ILC-HiGrade-Report-2015-001

Studies on automated defect detection at OBACHT

Y. Ulrich*, M. Wenskat*

January 2015

Abstract

A high-resolution camera and sophisticated illumination system has been mounted in a robot that facilitates the acquisition of images of the inner surface of superconducting cavities, which otherwise are hardly accessible. The OBACHT (Optical Bench for Automated Cavity inspection with High resolution images on short Timescales) system in use at DESY has been optimized for acquisition time and resolution. This note describes the algorithm used to detect surface defects in the images.

1. Introduction

The "Optical Bench for Automated Cavity inspection with High resolution images on short Timescales" (OBACHT) [1] is a camera system developed for automatic inspection of 9-cell 1.3 GHz SRF-cavity [2] surfaces. A photograph of such a cavity is shown in figure 1. Its principle of operation is shown in figure 2. More details are described in [3]:

*DESY, Hamburg, Germany

A tube with a camera mounted inside longitudinally slides into the cavity to inspect the surface quality and to detect surface defects.
An example OBACHT picture is shown in figure 3. The ridge in the center is the welding seam and the lateral lines correspond to the edge of the melting zones. Due to surface treatments they may vary from picture to picture. The image is about $w_m = 12\,\text{mm}$ wide and about $h_m = 9\,\text{mm}$ tall such $1\,\text{px}$ equals about $3.5\,\mu\text{m}$ at the equator.
There are numerous types of surface features that are visible in the OBACHT pictures. Some of them are considered to be surface defects. These defects may limit the achievable performance of a cavity as discussed in [3]. An overview of encountered defects is presented in appendix A.
A complete OBACHT inspections takes images at the equators, irises and cell regions (figure 2). By rotating the camera in steps of $\Delta\phi = 4.6°$ (for equators and cells) or 12° (for irises) the entire inner surface is captured. Thus 75 images per equator, 150 images per cell and 30 images per iris need to be analyzed. Due to the infeasibility of analyzing all 2325 images per cavity by hand, an automated approach using computer vision is proposed. The automation is vital for the proposed International Linear Collider [2] which requires in the order of 16,000 cavities to be examined.
This report is organized as follows: For each proposed algorithm a short theoretical description is provided. The implementation is discussed for both algorithm in section 3. In section 4 preliminary results are presented.

Figure 1: A TESLA cavity at DESY [4]

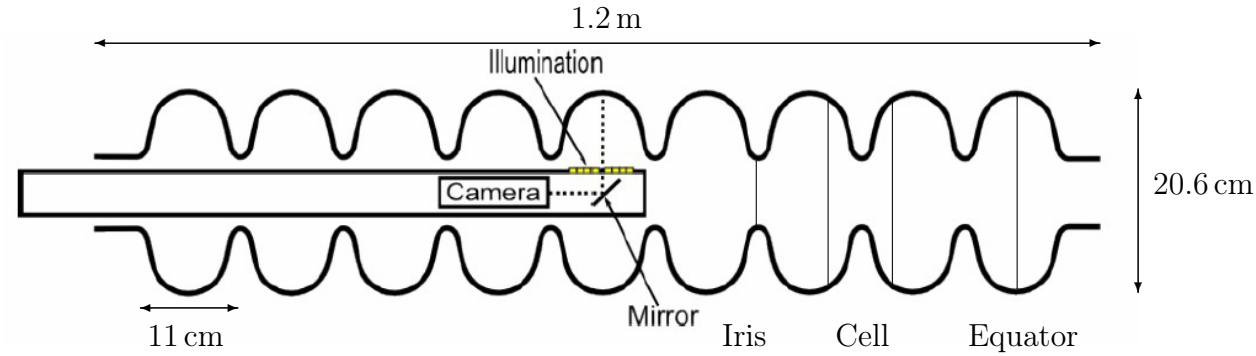


Figure 2: The principle of the OBACHT system [3]: The high-resolution camera is mounted in a tube which can be introduced into the 9-cell cavity.

