

Supplementary File:

AI-powered image enhancement for accelerated defect analysis of 3D integration technologies utilized in quantum-information processing

Raphael Wilhelmer¹, Fabian Laurent², Tatjana Djuric-Rissner³, Max Glantschnig², Johann Strasser⁵, Stefan Weinberger⁵, Tobias Herrmann⁵, Clemens Rössler², Peter Czurratis³ & Roland Brunner^{1,*}

¹ Materials Center Leoben Forschung GmbH, Leoben, Austria

² Infineon Technologies AG, Villach, Austria

³ PVA TePla Analytical Systems GmbH, Westhausen, Germany

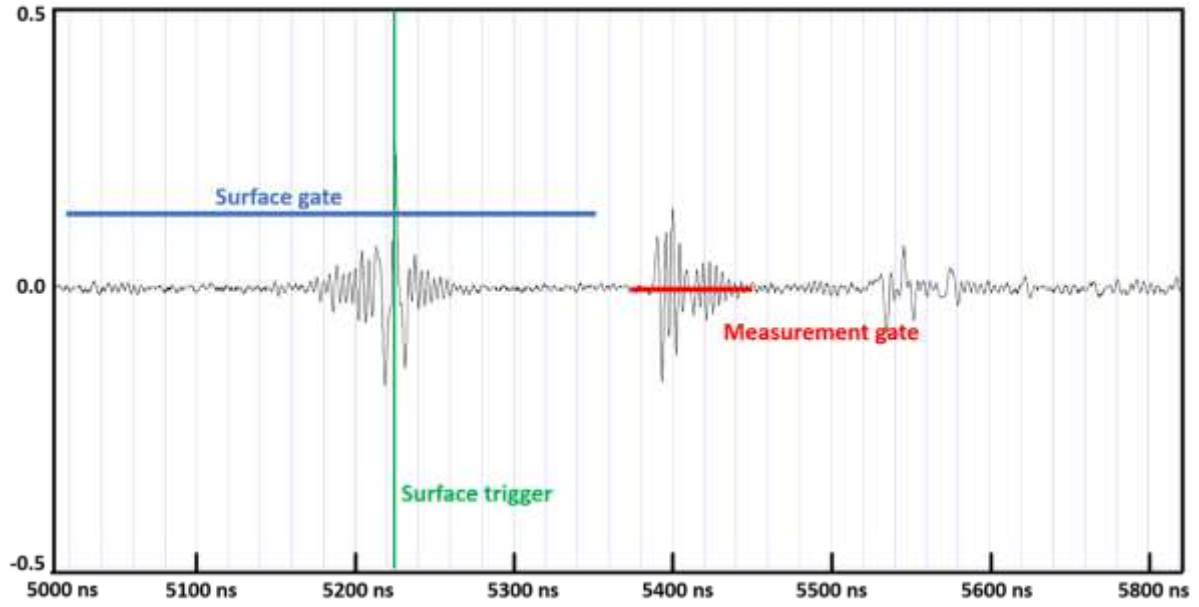
⁵ Infineon Technologies AG, Regensburg, Germany

