Generating Meaningful Synthetic Ground Truth for Pore Detection in Cast Aluminum Parts

Patrick Fuchs¹,², Thorben Kröger², Tobias Dierig², Christoph S. Garbe¹

¹Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Heidelberg, Germany, e-mail: patrick.fuchs@iwr.uni-heidelberg.de
²Volume Graphics GmbH, Heidelberg, Germany

Abstract
Labeling a data set is a tedious task, especially when identifying small pores in an artifact-prone three-dimensional computed tomography (CT) scan of a die cast. Modern deep learning algorithms have the ability to increase the quality and speed of automated inspection. However, they crave vast amounts of labeled data. Thus, we demonstrate a method to simulate a realistic CT-data set which allows ground truth labels to be derived automatically. We place procedurally generated pores inside lifelike material samples, yielding virtual aluminum parts. Using properties of real materials during the simulation, we are able to create scans comprising the typical CT-artifacts which impede the detection of defects, especially noise, beam hardening, and ring artifacts. To validate the realism of this data set, we use the simulated data to train different defect detection algorithms, including convolutional neural networks, and measure their prediction performance on real data showing the aforementioned artifacts. The corresponding ground truth labeling was derived from scans of higher quality of the same parts.

Keywords: defect detection, semantic segmentation, computed tomography, machine learning, deep learning, simulation

1 Introduction
In safety-relevant areas it is important to ensure that each die cast is free of critical flaws like pores and shrinkage cavities. CT-scans allow us to check the inside of a component without destroying it. Yet, the inspection is not an easy task. Artifacts arise due to the underlying physics behind the scanning process which give the auditor a hard time finding defects. Given scans of sufficient quality, image processing methods may be used for defect detection, e.g. the different defect detection modules commercially available in VGSTUDIO MAX (Volume Graphics GmbH). Modern machine learning-based algorithms promise to increase the accuracy of automated inspection, even given scans which trade image quality for faster scan times. However, to train such algorithms a vast amount of labeled data is needed. Manually labeling CT-scans might be even more tedious and expensive than labeling images as it involves a third dimension. Especially, for small objects, like defects, it is hard to tell noise apart from real target structures—even for experts.

In other fields it was shown that synthetic training sets have the ability to improve classification and segmentation results. Richter et al., for example, automatically labeled the road scenes of a recent computer game to improve the training of scene understanding algorithms [2]. Gaidon et al. even created their own virtual world for this task [3], and Shrivastava et al. used patches of computer animated eyes to train their gaze estimation algorithm [4]. The idea of training on synthetic data has already led to successful products, e.g. the pose-estimation used in Microsoft Kinect was trained using synthetic data [5]. In the world of CT-scans simulated data has been used to verify quality measures used in dimensional CT [6] and to develop systems for scatter correction [7].

Therefore, we propose a method to automatically generate a realistic, simulated data set of CT-scans of defective aluminum die casts. To validate the realism of this data set and its ability to serve as training data for novel, deep learning-based algorithms, in our case a fully convolutional networks, we show that defect detection algorithms trained only on our simulated data yield a similar prediction performance on both real and simulated data. The prediction performance is evaluated both in terms of the detection rate of the classifiers which are trained to distinguish between defective and defect-free structures in a CT-scan, and in terms of the quality of the segmentation mask they produce.

2 Simulating Defective Parts
The proposed simulation pipe-line consists of three major parts: (1) the procedural mesh generation producing defective virtual die casts, (2) the simulation which generates the CT-scans containing a broad variation of artifacts in different levels of severity and, finally, (3) the reconstruction and post-processing creating the final data set and corresponding defect masks. In the following section, we will provide more details about the individual steps and the tools they involve.