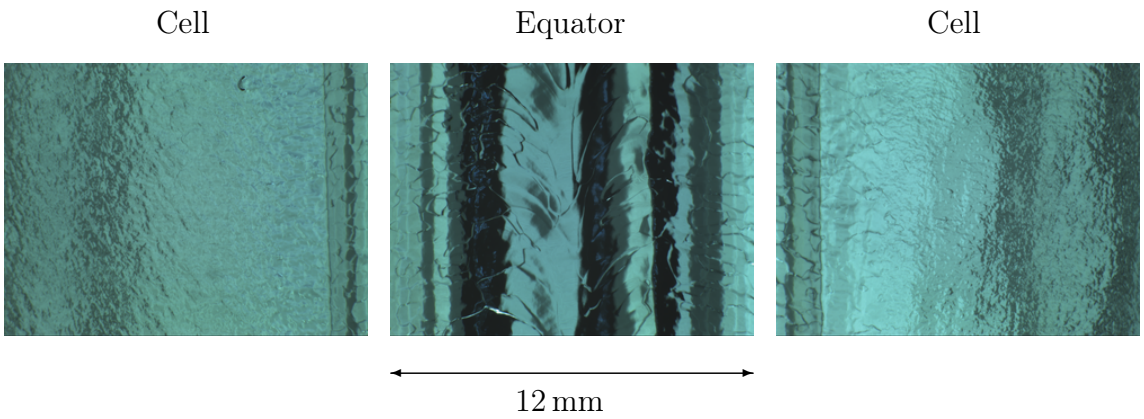


Figure 3: Equator and cell images of a cavity. In the left cell image a defect is visible

2. Proposed algorithm

The proposed algorithm consist of two parts. In a first analysis the program searches for abnormal features in the images using texture based analysis. However this approach can only find large or prominent defects. Thus, in a second step smaller features are detected using an object recognition algorithm. The need for the use of multiple algorithms is given by the fact, that a trained object recognition algorithm will only detect known and expected defects. Therefore, the first step is necessary to detect also previously unseen or rare defects that cannot incorporated in the training set.

2.1. Texture based analysis

OBACHT images show a large variety of surface textures not all of which are due to actual defects (see A Overview of defects). This has to be considered when using a texture based approach. Therefore, for all encountered surface textures, samples were gathered and the SFTA (Segmentation-based Fractal Texture Analysis) [5] vector $\mathcal{F}_s$ is computed and a naive Bayes classifier is trained to distinguish surface types.
The SFTA algorithm decomposes an image into a set of binary images for which the fractal dimension is calculated. The values of $\mathcal{F}$ consists, among other surface characteristics, of these fractal dimensions.
The flowchart of the actual detection operation is shown in figure 4. A grid with cell size $\Delta x \times \Delta y = (250\,\mathrm{px} \times 250\,\mathrm{px})$ is applied to the image. For each grid cell (x, y) the SFTA vector $\mathcal{F}_{x,y}$ is computed. Using the naive Bayes classifier the surface type s can be guessed and the predictions filtered (see below). Finally the distances

$$d_{x,y} = |\mathcal{F}_{x,y} - \bar{\mathcal{F}}_s|_F \tag{1}$$

$$d'_{x,y} = |\mathcal{F}_{x,y} - \mathcal{F}_{x\pm1,y\pm1}|_F \tag{2}$$

$$\mathcal{F}_{x\pm1,y\pm1} = \frac{1}{8} \sum_{i,j\in\{-1,1\}} \mathcal{F}_{x+i,y+j}$$

are computed where $\bar{\mathcal{F}}_s$ and $|\cdot|_F$ denotes the averaged feature vector for the surface type s and distance function of the feature vector space respective.With these set of equations, each grid cell is compared to known texture classes, represented by $\mathcal{F}_s$, and its neighbors $F_{x\pm1,y\pm1}$ and the extent of the deviations is quantified via the metric $|\cdot|_F$. The distance $|\cdot|_F$ is the standardized euclidian distance

$$|A - B|_F^2 = \sum_{i=1}^{n} \left(\frac{a_i - b_i}{\sigma_i} \right)^2 \tag{3}$$

where a_i and b_i is the ith element of A and b_i respective. σ_i is the standard deviation of a representative set of a_i.
Since there are some expected changes in the features (eg. at the onset of the heat affected zones) the values must be suppressed there to avoid distortion of the comparison. This results in an lower accuracy in this area.