*corresponding author: roland.brunner@mcl.at

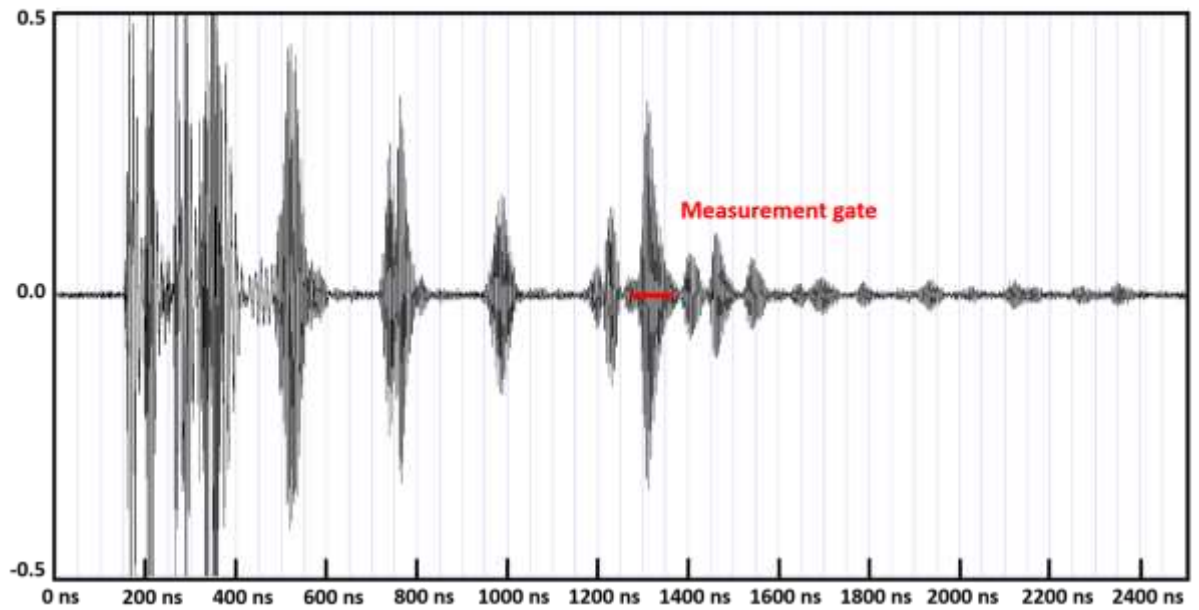
SUPPLEMENTARY NOTE 1: A-Scan images of SAM measurements

In Supplementary Figure 1 representative A-scans for the two different wafers investigated in the main text (bonded and TSV wafer) are shown. These images show the position of the surface trigger and gates. For the tone-burst measurement of the TSVs a long data gate without a surface trigger was applied.

A-Scan for the bonded wafer



A-Scan for the TSV wafer



Supplementary Figure 1: A-scans for the bonded and TSV wafer. Gates and surface triggers are indicated in the image. The intensity of the reflected wave ranges from -0.5 to 0.5.