2.1 Virtual Cast Aluminum Parts
To be able to procedurally generate a large amount of different geometries for simulated die casts, we define a set of basic geometries such as spheres or cubes, and a set of modifiers like Boolean operations to combine these basic geometries. We compose the virtual die casts by randomly choosing from basic geometries and modifiers. Furthermore, each basic shape and
of a defect is defined by the diameter of its bounding sphere. We create defects ranging from 0.2 mm up to 3.0 mm. In Figure 2a 
we have a repulsive force $f_r$ (Eq. 1) and an attracting force $f_a$ (Eq. 2) calculated from the minimum 

between two nodes $u$ and $v$ we further check if they are separated by the boundaries of the casting shape, as such separated defects do not influence each other. Between two nodes $u$ and $v$ we have a repulsive force $f_r$ (Eq. 1) and an attracting force $f_a$ (Eq. 2) calculated from the minimum 

Figure 1: A brief overview of the different shapes of the virtual die casts and how these shapes affect the scan.

Each modifier has a defined set of parameters to further vary the appearance of the final virtual die cast. The total edge length of about 40 mm of the combined and modified shapes is chosen to resemble material samples taken from larger elements. As unprocessed casts usually do not have even surfaces and sharp edges, we use a post-processing step to refine the generated models: We subdivide large surfaces and slightly distort the vertices to simulate an uneven surface and additionally round off the edges by iteratively beveling them. Yet, we leave some parts untouched and additionally include more complex forms like screw threads to aim for processed cast aluminum parts, too. Figure 1a shows four examples of virtual die casts created with this method.

Since the shape of an object affects the resulting image, we choose the shapes to be complex enough to introduce a variety of geometry-based artifacts. One of the major problems are, for example, streaking artifacts (Figure 1b [8, 9]). Those bright and dark bands result from portions of the beam having varying penetration lengths. Further, due to the scan setup, we obtain cone-beam artifacts for large, flat surfaces orthogonal to the rotation axis. Of course, the severity of these artifacts can be mitigated by tilting the object during the scan process. Nevertheless, we take no measures to reduce artifacts as we prefer having more complicated data to be able to properly train reliable pore detectors.

Inside the virtual die casts we place procedurally generated defect meshes which have typical shapes of air and gas inclusions, cavities, and (large) cracks [10]. The inclusions are modeled as simple bubbles with slight deformations, whereas the cavities are characterized by sharp and pointy edges and corners. Cracks are flat and sharp, winding through the data set. To obtain a large number of different defect models, the defects are extruded from a simple basic shape using randomly chosen lengths and iterations. Afterwards, we randomly add deformations to the defect. Here, each defect type has its own basic shape, extrusion rules, and post-processing instructions: Inclusions are extruded from a cube and primarily smoothed using only a little distortion; cavities, in contrast, are based on varying cuboids whose vertices are heavily distorted to obtain the pointy edges; finally, the cracks are created by extruding only the narrow faces of a flat cuboid whereby it is important to avoid self intersections. The size of a defect is defined by the diameter of its bounding sphere. We create defects ranging from 0.2 mm up to 3.0 mm. In Figure 1a we show three examples of each defect type.

Besides pores which potentially can be detected as individual instances, we add larger areas of lower density to simulate structural defects where individual pores cannot be resolved. These areas are modeled similarly to the air and gas inclusions and can co-occur with detectable pores, of course.

