Automated defect recognition in X-ray projections using neural networks trained on simulated and real-world data

Tobias Schön¹, Bishwajit Gosswami¹, Rasmus Hvingelby², Daniel Suth¹, Lukas Malte Kemeter², Paulina Sierak²

¹Fraunhofer Development Center X-Ray Technology EZRT, a Division of the Fraunhofer-Institute for Integrated Circuits IIS, Flugplatzstraße 75, 90768 Fürth, Germany, e-mail: tobias.schoen@iis.fraunhofer.de
²Center for Applied Research on Supply Chain Services of the Fraunhofer Institute for Integrated Circuits IIS, Nordostpark 84, 90411 Nürnberg, Germany, e-mail: rasmus.hvingelby@iis.fraunhofer.de

Abstract
In this contribution, we investigate a methodology based on neural networks for efficient learning of light metal castings for defect detection in X-ray imaging. The motivation comes from the high effort in time and costs which is currently required to set up new objects or parts. To overcome this drawback, on the one hand, we want to reduce the complexity for the user by applying neural networks for defect detection. On the other hand, we try to use as much data from the simulation as possible instead of real data which is costly to collect. The performance of the investigated approaches is evaluated using real-world data from wheel inspection. We show that training on simulated data only is inferior to training with costly real world data. Combining both types results in the best performance and the closer the simulated data matches the real-world, the better the performance.

Keywords: X-ray, Automated defect recognition, Object detection, Domain adaptation, Simulated training data

1 Introduction
The fully automated inspection of cast components with 2D X-ray projections (as shown in Figure 1) is considered a solved task using classical image processing algorithms [1]. However, the effort required to set up new components or component variants is not inconsiderable and requires the user to have a deeper understanding of the functionality of the algorithm, a high level of training and a good experience in non-destructive testing with the help of x-ray inspection. In foundries, a wide variety of components (wheels, components of car engines, etc.) are usually produced using different manufacturing processes. It is necessary to inspect these components for defects in or after production. The high manual setup and staff training effort that is required to inspect new wheel types using 2D X-ray imaging states a major challenge.

Machine Learning (ML) methods have the potential to address these shortcomings. So, the overall goal of this work is therefore to develop a methodology based on neural networks to efficiently train models for new wheel types. Specifically, we want to achieve that ML methods can be used for defect detection early after the start of production as well as cost-efficiently and preferably by simulation. The network used will be trained mainly on simulated data. In contrast to real training data, these can be generated at low cost. Furthermore, it should be investigated which quantity ratio of simulated and real training data is necessary to achieve a given detection rate.

2 State of Art
The fully automated inspection of cast components with 2D X-ray projections is considered a solved task with classical image processing algorithms [1]. E.g., Fraunhofer EZRT offers the ISAR software for this purpose. However, the effort required to set up new components and component variants is not inconsiderable and requires the user of the software to have a deeper understanding of how the algorithms work and often a high level of training. The setup of a new type with ISAR takes an average of about four hours for a user with expert knowledge.
With the "Sentin-Explorer" and the solution of the company "Chooch", ML-based products already exist, with the help of which X-ray projections can be automatically checked for defects [2] [3]. To the best of our knowledge, however, no exact figures on
Detection quality (precision, recall, etc.) as a function of image quality and defect size have yet been published. The use of neural networks to detect irregularities in medical radiographic projections has already been investigated by several groups [4], and AI-based approaches have been shown to achieve the detection and classification rates of inexperienced physicians. First feasibility studies for the detection of defects in X-ray projections of cast components using neural networks exhibit promising detection rates [5-7]. Classification, object-detection, and segmentation architectures have all been used in these studies. To the best of our knowledge, all published architectures have been trained and evaluated either exclusively on real X-ray projections [5-7] or on real X-ray projections with simulated defects [11].

On the other hand, domain adaptation based on simulated training data for the application on real image data has already been explored in different domains. Besides the most common application in the field of autonomous driving [12], domain adaptation has been applied also in robotics [13] as well as in classical image recognition [14]. These studies show that simulated data alone is not sufficient to train models for application on real data but can be a sound asset to reduce the cost-effective generation of data with only slightly worse results [15]. However, such approaches are also already being used in the field of imaging quality investigation [16].