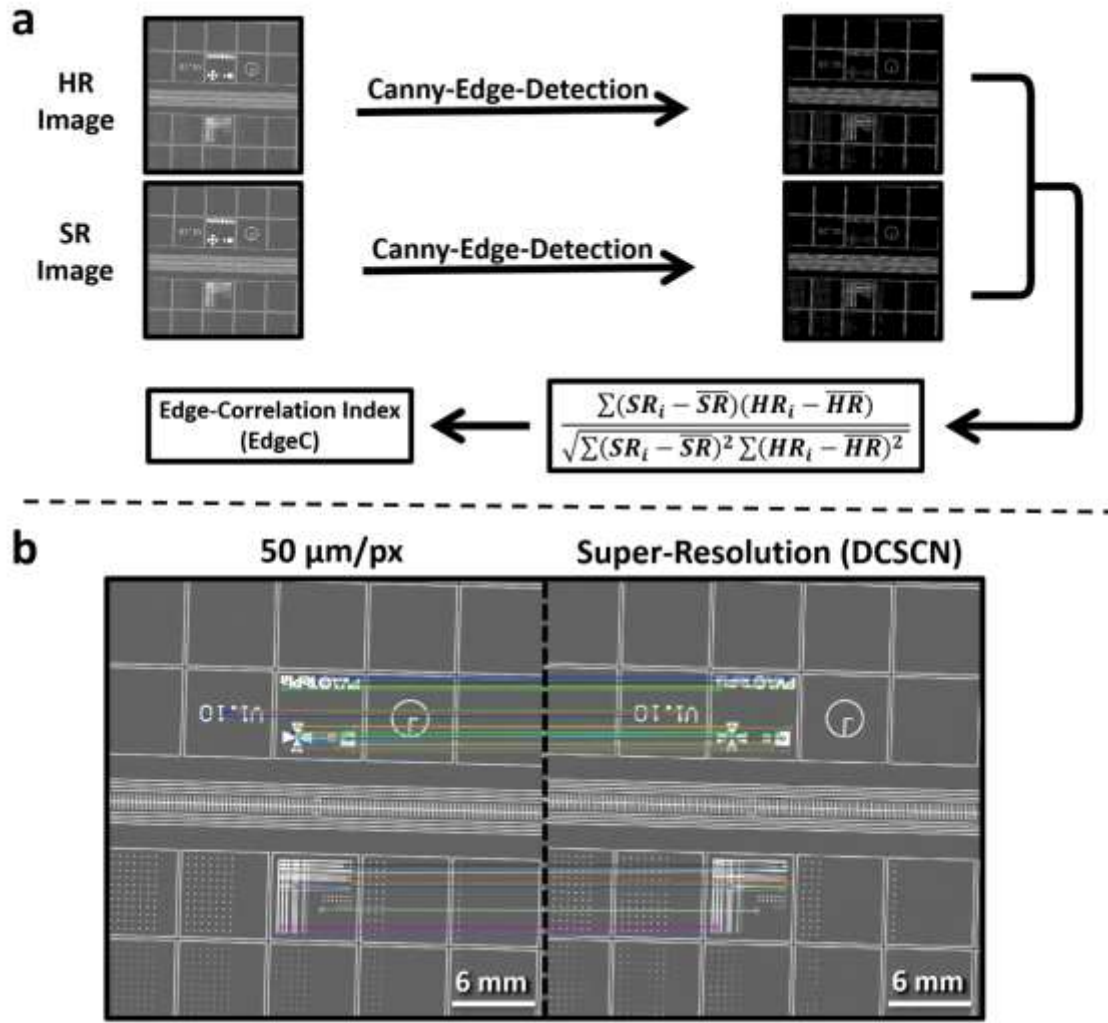
SUPPLEMENTARY NOTE 2: Additional metrics for evaluation

In the main text of the paper we defined two additional metrics to complement the PSNR and SSIM values. These two metrics are the edge correlation index (EdgeC) and the count of congruent features found by a SIFT¹ algorithm.

The procedure of calculation EdgeC is presented in Supplementary Figure 2a: First, we use a canny edge detection algorithm (implemented in the scikit-image python package) to find edges in the high-resolution (HR) and super-resolved (SR) image. The detected edges are then put into the normalized correlation function, which outputs a number between -1 and 1. Values close to one indicate that all edges are correlated, i.e. the reconstruction via SR is good. Values close to zero show no correlation at all. Negative outputs define an anti-correlation and may indicate systematic errors in the model.

An example of feature matching with SIFT is shown in Supplementary Figure 2b. Here, the SIFT algorithm implemented in the OpenCV2 python package is used to calculate important features in both the HR and SR image. Then, similar features are matched and the number of matches is counted. The more matches are found, the better the overall reconstruction quality.

Since both metrics employ different hyperparameters which have to be chosen manually, their exact outputs depend on the user. However, the trends in the results (e.g. that machine learning based SR outperforms classical methods) stayed true under all tested hyperparameters. For the final measurements we chose the parameters displayed in Supplementary Table 1. If not mentioned otherwise, the standard parameters of scikit-image and OpenCV2 are kept.



Supplementary Figure 2: Definitions of metrics used in the main text. (a): Definition of the EdgeC (Edge Correlation) metric. First, a canny edge detection algorithm is used to extract the edges from the HR and SR image. Then, the correlation function between those edges is calculated. The sum runs over all pixels. An output value of 1 means a perfect reconstruction of edges whereas a value of 0 indicates that no correlation exists. Negative outputs correspond to anti-correlations. (b): Schematic of our SIFT-based metric. A SIFT algorithm is used to detect congruent features in the high-resolution image and the reconstruction. The more features are found to be similar, the better the reconstruction. Matches are displayed as colored lines between the 50 $\mu\text{m}/\text{px}$ and super-resolution image.