We distribute the defect meshes using an adaption of a force-based graph drawing algorithm [11] to reduce overlaps and prevent the defects from touching the surface of the virtual die cast. This allows us to separate surface defects into a different defect class. So, we can implement special consideration on surface defects like increasing their minimum size and restricting their shape. For this purpose we do not only simulate attracting and repulsive forces between instances of pores, but add repulsive forces between pores and the surrounding shape boundary given by the virtual die cast (Figure 2b). To apply the force-based algorithm, we treat every defect as node of a graph in three-dimensional space. Then, we randomly add connections between the nodes, to enable the formation of defect clusters. We grant bigger defects a higher connectivity than small defects. Hence, big defects are able to rally a lot of small defects, whereas small defects form more homogeneous structures among one another. As the original algorithm treats each node as a point without any extent, we need to add a minimum distance to the force calculation to prevent overlaps. To compute the forces between defects and the wall, we project 18 rays equally distributed in all directions and test whether they hit the shape boundary within a given distance. For the calculation of the forces between two defects we further check if they are separated by the boundaries of the casting shape, as such separated defects do not influence each other. Between two nodes $u$ and $v$ we have a repulsive force $f_r$ (Eq. 1) and an attracting force $f_a$ (Eq. 2) calculated from the minimum
distance \( d_{\text{min}} = r + r + b_{\text{min}} \), the preferred distance \( d_{\text{pref}} = r + r + b_{\text{pref}} \) and the actual distance \( d_{\text{actual}} \), where \( b_{\text{min}} \) and \( b_{\text{pref}} \) are computed depending on the radius \( r \) of the bigger defect and an user given factor. It is important that these biases satisfy \( b_{\text{min}} \geq b_{\text{pref}} \). For the repulsive force between a defect and the shape boundary (Eq. \( 3 \)) things are a bit different: If the defect is outside the virtual die cast, we let the force grow unbounded as we want to push the defect back inside. To prevent corner cases we filter all pores afterwards and remove those touching the surface or being outside the virtual die cast.

\[
f_r = \frac{(d_{\text{pref}} - d_{\text{min}})^2}{d_{\text{actual}} - d_{\text{min}}} \\
f_a = \frac{(d_{\text{actual}} - d_{\text{min}})^2}{d_{\text{pref}} - d_{\text{min}}}
\]

After placing pores inside a virtual die cast, we add more defects on its surface in particular. We do this in a separate step to be able to distinguish between inner and surface defects during the training processes. To generate the right ground truth mask later on, we need to compute the intersection of the generated defects and the virtual die cast to prevent any overhanging defects and false positive labels in the surrounding air.

For the simulation and the ground truth generation it is important that all generated meshes are watertight, as both use ray-based methods which need to be able to precisely determine intersections between the ray and the mesh. As we cannot assure that after all Boolean operations are performed all pore meshes are still watertight, we remove non-watertight parts from the final mesh.

This defect positioning algorithm is not restricted to our set of virtual die casts by any means, but can be used to place defects inside arbitrary STL models. However, it is only a rough but fast abstraction and has no physically sound basis.

### 2.2 The Simulation Process

The technical properties of a CT-setup span a huge parameter space: We can change the voltage of the source and use different filters to change the spectrum; we can use detectors of different pixel size to vary the spatial resolution; we can influence the focal spot by increasing the power, etc. From an image processing point of view, however, many of these configurations yield quite similar CT-scans. Hence, we decided to reduce this high-dimensional parameter space to an artificial three-dimensional artifact space for which we vary the underlying physical parameters to change noise, beam hardening, and ring artifacts. The individual projections of the CT-scan are then simulated using aRTist [12], which is also used in [5].

We fix the parameters relevant for the spatial resolution, so that all defects we want to observe can be observed barely. We set voltage and power to 225\( \text{kV} \) at 225\( \text{W} \) and obtain a stable focal spot size of 225\( \mu \text{m} \times 225\mu \text{m} \). These parameters were chosen by comparing the data sheets of several vendors of X-ray tubes. The detector is fixed to 1000 \( \times \) 1000 pixel with a pixel size of 200\( \mu \text{m} \times 200\mu \text{m} \). The distance between source and detector is 1000\( \text{mm} \); the distance between the source and the object is 450mm. The reconstructed voxels therefore have a size of 110\( \mu \text{m} \) cubed, while the diameter of the smallest defect is 200\( \mu \text{m} \).