3 Data and Simulation

Training a machine learning model to detect defects requires a large amount of annotated images. Managing and annotating defects in real X-ray casting images is a very expensive task - as such images are not largely available due to proprietary issues and labeling casting defects on those X-ray images requires expert domain knowledge. As the main objective of this paper is to investigate, to what degree the use of simulated data can reduce the need for costly annotated real X-ray images, we propose the following setup. A set of aluminium alloy wheels are collected and CT scanned with a high-resolution CT system. We leverage this resource to create a database only of real defects. Based on this defect database, a simulation pipeline is proposed to produce data with different degrees of sophistication.

An annotated real data set of X-ray wheel-casting images has been also prepared to test and validate our trained model. A total of 4443 real X-ray images are acquired from the inspection process of light aluminium alloy wheels from our industry partner [17]. To narrow the problem scope, only one specific wheel type is used to acquire the images. In this sample, only wheel spokes are considered from various rotated positions, using Comet MXR-160/20 as the X-ray source and XEye-4020 as the detector. The available images contain a variable number of imperfections. The images were labeled in a two-step process. At first, Automatic Defect Recognition (ADR) algorithms [18] are used to detect defects. The ADR algorithms are parameterized in such a way that all relevant anomalies are detected. The algorithms also detected structures in objects which are not real defects (false positives). Thus, all produced labels are checked manually in the second step. This manual true-false labeling is reviewed again to minimize human error. Figure 2 illustrates the proposed simulation chain. Already existing software tools to create X-ray projections are used in this simulation process to create training data with defects at low cost.

In the simulation chain, the same type of wheel is used as in the available real data. Our industry partner [17] provided one reference wheel (without any defects) and 23 defective wheels of light Aluminium alloys. A geometric model of this defect-free reference wheel is created by extracting the surface from reconstructed high-resolution CT scanned data. This surface-extracted 3D wheel model (Figure 3a) is used as our reference wheel model.

Next, the real defects found in the provided defective wheels, using the VGSTUDIO MAX [19] defect detection method, and their geometric descriptions (triangular meshes of defect surface) are collected in a database (Figure 3). Parameters for detection were set carefully to account for smaller defects that also come with a higher rate of detecting false positives. These 3D defects are then manually validated by domain experts to remove all the false positives. The heat-map image (Figure 3b) shows the distribution of the collected real defects on the wheels. Next, a method is developed to insert a given number of defects fetching from the
database into the 3D reference wheel model at randomly selected positions. Before inserting into the model, the individual defect is also augmented randomly, in terms of certain scaling and rotation. Through this random defect augmentation, our method can simulate a wide range of real-world-like scenarios – from a small group of porosities to moderate-sized defects.

This provides the opportunity to create and save a large number of 3D wheel models with the inserted random shape of defects at very low cost. Additionally, the 3D model of only the inserted defects is also saved together. The information regarding each defect’s position and size allow for generating object-detection suitable labels in terms of axis-aligned bounding boxes with almost zero cost. To make the 3D models usable for image-based machine learning, an industry-standard X-ray simulation tool (aRTist [20]) is used to create 2D projections of the simulated defect wheels. The source spectrum and the detector parameters are calculated according to the used source and detector system in capturing real images. The FOD and DOD are also maintained according to the real-life setup. Source and detectors are rotated and positioned together to create multiple projections from one single 3D wheel model so that it covers the complete wheel. To emulate the real-world positioning of a wheel, a random delta in the wheel positioning is introduced while rotating the source and the detector. Examples for the final 2D training images can be seen in Figure 5. The red line profiles display the change in the grey scale values and represent the position of the spokes (negative peaks). Comparing the two line profiles indicates that there is a difference between the simulation and real-world images, as the colour change is a lot fuzzier for the real X-ray image. We evaluate in section 5.1 if the existence of such differences (domain shift) limits the potential of using simulated data for training a model that is tested on real-world data.
Our setup excels in producing both 2D and 3D models with high and sensitive defect quality together with almost free-of-cost automatic defect annotation. As we are in control of the degree of augmentation, four different sets of simulated data are produced. The characteristics of these datasets are described in Table 1.

