (MSE). MSE is calculated as the mean of squared pixel intensity differences between the given pair of images. MSE is calculated at pixel level and does not take into account the neighborhood relations. However, MSE highly penalizes large deviations in pixel intensities. This factor greatly helps in classifying the borderline positive samples, where the parts-of-interest in the image pair are mostly similar with only some small differences, see figure 22.1(b). To classify a sample as negative or positive based on MSE, we need a threshold. We choose the threshold as a trade-off between the sensitivity and specificity measures. The sensitivity and specificity measures are calculated as

$$\text{Sensitivity} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}, \tag{2.1}$$

$$\text{Specificity} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}. \tag{2.2}$$

We calculate the sensitivity and specificity over a range of different thresholds and finally choose the threshold where the sum of both measures is maximum. We use 60% of our test samples, both negative and positive for this purpose. Also, to have a balanced dataset for threshold calculation, we perform data-augmentation to generate artificial positive samples, by simply swapping the CAD image of a given image pair with some different CAD image.

## 3 Results

We performed a set of experiments to evaluate the performance of our pipeline with pix2pix and U-Net generator architectures. The architectures in all our experiments were trained using the Adam optimizer [8] with a learning rate of 0.0002. In case of discriminator we used Mean squared error (MSE) as the loss function, whereas in case of generator we used a weighted sum of Mean squared error (MSE) and Mean absolute error (MAE) as suggested in [5].

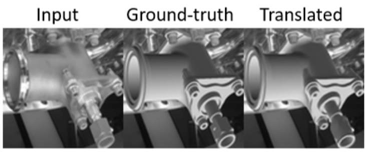
The results we report in this section were obtained for real-world to CAD image translation and by using images with background. Later in this section, we explain our observations on why translating CAD images to real-world results in poor translation quality and why it is not a good idea to use images without background for training. Also, the numbers reported in this section were obtained over original positive and negative test samples, no augmented data was included in these calculations.

**Table 1:** Image translation and classification results

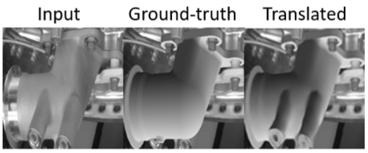
| | **Translation quality (Avg. SSIM)** | | **Classification performance (based on MSE measure)** | |
|---|---|---|---|---|
| **Architecture** | **Train-set** | **Test-set** | **Sensitivity** | **Specificity** |
| pix2pix | 0.93 | 0.92 | 0.75 | 0.97 |
| U-Net | 0.94 | 0.93 | 0.85 | 0.98 |

Table 1 lists the results of the best performing pix2pix and U-Net generator models. The pix2pix generator model took relatively longer training time compared to U-Net for achieving similar translation quality. In both cases, image translation quality remains good and consistent over train and test sets, indicating that the models learned to generalize. Figures 22.1(a), 22.1(b), 22.1(c) show some sample image translation results obtained with the U-Net model. In terms of classification performance, although the specificity is almost the same in both cases, we achieved better sensitivity with U-Net translation. While the pix2pix pipeline misses to detect 25% error samples in the test set, U-Net performs slightly better by missing 15% defects.

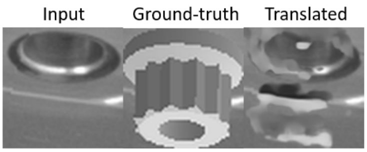
The results in table 1 were obtained using a common MSE threshold for all the 10 categories of objects. But, in production different types of errors can occur for different categories. For example, in our dataset, missing bolt is a most common error in case of bolts, whereas, in case of air-adapters the most common error is mount-



(a) Negative sample (No error)



(b) Positive sample (Wrong part)



(c) Positive sample (Missing part)

**Figure 3.1:** Image translation results

ing a wrong part. When the types of errors differ, the MSE values in these cases differ too and therefore it would be beneficial to use different cut-off values for each category of objects rather than using a common cut-off for all the categories. Table 2 summarizes the results we obtained after choosing individual cut-offs for each category. These results were obtained using the U-Net generator mentioned in table 1. Except for the Stiffeners category all other categories have sensitivity of 1.0 i. e., 100% error detection.

Stiffeners are a special category of objects which have an extreme aspect ratio, see figure 3.2. When these images are resized to a aspect ratio of 1 : 1, a lot of information about the part of interest is lost, and therefore it would be difficult to detect errors in such cases. We solved this problem by training the CGAN on image tiles, obtained by splitting each Stiffener image into multiple small tiles. And during classification, if any tile of an image is classified as positive then the image itself is classified as positive. Using this approach we achieved 100% error detection in case of Stiffeners too.

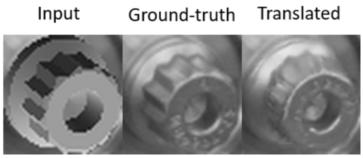
**Translating CAD images to real-world:** Figures 22.3(a) and 22.3(b) show the image translation results we obtained by training a CGAN model to translate CAD images to real-world images. The training

**Table 2:** Category-wise classification results obtained after choosing individual cut-offs for each category

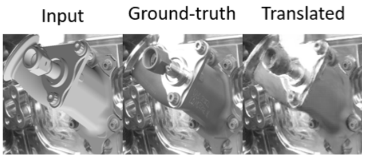
| Category | Sensitivity | Specificity |
|---|---|---|
| Air-Adapters | 1.0 | 1.0 |
| Bolts-1 | 1.0 | 0.99 |
| Bolts-2 | 1.0 | 0.99 |
| Bolts-3 | 1.0 | 0.98 |
| Bolts-4 | 1.0 | 1.0 |
| Mounting-Plates | 1.0 | 0.97 |
| Stiffeners | 0.75 | 0.98 |
| Swivel-Nuts | 1.0 | 1.0 |
| Vent-Tubes | 1.0 | 1.0 |
| Miscellaneous | 1.0 | 0.96 |