Then, the artifact space is formed by varying the following parameter: To change the strength of beam-hardening we change the thickness of the copper prefilter which filters the soft portions of the raw spectrum of the source. We use 1\( \text{mm} \) for low, 0.5\( \text{mm} \) for medium, and no filter for severe beam-hardening. To change the content of noise, we change the exposure time so that the detector receives once half the dose and once twice the dose of the initial configuration. Finally, to change the severity of ring artifacts we use three bright images with different amounts of noise during reconstruction.

Along each dimension of the artifact space we simulated three grades, yielding 27 scans of different artifact load per virtual die cast. For the simulation, we virtually fill our injection molds with a widely used aluminum alloy: EN AW-2014, which consists
of about 5% copper, leveraging the effects of beam-hardening. Moreover, we simulate only 720 projections, so we obtain minor aliasing artifacts due to an undersampling. To take X-ray scatter into account, we use a Monte Carlo method to reproduce this effect in our simulations.

2.3 Reconstruction and Post-processing
For an automated CT-reconstruction, we pass the scan configuration and the simulated projections to VGinLINE (Volume Graphics GmbH) and its reconstruction algorithms in a batch-processing setup. Knowing the geometry of the scan setup allows us to exactly align the defect meshes with the simulated scan, which is necessary to produce a precise per-voxel ground truth. We produce this ground truth by evaluating \( n \) sample points along a regular grid inside each voxel, so that we can approximate with arbitrary precision how much of each voxel lies within a defect mesh. Further, we can arbitrarily change the spatial resolution, by subdividing each voxel and sampling inside each “sub-voxel”. The ground truth generation procedure has to be performed only once per virtual die cast, as the mask is the same for all configurations in the artifact space.

Using this method we created a data set comprising 25 different virtual die casts in 27 different artifact configurations, totaling in 675 CT-scans, each of which has about 550 defects, and a total size of 1.35 TB. Figure 3 shows a few examples of the simulated data set (top row) in comparison to real data (bottom row).

3 Evaluation
As a proxy for evaluating the realism of our simulated data set, we turn to the prediction performance of defect detection algorithms. Prediction performance, in this case, refers to the segmentation quality. The algorithms are trained solely on our simulated data and are then able to achieve a similar performance on real data. It is important to note that when talking about evaluating defect detection we do not presume to separate “critical” from “non-critical” defects, but to tell apart artifacts in the CT-scan from real, physically present defects.

3.1 Acquisition of Real Data
Obtaining ground truth for real data is not only a laborious task, but also a tricky one: Each expert has a different understanding of how many voxels in the CT-scan belong to a pore or whether a pore is worth considering it as such at all. However, to evaluate the segmentation results, real data with an accurate ground truth is necessary. We, therefore, have a high-quality scan of different cast aluminum parts made for the labeling task and a “normal” scan of the same object for evaluation. Then, we ask eight experts to manually segment all physically present pores in the high-quality scan, which they can clearly recognize as such. The segmentation masks are merged to a single ground truth mask. The idea is that in the high-quality scan creating a segmentation mask is easier, because it contains less artifacts, while the “normal” scan provides the intended challenges for the defect detection algorithms. The necessary quality is achieved by (1) increasing the exposure time to reduce the image noise and (2) increasing the spatial resolution to better resolve smaller defects. To evaluate the segmentation results of the defect detection algorithms, we use the segmentation mask created on the high-quality scans for comparison. Figure 4 shows the same slice of a high-quality scan and its according “normal” scan. The necessary quality is achieved by (1) increasing the exposure time to reduce the image noise and (2) increasing the spatial resolution to better resolve smaller defects. To evaluate the segmentation results of the defect detection algorithms, we use the segmentation mask created on the high-quality scans for comparison. Figure 4 shows the same slice of a high-quality scan and its according “normal” scan. The necessary quality is achieved by (1) increasing the exposure time to reduce the image noise and (2) increasing the spatial resolution to better resolve smaller defects. To evaluate the segmentation results of the defect detection algorithms, we use the segmentation mask created on the high-quality scans for comparison. Figure 4 shows the same slice of a high-quality scan and its according “normal” scan. The necessary quality is achieved by (1) increasing the exposure time to reduce the image noise and (2) increasing the spatial resolution to better resolve smaller defects. To evaluate the segmentation results of the defect detection algorithms, we use the segmentation mask created on the high-quality scans for comparison. Figure 4 shows the same slice of a high-quality scan and its according “normal” scan.