<table>
<thead>
<tr>
<th>Simulated data set ID</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Contains random samples from the original defects without any defect augmentation. Only the wheel spokes are considered.</td>
</tr>
<tr>
<td>+ Defect Augmentation</td>
<td>Default settings. Original defects from the database are augmented through random scaling (factor from 1 to 3) and random rotation before being simulated into a reference wheel.</td>
</tr>
<tr>
<td>+ Wheel Rotation</td>
<td>Default + Defect Augmentation settings. To emulate the real-world positioning, a random delta in the wheel positioning is introduced while rotating the source and detector.</td>
</tr>
<tr>
<td>+ Complete</td>
<td>Default + Defect Augmentation + Wheel Rotation settings. Additionally, images are taken from the complete wheel (spokes, rim and hub)</td>
</tr>
</tbody>
</table>

Table 1: Explanation of the four different simulated data sets.

4 Experimental Setup

In this section, we will describe the models, data and metrics used in our experiments. In all of our experiments, we train a Faster R-CNN with the same hyperparameters to detect defects in the X-ray scans of aluminium wheels. We are investigating how well we can get a model to perform on real-world data by varying the amount of real-world and simulated training data.

4.1 Detection model

In our experiments, we use the Faster R-CNN [21] for object detection. This model is comprised of two modules: a region proposal network and a detector module that classifies the proposed regions. Both modules share layers in order to facilitate shared computation and the whole architecture can be trained end-to-end [22]. The model is trained via a weighted combination of a regression loss $L_{reg}$ quantifying the quality of the proposed regions and a classification loss $L_{cls}$ that judges the prediction quality regarding the classification of regions.

We use a VGG16 architecture [23] pre-trained on ImageNet [24] as the backbone for the Faster R-CNN. We train the full network using SGD with momentum set to 0.9 and a weight decay of 0.0005 and a one-cycle learning rate scheduler [25] with an initial learning rate of 0.004 and a max learning rate of 0.008. We set the batch size to 12 and resize the images such that the shorter side has a length of 500 pixels. All our experiments are repeated over 3 different random seeds. As we are using different sizes of datasets for training we fix the number of training steps to 75,000 for all experiments.

4.2 Metrics for object detection

To decide whether a predicted bounding box is correct according to a ground truth bounding box the Intersection over Union (IoU) is used. It is defined as the intersection of the predicted bounding box and the actual bounding box divided by their union. The prediction is considered to be correct if the IoU is above some threshold. A widely used metric for evaluating the performance of object detection models is the mean average precision (mAP). This metric tries to account for the trade-off between precision and recall by summarising the precision-recall curve into an Average Precision (AP) for each class for a given set of IoU thresholds. While mAP is usually defined as the mean over all class AP scores, we omit the mean over classes and use the AP of the single object class that we want to detect. To calculate this AP we use 50% as the threshold for IoU as used in the Pascal VOC Challenges [26] and denote this metric as AP@50. Because the cost of missing a defect is higher than a false alarm, we also consider recall in our experiments. While the exact overlap of the predicted and ground truth bounding box is not as important as the rough location of a defect we consider the recall for the IoU thresholds 20%, 30%, 40% and 50% and denote them as $R@20$, $R@30$, $R@40$ and $R@50$. The average over these recall scores is denoted as Avg. Recall.

4.3 Datasets

As we are interested in the capability of the defect detection models in real-world applications we have gathered a dataset with real-world X-ray scans. Defects were then detected using an existing defect detection pipeline and all identified defects were inspected by a human expert. We split this dataset into 5 folds of equal size and use one fold as a holdout test set. We then define Full Train as all of the 4 training folds and Part Train as using only one of the folds. The statistics of the dataset can be seen in Table 2.

For using the simulation data in our experiment we utilize the four different sets of simulated data described in Section 3. Each of the simulation datasets was split into 80% train data and 20% test data. The statistics of these datasets can be seen in Table 3.
5 Results

We have run different configurations of the train data and in this section, we provide the results of our experiments. An overview of the different dataset configurations and their performance on the real-world test data can be found in Table 4.