**Figure 3.2:** Image of a Stiffener with its original aspect ratio (14 : 1)

loss and the image quality stopped improving after we trained the model for a few hundred epochs. The possible reason here for poor image translation quality could be that, the process of transitioning from real-world to CAD-world is like a simplification process, the other way is not. Lets say there is a product which contains a part $X$, the part's CAD model image is $X_c$. Lets say for the purpose of training we obtained real-world images $X_{r1}$, $X_{r2}$, $X_{r3}$ of this part when the product was assembled three different times. Now, when we train the CGAN model with $X_{r1}$, $X_{r2}$, $X_{r3}$ as inputs and $X_c$ as the common ground-truth for all three inputs, we are essentially training the model to simplify the inputs and converge the output to the pixel values seen in $X_c$. But, when we train the model with $X_c$ as input and $X_{r1}$, $X_{r2}$, $X_{r3}$ as ground-truths, the model learns to generate a mean output image that equally satisfies the constraints of all three ground-truths it has seen during training. Therefore, translating CAD images to real-world might always result in blurry outputs.



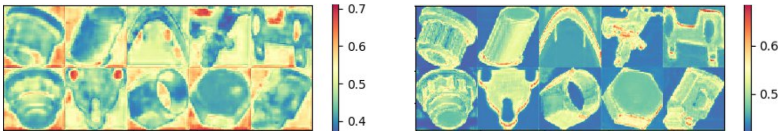
(a) Test sample 1



(b) Test sample 2

**Figure 3.3:** Blurry translation results obtained from a CGAN model trained to translate images from CAD domain to real-world domain

**Using images without background:** In the experiments where we trained the CGAN on images without background, we observed that the translation quality on training images was satisfactorily good, but the quality of outputs obtained on test-set was poor, indicating over-fitting. Figures 22.4(a) and 22.4(b) show the mean activation maps [9] we plotted for the top layers of the CGAN generator model to understand its behaviour. In figure 22.4(a), where we trained the model on images without background, we see that, in almost all categories of input images, the high activation values are in the background region (The reddish regions in the activation maps indicate high activations). But, in case of figure 22.4(b), where we trained the model on images with background, the activation values in the background region are lower than the part-of-interest, indicating that the model learned the structure of the part-of-interest. When a model is trained on images without background, the network might find it easier to learn about the black patches in the background, than learning about the complexity of structures in the region of interest. The Convolutional Neural Networks usually learn to find or extract the most common features that can differentiate one class from the other. Therefore in order to drive the network towards learning the right features for a given part of interest, we simply have to make sure that no other part or region in the image highly correlates with the part-of-interest. But this is not the case with black-background images. Whenever there is some specific part-of-interest in an image, there are always corresponding black-patches as well. Then the network might learn about the black-patches rather the part of interest. In short, the lower the correlation between background and



(a) Without background (b) With background

**Figure 3.4:** The activation maps of each category highlight the difference in behaviour of the CGAN model when trained on with and without background images. (Reddish regions in the images indicate higher activation values)

the region of interest, the better. The background added from the real-world image and the data-augmentation helps in achieving this randomness.

## 4 Summary

In manual assembly tasks, inspection of products assembled by humans is essential. One way of doing this is by automated visual inspection using camera images. Images captured in the real-world can be compared with images rendered from the CAD model to detect errors. The existing approaches focus on comparison of edges detected in rendered and real image pairs. However, when the products/parts or lighting conditions change, the parameters of these comparison algorithms have to be adjusted again by subject matter experts. In the current work we introduce a data-driven learning approach to solve this problem. We use the idea of image domain translation to translate the real-world images into rendered like ones, so that the translated and ground-truth images can be compared using simple image comparison measures, thus minimizing involvement of human experts. We use CGAN for the purpose of image domain translation and MSE for the purpose of image comparison. By choosing individual MSE thresholds for different types of parts and for some parts (with extreme aspect ratio) training on image tiles instead of whole image at once, we achieved 100% error detection while mis-classifying only 0.5% correct assembly samples as errors.

## References

1. S. Sauer, T. Dunker, and M. Heizmann, “Ein Framework zur Simulation optischer Sensoren,” in *20. GMA/ITG-Fachtagung Sensoren und Messsysteme 2019*. Nürnberg: AMA Association for Sensors and Measurement, 2019.
2. S. Sauer and D. Berndt, “Optische Montageprüfung unter Nutzung intelligenter Algorithmen,” in *3D-NordOst 2018*, Berlin, 2018, pp. 35–42.
3. M. Mirza and S. Osindero, “Conditional generative adversarial nets,” *arXiv preprint arXiv:1411.1784*, 2014.
4. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
5. P. Isola, J. Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *Proceedings CVPR 2017*, vol. 2017-Janua, pp. 5967–5976, 2017.
6. O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *Lecture Notes in Computer Science*, vol. 9351, pp. 234–241, 2015.
7. Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.
8. D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
9. F. Chollet, *Deep Learning with Python*, 1st ed. USA: Manning Publications Co., 2017.