Supplementary Table 1: Hyperparameters used for canny edge detection and SIFT. Parameters which are not mentioned explicitly are kept to the standard values of the employed python packages.

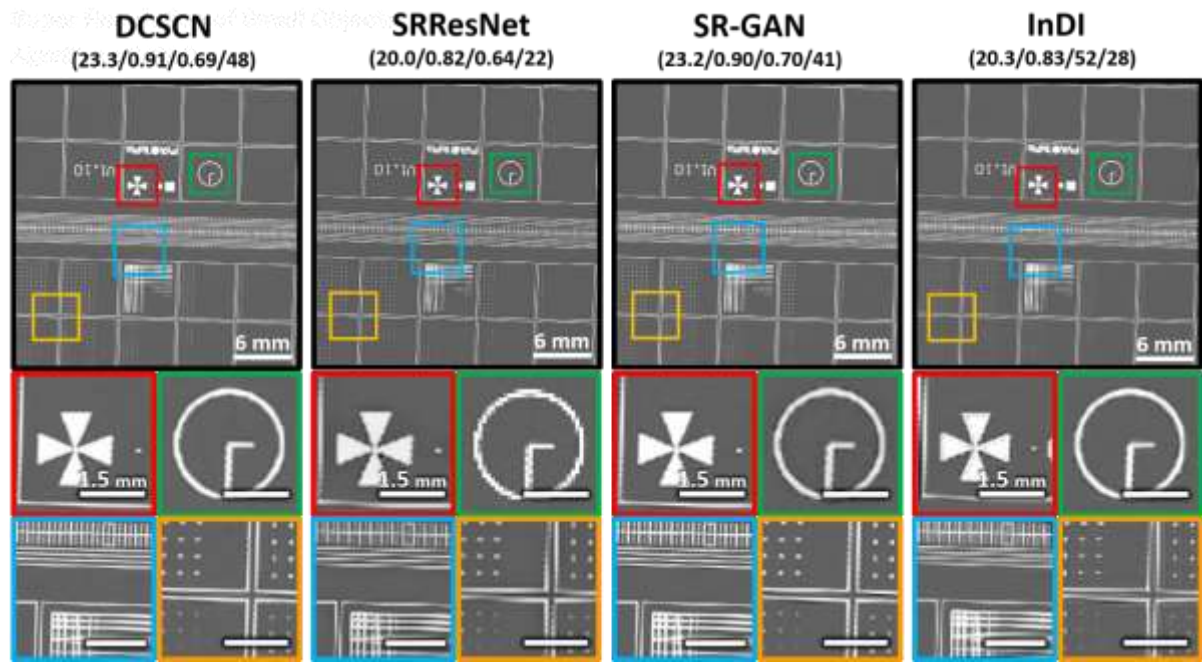
	Low Threshold	High Threshold	nFeatures	Contrast Threshold	Sigma
Canny	0.2	0.6	-	-	-
SIFT	-	-	100	0.06	1

SUPPLEMENTARY NOTE 3: Evaluation of other SR models on real-world data

In the main text of the paper only the DCSCN and InDI SR models are tested under real-world conditions using a wafer with test structures. In this section, we present a comparison between DCSCN, InDI and

the other mentioned SR models under the same conditions. This comparison can be seen in Supplementary Figure 3.