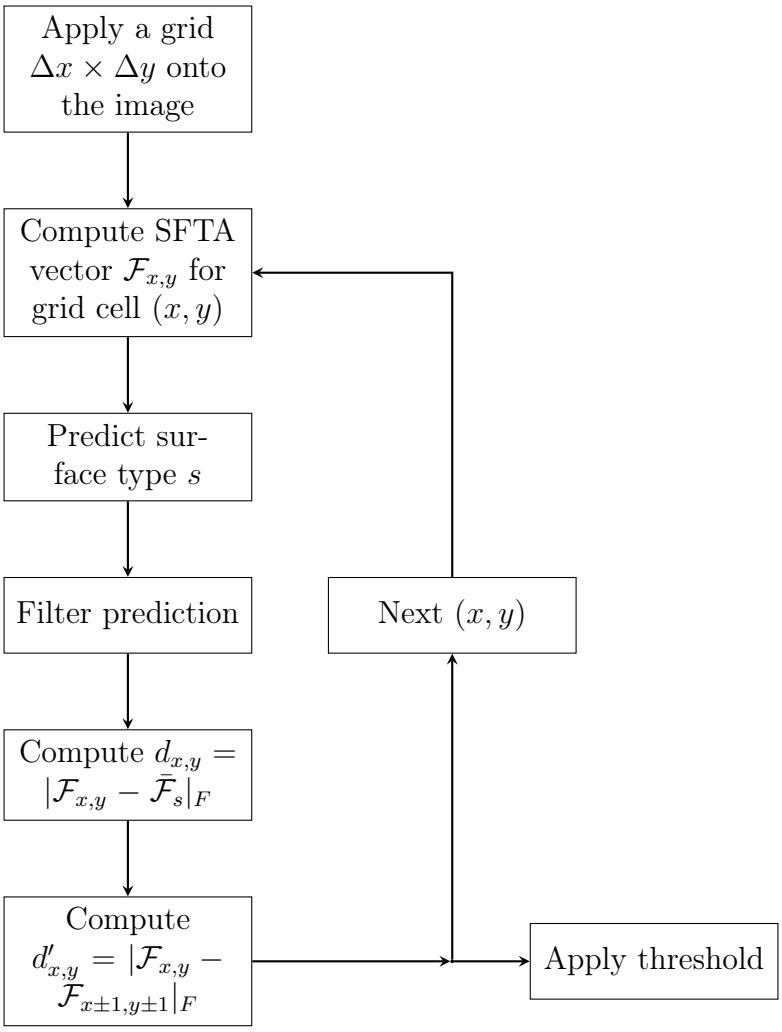


Figure 4: For each grid cell the SFTA vector $\mathcal{F}_{x,y}$ is compared with its neighbors and a training set obtained by predicting the surface type.

2.2. Object based analysis

The Viola-Jones algorithm [6] uses a scale-invariant strong classifier that was trained to find certain features, ie. a defect, in an image. To reduce the false positive ratio, a cascade of multiple classifiers is used to probe the image and to compute a feature value. Using the computer vision library OpenCV [7] a classifier was trained to detect pits and cateyes by [8].

3. Implementation

The texture based detection was implemented in MATLAB [9] using its Image Processing Toolbox and the Parallel Computing Toolbox. The object based detection was implemented in C++ using OpenCVand is described in detail in [8] and a brief description will be given here.
The source code contains the following files:

- `texture/`: main folder for the texture based analysis
  - `configure.m`: initialize standard configuration like the grid size
  - `sftaTrain.m`: computes $\mathcal{F}_s$ and trains the naive Bayes classifier
  - `process.m`: computes $\mathcal{F}_{x,y}$ for each grid cell and image
  - `neighborbased.m`: post processing analysis by computing $d'_{x,y}$
  - `trainingbased.m`: post processing analysis by computing $d_{x,y}$
  - `tools/`: necessary subroutines
- `object/`: main folder for the object based analysis
  - `objectDetection.cpp`: main C++ file
  - `cascade.xml`: a Haar classifier to detect cat eyes.
  - `Makefile`: a GNU make file for compilation

3.1. Texture based analysis

The most important functions of the texture based analysis and their usage is described in the following.

3.1.1. `sftaTrain.m`

The `sftaTrain`-function takes a series of OBACHT images specified by the user and opens a graphical user interface (GUI, figure 5). In the shown image, the user is asked to select the correct surface type for each grid cell by clicking. To continue, the user closes the window.

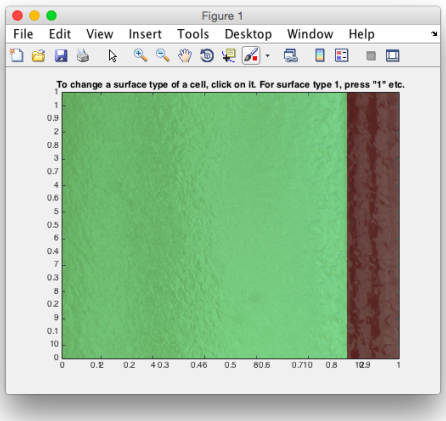


Figure 5: The GUI of the `sftaTrain` function

The function will return the averaged feature vector $\bar{\mathcal{F}}_s$ (`sfta_means`) and a trained naive Bayes classifier (`nb`). It will also save these and other variables to a MATLAB file `training.mat`.
Assuming that the training set was sufficiently large this needs to be done only once.