<table>
<thead>
<tr>
<th>Simulation Data</th>
<th>Real Test Performance</th>
<th>Real Test Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AP@50</td>
<td>R@20</td>
</tr>
<tr>
<td>Default</td>
<td>0.00</td>
<td>2.10</td>
</tr>
<tr>
<td>+ Defect Augmentation</td>
<td>0.30</td>
<td>17.67</td>
</tr>
<tr>
<td>+ Wheel Rotation</td>
<td>0.33</td>
<td>16.10</td>
</tr>
<tr>
<td>+ Complete</td>
<td>0.35</td>
<td>10.65</td>
</tr>
<tr>
<td>✓ Standard</td>
<td>5.40</td>
<td>52.30</td>
</tr>
<tr>
<td>✓ + Defect Augmentation</td>
<td>6.43</td>
<td>52.73</td>
</tr>
<tr>
<td>✓ + Wheel Rotation</td>
<td>6.67</td>
<td>53.07</td>
</tr>
<tr>
<td>✓ + Complete</td>
<td>8.00</td>
<td>53.67</td>
</tr>
<tr>
<td>✓ Standard</td>
<td>10.53</td>
<td>70.33</td>
</tr>
<tr>
<td>✓ + Defect Augmentation</td>
<td>12.97</td>
<td>70.13</td>
</tr>
<tr>
<td>✓ + Wheel Rotation</td>
<td>13.03</td>
<td>68.93</td>
</tr>
<tr>
<td>✓ + Complete</td>
<td>15.07</td>
<td>71.03</td>
</tr>
<tr>
<td>✓</td>
<td>3.43</td>
<td>43.87</td>
</tr>
<tr>
<td>✓</td>
<td>11.03</td>
<td>65.50</td>
</tr>
</tbody>
</table>

Table 4: Performance of the Faster R-CNN trained on different data configurations and evaluated on the real-world test data.

5.1 Using only simulation data

To understand how suitable the simulation data is for training a model that will perform well on our real-world data we run the Faster R-CNN on simulation data only and evaluate the model on the real-world test data. In Table 4 we can see that all the models trained on the simulation datasets are performing drastically worse compared to the models trained on the real-world train data. Although the performance on the real-world data is low, we can observe in Table 5 that the performance of a model trained on the Wheel Rotation and Complete simulation datasets obtains a high performance when evaluated on the test set from the corresponding simulation test datasets. This is a clear indication that the domain shift between the simulated data and the real-world data has a negative influence.

<table>
<thead>
<tr>
<th>Simulation Data</th>
<th>Real Test</th>
<th>Simulation Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AP@50</td>
<td>Avg. Recall</td>
</tr>
<tr>
<td>+ Wheel Rotation</td>
<td>0.33</td>
<td>9.50</td>
</tr>
<tr>
<td>+ Complete</td>
<td>0.35</td>
<td>6.35</td>
</tr>
</tbody>
</table>

Table 5: Performance on the simulation test sets and the real-world test set of Faster R-CNN trained on the Wheel Rotation and Complete simulation dataset.
5.2 Simulation and Real-World Data

We can observe that combining the simulation data with real-world data in the training intuitively also improves the real-world performance of the models. While both the default simulation as well as the addition of defect augmentation, and wheel rotation has a similar performance we can see that the complete simulation data in combination with real-world data results in the best performance of all the different types of simulation data. In Figure 6 it is made clear that the complete simulation data has the best performance both in combination with part of and the full real-world train data. We hypothesize that the better performance of the complete simulation data can be attributed to both the variations captured in the simulation as well as a regularizing effect from being trained on the hub and the rim views of the wheel in addition to only the spokes in the other datasets. The improvement of the models when adding real-world data shows that the models are able to better handle the domain shift from simulated to real-world data when seeing data from the real-world during training.

In Figure 6 we can also observe that the combination of simulated data and real-world data is better than training on the real-world training data alone. Although we can utilize as much simulation data as we want, we can see that the differences in the amount of real-world data still have a big influence on the performance.

![Figure 6: Performance according to AP@50 and Avg. Recall on the real-world test data for the different simulation data combined with varying amount of real-world train.](image)

6 Conclusion

In this paper, we have explored a flexible technique for creating simulation data that can be used to train an object detection neural network for detecting defects in real-world data. We can conclude that it is possible to use the simulation data to detect defects, but the performance is drastically improved when utilizing real-world data during training due to the domain shift between the simulated data and real-world data. We believe that our results are promising for future works using various techniques for domain adaptation to reduce the need for costly labelled data even further.

We can also see that the various improvements of the simulation data introduced in this work are also reflected by improvements in the performance, which encourages further work into improving the simulation data and thereby minimizing the domain shift between the simulated data and the real-world data.

Acknowledgements

The authors would like to thank the company BORBET Austria GmbH for providing a sufficient number of wheels for scanning and tests.

This work was supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of BAYERN DIGITAL II (20-3410-2-9-8) as well as the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A.
References