Our two evaluation objects are 40 mm \( \times \) 40 mm \( \times \) 20 mm and 80 mm \( \times \) 120 mm \( \times \) 70 mm big. The first part was scanned using 180 kV at 160 \( \mu \)A with a filter of 0.5 mm copper, the second using 295 kV at 320 mA and a filter of 1.0 mm copper. The high-quality scans comprise 2000 projections and a total scan time of 120 min each, while the “normal” scans are composed of only
9th Conference on Industrial Computed Tomography, Padova, Italy (iCT 2019)

Figure 4: Comparison between the high-quality scan (left) with longer integration time and better spatial resolution, the “normal” scan (middle), and the combined ground truth for this data set as colored overlay (right).

1500 projections taken in 15 min.

When receiving the scans from the service providers, they usually are rotated and shifted against each other. However, for a correct comparison we need them to be aligned precisely. Hence, we use VGSTUDIO MAX (Volume Graphics GmbH) to perform a locally adaptive surface determination of each object and use the registration methods to align the “normal” scan to the high-quality scan. After the registration process we re-sample all expert labels with respect to the “normal” scan. Having all segmentation masks and the “normal” scan in the same coordinate system and the same spatial resolution, we can combine the 8 segmentation mask created by the experts into a single mask. The gradations in the ground truth mirrors the accordance of the segmentation masks (Figure 4). At the center of a defect we observe a high accordance of all segmentation masks while it falls off towards its borders. Instances with low overall accordance are presented to the expert committee again for a final decision. For evaluation we threshold the mask at an accordance of 50%.

3.2 Pore Detection Algorithms

The algorithms, whose results we compare, are (1) a filter-based image processing technique, (2) a downstream classification using hand-designed features (both from [13]), and (3) a deep-learning method using a fully convolutional architecture which is similar to the one used to segment fine structures in images [14], which are challenging in a similar way as our tiny defect structures.

Filter-based Method

Typically, applying a simple threshold for defect detection is not possible as the above mentioned artifacts cause the same gray-value to belong once to a pore and once to an artifact depending on the position in the scan. To deal with such artifacts we need to prepare the data before segmenting it with a threshold. In [13], the authors filter the volume to create a defect-free image. Then, the original image is subtracted from the defect-free image. In this difference image, a threshold is applied to select defect candidates, as large differences (peaks in the image) resemble defects. This candidate selection method can be used as an independent defect detection method. Though, at the transition from material to air this can lead to unintended false positive responses. Therefore, it is recommended to use a cleaned surface mask to limit the search area to the material part.

Results show that the method works well on low noise data, independently form the presence of beam hardening artifacts. Noise, however, causes the results to drop severely. The additional template matching filter proposed in [13] reduces the false positive candidates along ring and streaking artifacts to a minimum. Yet, it filters defects that do not match the size of the template or deviate too much from a round shape. Therefore, we decide to omit it.

Subsequent Classification

The first method works out of the box and has no need for training data. With a subsequent classification which has the possibility to reduce false positive responses, however, the need for training data arises. The second method, with which we evaluate the simulated data set with, is a traditional learning-based approach: The results of the filter-based method are treated as candidates. For each candidate a set of 29 hand-designed features based on its gray-values and its curvature (as described in [13]) is extracted. Finally, a random forest [15] classifier trained on our simulated data decides whether the candidate is a real defect or a false positive alarm. This method is further referred to as “traditional classifier”.