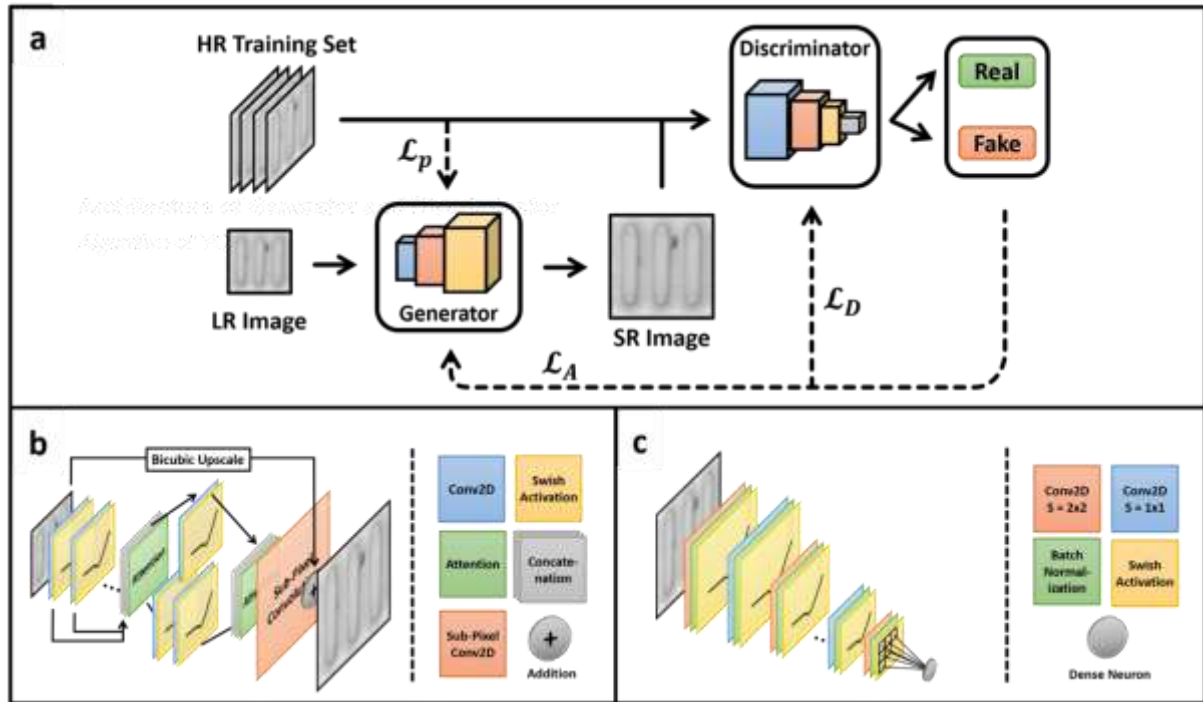
As it is obvious from the images and the metrics, DCSCN performs best under real-world conditions and is closely followed by the SR-GAN model in terms of the metrics. However, when looking at the images, the SR-GAN seems to introduce artefacts including ringing and checkerboard-artefacts. Also, a slight over sharpening is observed. Even though there are no checkerboard artefacts introduced by InDI, the output looks slightly blurrier and is not able to remove jittering (see the blue image). SRResNet, on the other hand, performs poor in general, not even being able to reduce the pixelated nature of the low-resolution input image.



Supplementary Figure 3: Comparison between the different SR models mentioned in the main text. The round brackets contain the metric introduced in the main text: (PSNR, SSIM, EdgeC, SIFT). All metrics show a clear advantage of the DCSCN approach compared to other SR based methods. The images are obtained by upscaling a 100 $\mu\text{m}/\text{px}$ C-SAM image of the test wafer discussed in the main text.

SUPPLEMENTARY NOTE 4: GAN model architecture

The architecture and training procedure of our SR-GAN implementation is illustrated in Supplementary Figure 4a-c. Details on the architecture and training are given in the main text in the Methods section.