3.1.2. `neighbor.m`

This MATLAB subroutine will compare each grid cell with its eight neighbors evaluating (2). To do this, it will loop over each grid cell and compute the `neighbor` variable, a 3×3 matrix of the cell's eight neighbors. By averaging over all non-zeros elements in $\texttt{neighbor} - \texttt{neighbor}_{2,2}$ the $d'_{x,y}$ variable is computed.

3.1.3. `trained.m`

This MATLAB subroutine will compare the cells with data obtained by `sftaTrain`. At first, it will predict the surface type using the naive Bayes classifier stored in `nb` and a map $s(x, y)$ is generated. Since prominent defects have a different texture that might disturb the prediction, the generated map s(x,y) will be filtered by the dilute-function as follows

$$s(x,y)' = \begin{cases} s(x,y) & \text{there is a neighbor } s(x \pm 1, y \pm 1) = s(x,y) \\ \mathrm{mfe}\big(s(x \pm 1, y \pm 1)\big) & \text{otherwise} \end{cases} \tag{4}$$

where the mfe-function selects the most frequent element in the respective neighborhod. An example is shown in figure 6 where the single red element is filtered out. Then $s(x, y)'$ can be used to select the training vector for comparison to compute $d_{x,y}$.

3.2. Object based analysis

The implementation of [8] was only slightly modified. The C++ routine loads the image and the cascade classifier which is applied to the image. Optionally, the defects can be highlighted by a circle.

4. Results

For all 1350 cell images of one cavity the feature vector $\mathcal{F}_{x,y}$ was computed which took 34.4 s ± 1 s each on an AMD Opteron(TM) Processor 6276 core. Using 12 cores in MATLAB, analyzing all images took about one hour.
The images in figure 7 has an arc-like structure and the one in figure 8 shows scratches. The values for $d_{x,y}$ and $d'_{x,y}$ are shown for both images. One can clearly see, that the algorithm can find these defects as indicated by the high values of $d_{x,y}$ or $d'_{x,y}$.
Even though the results of $d'_{x,y}$ look much better in comparison to $d_{x,y}$, $d_{x,y}$ should be incorporated in the final decision since it will not perform good on the onset of the heat affected zone (see section 2.1).
To finally mark an image, a thresholding operation is necessary. A threshold low enough to detect all features might falsely tag images. An example is shown in figure 9.
An example of a detected cateye is shown in figure 11. [8] measured the processing time to be below 1 s. However, training took about 3 days on a normal PC and needs to be performed for each defect the algorithm should be detected.
A sample set of ten images has been tested. Five of these images are considered as a defect. Using the definitions

$$\text{Accuracy} = \frac{\#\text{True positive} + \#\text{True negative}}{\#\text{images}} \tag{5}$$

$$\text{Positive predictive value} = \frac{\#\text{True positive}}{\#\text{Test outcome positive}} \tag{6}$$

$$\text{True positive rate} = \frac{\#\text{True positive}}{\#\text{defects}} \tag{7}$$

the quality of the algorithm can be approximated. These can be plotted as a function of the threshold as shown in figure 10.

5. Summary

Using a texture-based analysis, an automated defect detection algorithm for OBACHT has been proposed which yields good results as discussed in section 4. However, the false positive rate should be reduced further. In addition, further work is required to also apply the algorithm to equator and iris images to fully automate the analysis procedure.