For the candidate selection we use a more generous threshold to obtain more candidates, as the subsequent classifier can filter false positive responses, but cannot correct false negatives. The 29 features comprise 24 based on the gray-values of the candidate, 3 based on its curvature, and 2 based on the differences between original and defect-free image. The 24 gray-value-based features are calculated from the gray-level co-occurrence matrix (GLCM) of the candidate and the surrounding material. The GLCM is a histogram illustrating how often a gray-value is neighbor to another gray-value in a given distance. In total, six GLCMs are
computed for neighboring gray-values in three distances: 1, 2, and 3 voxel apart. The features are composed of energy, contrast, entropy, and standard deviation of these GLCMs. The 3 curvature-based features are computed by combining the principal curvatures obtained from the second derivative of the image to a single shape index. Mean, standard deviation, and entropy of the shape indices of the candidate make up the final features. The remaining two features are the mean gray-value difference and the mean shape index difference between original and defect-free image.

Admittedly, this method involves quite a lot parameters which have to be tuned and each of which has a huge impact on the outcome. Unfortunately, [13] spared some important details in their algorithm which we had to implement to the best of our knowledge. The random forest classifier itself which was trained on the afore mentioned features works very well. The candidates, however, need to be segmented pretty accurately for the feature extraction. Partially segmented defects, for example, tend to obtain a low probability. Moreover, large ring artifacts tend to disturb the classification pipeline as they have similar feature responses as defects.

Fully Convolutional Networks

The initial idea behind creating an artificial defect data set is to have a sufficient amount of annotated data to train a convolutional neural network (CNN) for defect detection. While the first two evaluated methods work per instance/defect, the deep learning-based method yields a defect probability per voxel. In order to create a top-performing defect detector, we combine recent developments and ideas in the field of deep semantic segmentation and object recognition to a modern and strong fully convolutional network (FCN).

Figure 5 provides an overview over the architecture of the FCN: The base is formed by an encoder-decoder model with an additional refinement step in the end as it is used by [14] to separate fine structures like hairs from a cluttered background. We employ the same skip-connections between encoder and decoder as in U-Net and V-Net architectures [16, 17] to tackle the problem of vanishing gradients, on the one hand, and to pass information about fine structures to the decoding layers, on the other hand. Further, we induce more gradient information in deeper layers by using additional up-sampling layers which directly produce a segmentation from the deep layers [18]. During training these intermediate result masks are considered in the loss function. The individual layers of encoder and decoder use three-dimensional convolutions and are organized as residual layers [19, 20]. Each residual layer comprises two convolutional and two batch normalization [21] layers (Figure 5 dashed box). Instead of pooling layers which reduce the spatial resolution following a given rule, we use strided convolutions to learn how to condense the spatial information [22]. Finally, the refinement step combines the output of the decoder, the intermediate outputs, and the original image to the final segmentation mask using four more convolutional layers. During the training process we use dropout layers [23] with a probability of $p = 0.5$ that an element is kept, and augment the training data by rotating, mirroring, and randomly cropping patches from the data set to prevent overfitting to the training set [24]. For training we use patches of $128^3$ voxel so that the complete FCN fits in the limited memory of the GPU. For the learning process we utilize an Adam optimizer [25] to update the weights and start with a learning rate of 0.0001 which is further reduced over time.
Table 1: Evaluation results: filter-based method (1), traditional classifier (2), deep learning-based method (3). The values of the POD 75/75 values are rounded to the next complete voxel.