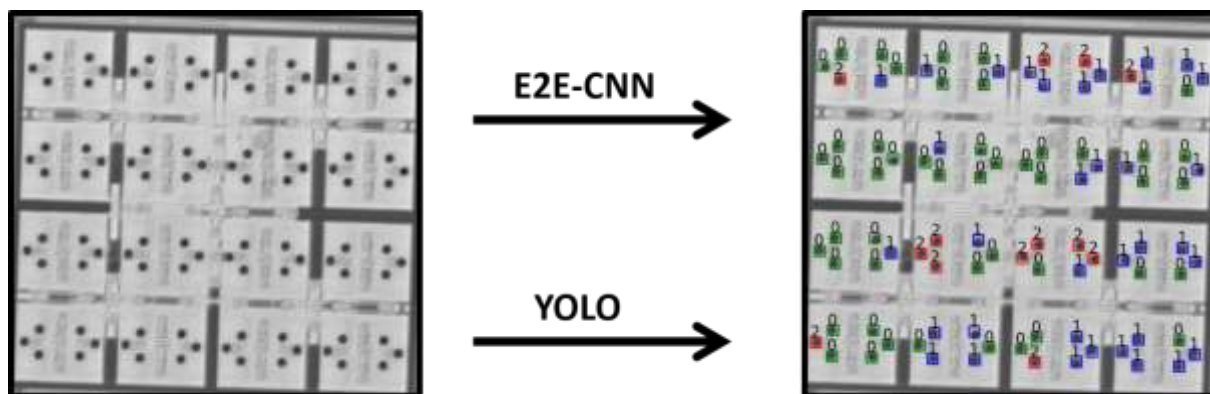
Supplementary Figure 4: Training and model architectures for the GAN generator (DCSCN) and discriminator. (a): Training procedure of SR-GAN. The low-resolution image is passed through the generator. The generated image is then judged by the discriminator to be real or fake and the feedback-loss \mathcal{L}_A is given to the generator. Simultaneously, the generator is constrained by \mathcal{L}_p to be as close to the HR image as possible. \mathcal{L}_D provides information for the discriminator to get better at catching unrealistic images. (b): Model architecture for the generator. The first block consists out of convolutional layers with 176, 160, 144, 128, 112, 96, 80, 64, 48 and 32 filters. The second block (reconstruction block) is split in two. It has convolutional layers with 32 and 32, 64 filters. The kernel size is 3 except for the first layers in the reconstruction block, where we use a kernel size of 1 for feature extraction. (c): Model architecture of the discriminator. We use convolutional layers with strides $S = 2, 1, 2, 1, 2, 1, 2$. After every strided convolution, the number of filters is doubled, starting with 64 filters in the first convolution. A kernel size of 3 is used for all layers.

SUPPLEMENTARY NOTE 5: Comparison of YOLO and E2E-CNN

As mentioned in the main text, we claim a speed improvement of a factor of 60x for our YOLO model in comparison to the state of the art E2E-CNN approach.² In this section, we present the details on how this comparison was done.

The main purpose of E2E-CNN is to localize TSVs and classify them into different defect classes, as can be seen in Supplementary Figure 5 and the original paper. To make a comparison between E2E-CNN and YOLO, we use the original data provided in the E2E-CNN publication. This data is provided by the authors of that paper. We labeled around 4000 images in the proprietary YOLO format. This is less than the 7000 to 20.000 images used for training both CNNs in the original paper. However, every one of our images contains on average 2-3 TSVs, since image tiles of 640x640 are used for training YOLO.

The original E2E-CNN paper states a value of 52s for classifying 96 TSVs, not including localization. To test the speed of YOLO, we utilize the same hardware (NVIDIA Quadro P4000 GPU) and obtain an evaluation time of roughly 1s for localization AND classification of 96 TSVs. This is equivalent to an increase in speed of around 60x when assuming that the localization in the E2E-CNN model takes around 6s. Our model has an accuracy of 96% for classifying TSVs in the right class, which is similar to what the E2E-CNN is capable of.



Supplementary Figure 5: TSV samples for our comparison. The E2E-CNN and YOLO object detection algorithms are applied to the same dataset to measure the performance of the models under equivalent conditions. Different colors in the right image correspond to different TSV classes as defined in the original E2E-CNN paper.

References

1. Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vis.* **60**, 91–110 (2004).
2. Paulachan, P., Siegert, J., Wiesler, I. & Brunner, R. An end-to-end convolutional neural network for automated failure localisation and characterisation of 3D interconnects. *Sci. Rep.* **13**, 9376 (2023).