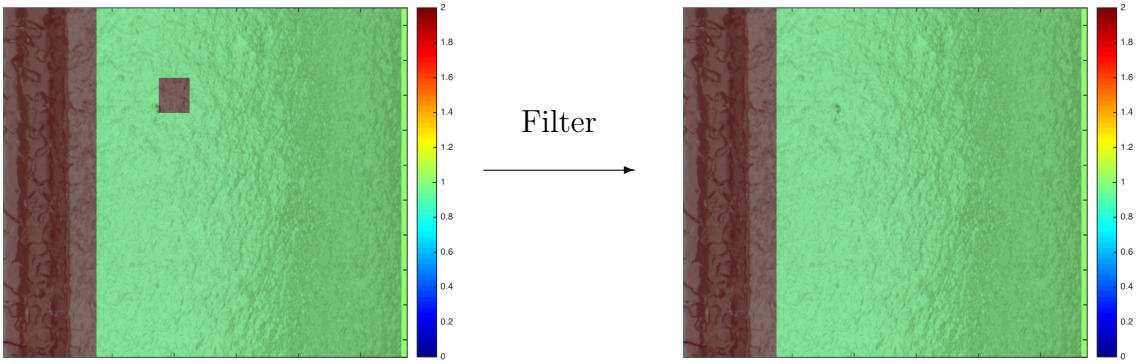


Figure 6: An image overlayed with $s(x, y)$ and $s'(x, y)$ before and after application of the `dilute`-function respective

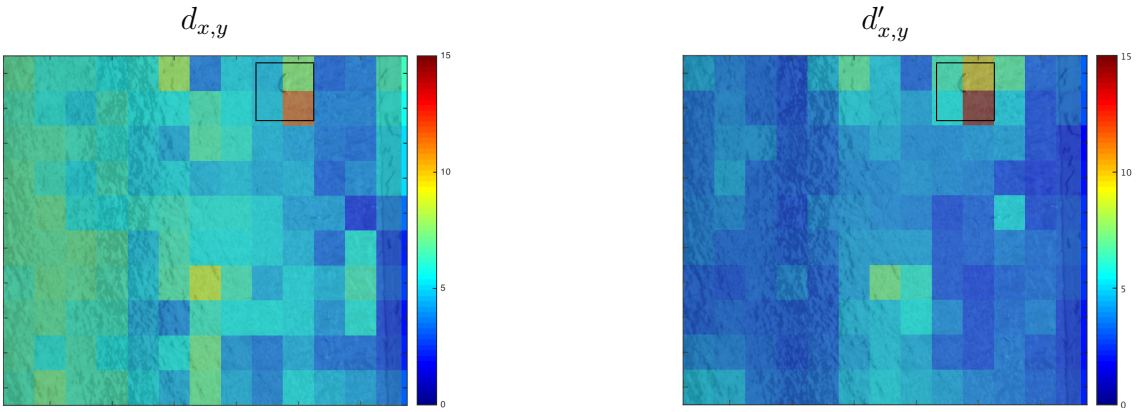


Figure 7: An image with a defect marked with a rectangle. The overlay corresponds to $d_{x,y}$ (left) and $d'_{x,y}$ (right) in σ

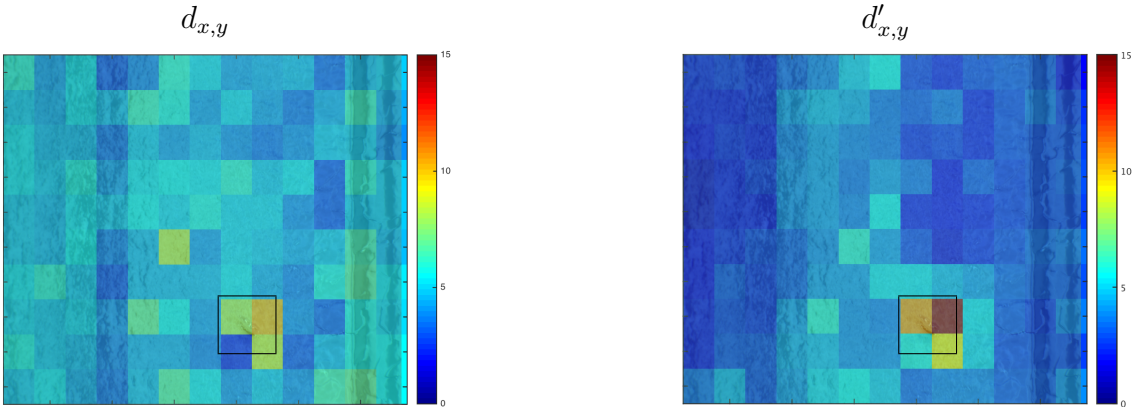


Figure 8: An image with a defect marked with a rectangle. The overlay corresponds to $d_{x,y}$ (left) and $d'_{x,y}$ (right) in σ