<table>
<thead>
<tr>
<th></th>
<th>Simulated data</th>
<th>Real data</th>
</tr>
</thead>
<tbody>
<tr>
<td>max. IoU [%]</td>
<td>(1) 75.9</td>
<td>70.5</td>
</tr>
<tr>
<td></td>
<td>(2) 85.1</td>
<td>65.5</td>
</tr>
<tr>
<td></td>
<td>(3) 93.7</td>
<td>85.9</td>
</tr>
<tr>
<td>POD 75/75 [vox]</td>
<td>Simulated data</td>
<td>(1) 11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3) 5</td>
</tr>
<tr>
<td></td>
<td>Real data</td>
<td>(1) 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2) 7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3) 3</td>
</tr>
</tbody>
</table>

(a) Intersection over Union

(b) Probability of Detection

3.3 Results

For the evaluation process we choose two measures: (1) intersection over union (IoU, or Jaccard-index) [26] which allows us to draw conclusions about the quality of the defect segmentation mask, as it is measured per-voxel, and (2) probability of detection (POD) [27] which tells us how many defects of the evaluation set we can find, depending on their size and the contrast resolution of the scan. While IoU is an established measure for unbalanced data in the field of machine learning and semantic segmentation, POD is the standard evaluation tool in the field of non-destructive testing.

Intersection Over Union

Each voxel can either be classified correctly as belonging to a defect (true positive, TP); classified correctly as not defective (true negative, TN); misclassified as belonging to a defect (false positive, FP); or misclassified as not defective (false negative, FN). When measuring the accuracy (Eq. 4) of a classifier, all four categories are considered. CT-scans of cast aluminum parts, however, mostly consist of non-defective voxels. So, a classifier labeling everything as “not defective”, would achieve an accuracy of over 95% already. Therefore, we choose to evaluate the IoU measure instead which is able to deal with highly imbalanced data sets by discarding the true negative category (Eq. 5).

\[
\text{Acc} = \frac{TP + TN}{TP + FP + TN + FN} \quad (4)
\]

\[
\text{IoU} = \frac{TP}{TP + FP + FN} \quad (5)
\]

Table 1a shows the maximum IoU which the different methods achieve on the simulated and on the real evaluation set. We observe that all methods produce comparable results on both data sets. The bad performance of the traditional method on real data is caused by the candidate selection process, which would need separate fine-tuning for individual regions of the evaluation set. We, therefore, reason that the simulated data set resembles real world CT-scans of cast aluminum parts to a level where it can be used for training learning-based algorithms. The differences between simulated and real data can be explained by the dissent of our experts regarding small defects that are present in large numbers and the exact defect boundary (see qualitative analysis).

In Figure 6a we further compare the performance of the different defect detection methods among each other. The IoU is plotted over possible cut-offs of the result output, i.e. the filter response for the filter-based method and the probability for the learning-based methods.

Classifier Diagnostics

Usually, a receiver operating characteristic (ROC) plot is used to visualize the trade-off between sensitivity and precision. For imbalanced data, however, we have the same problem we have with the accuracy and, therefore, better use a Precision-Recall-Curve (PR-curve) [28]. The precision (Eq. 6) measures how many voxels labeled positive (defective) are actually positive. The recall (Eq. 7) measures how many of all positive (defective) voxels were found. The PR-curve for the evaluated methods is
Figure 7: Qualitative comparison between the different methods: (1) filter-based method, (2) traditional classifier, and (3) deep learning-based method. The patches show the accordance to the ground truth: TP (green), FP (blue), and FN (red); TN are omitted.

shown in Figure 6b

\[ \text{Prec} = \frac{TP}{TP + FP} \]  \hspace{1cm} (6) \hspace{1cm} \[ \text{Rec} = \frac{TP}{TP + FN} \]  \hspace{1cm} (7)

Probability of Detection

A drawback of the IoU is that it tells nothing about whether all defects of a given size were found or not. Missing out many tiny defects can have the same effect on the IoU as missing out a single medium sized defect. To tackle this disadvantage we compute the POD, too. A defect is recognized with a confidence of \( \hat{a} = \frac{TP}{TP + FN} \). We compute this confidence level for each defect and cluster them by their equivalent sphere diameter (ESD). For each cluster we compute the mean \( \mu \) and standard deviation \( \sigma \) to estimate a normal distribution \( g \) over \( \hat{a} \). The POD of a defect cluster \( a \) is then computed as integral over \( g \) from \( \hat{a} \) to \( \infty \):

\[ \text{POD}(a) = \int_{\hat{a}}^{\infty} g(\hat{a}) d\hat{a} \]  \hspace{1cm} (8)

with \( \hat{a}_d = 0.75 \) where \( \hat{a}_d \) denotes the minimum confidence for a defect to be counted as found. An important characteristic of the POD is the \( a_{p/\hat{a}_d} \) value indicating the defect size (i.e. its ESD) where all defects can be found with a probability of \( p \) and a confidence of at least \( \hat{a}_d \). To control the false positive alarm rate we evaluate the POD for the threshold for which we receive the best IoU. When plotting the POD over the ESD \( a \) of the defect clusters (Figure 6c), we see that the learning-based methods handle smaller defects better than the simple filter-based methods, as they can better distinguish between noise induced false positive candidates and real defects. This observation becomes clearer when looking at the \( a_{75/75} \) value shown in Table 1b, as the learning-based methods reach 75% detection probability with a confidence of 75% for much smaller defect sizes. Switching to the real data sets we note that the POD in general improves which indicates that our “normal” scans are of slightly better quality than the simulated evaluation scans. The POD of the traditional method, on the other hand, drops significantly due to the drawbacks of the candidate selection procedure.

Qualitative Evaluation

In Figure 7 we show a few qualitative examples comparing the results of the different defect detection methods on simulated and real data. The patches of the simulated data were chosen to give a glance at its broad quality spectrum. From left to right we have a quite homogeneous area in good scan quality, defects near post-processed surfaces like screw threads, defects near heavy ring artifacts around the rotation axis and small defects in a scan containing strong noise. It can be seen that the deep-method has less problems in dealing with noisy data without producing false positive responses. Unfortunately, all three methods fall for the defect like structures in the center of the ring artifacts. The results of the traditional classifier severely depends on the underlying candidate selection method and tends to reject incompletely and brittly segmented candidates completely. For the real data we show different formations of defects from small dispersed pores to large defect clusters combined with structural loosening. We mainly notice that there is more dissent about the exact dimensions of the defects and that the traditional classifier has some problems with defect clusters where defects are too close together.

Besides the parameter simplification, another aspect of our artifact space is that we are able to obtain a few insights concerning the influence of different artifact types on the segmentation results. To examine the influence, we project all results onto a single
Figure 8: Insights from the artifact space: We project all results onto a single dimension of our artifact space and plot mean and standard deviation of the IoU for each of them to see how much influence an artifact type has. For visualization we add a sample patch for each artifact gradation underneath the plots.

Figure 9: Examples showing the possible separation of inner defects (blue) and surface defects (green) on a real 3D-printed part when training the deep method with separated class labels. Note: We did not use any surface determination to prune the results.

4 Conclusion
We have demonstrated how to automatically generate an arbitrarily large synthetic data set of realistic CT-scans of defective cast aluminum parts with a precise ground truth. This data set is able to leverage the training of learning-based algorithms. We have shown that it can be used to train a FCN which is able to segment even small defect structures in the presence of image artifacts, and is especially robust against image noise which we consider to have the strongest influence on the results in this study. Further, we can tell apart surface defects from inner defects in the training set and so are able to train on both classes separately.

This method has the potential to be extended to other materials (e.g. plastics), other types of scans (e.g. helix CT), and other types of defects (e.g. sand inclusions) easily. Furthermore, this procedure can be interesting for other fields of application, too, where a manual labeling of training data is almost impossible (e.g. fiber composite materials).

Acknowledgments
Here’s to all the brave experts digging through over nine billion voxels to manually label 5882 defects for the real data ground truth. Further, we thank all the service providers who want to remain anonymous for preparing the high-quality scans.
References