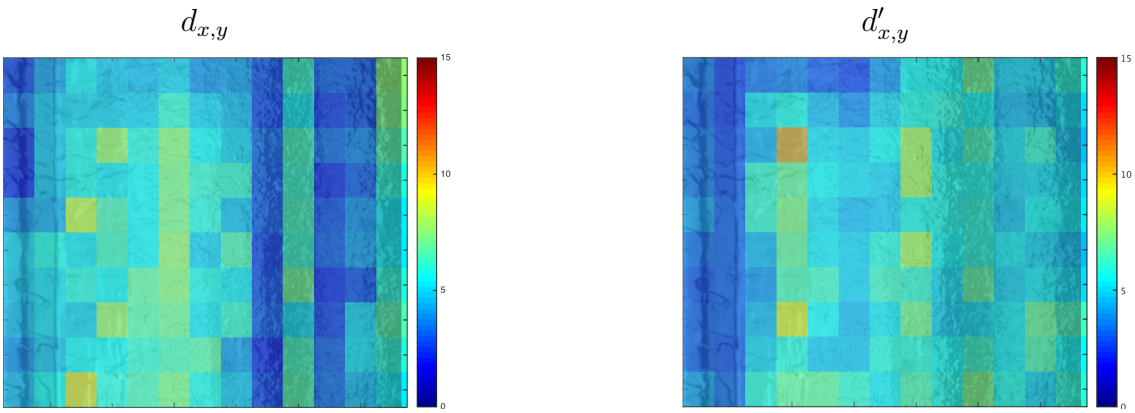


Figure 9: An image without a defect. The overlay corresponds to $d_{x,y}$ (left) and $d'_{x,y}$ (right) in σ

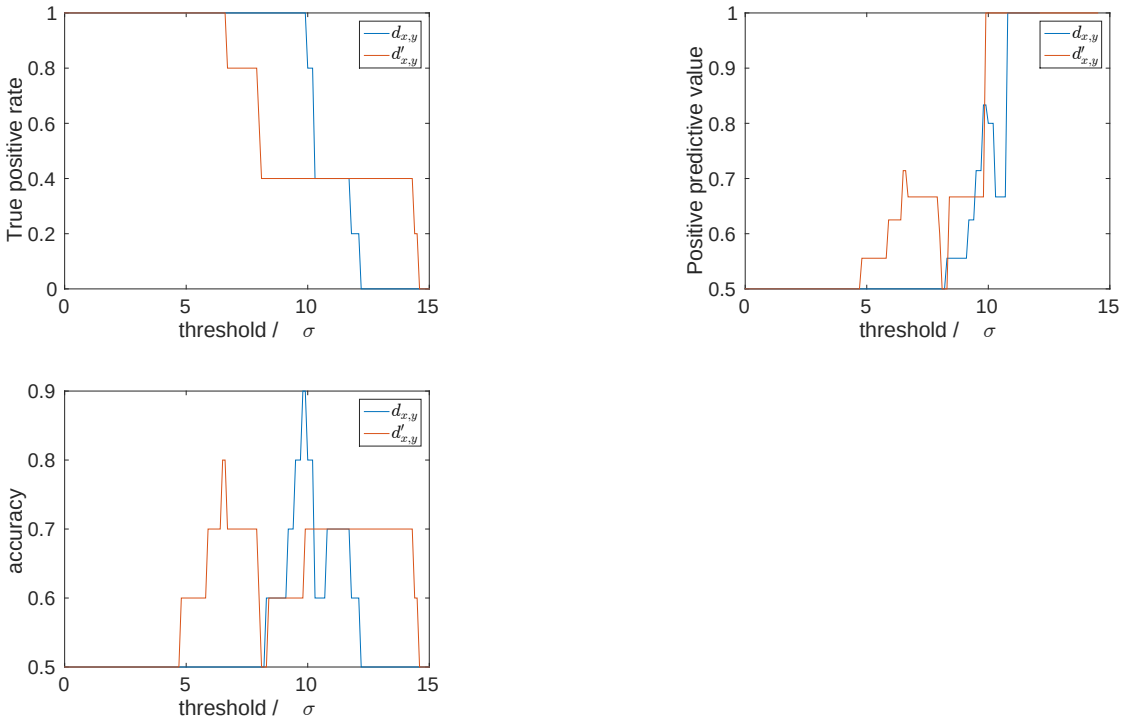


Figure 10: Accuracy, positive predictive value and true positive rate as a function of the selected threshold

References

[1] M. Lemke, et al. OBACHT - Optical Bench for Automated Cavity Indespection on Short Time Scales. ILC-HiGrade Report 001, DESY, Hamburg, Germany, 2013.

[2] Chris Adolphsen, et al. The International Linear Collider Technical Design Report - Volume 3.I: Accelerator R&D in the Technical Design Phase. 2013.

[3] S. Aderhold. *Study of field-limiting defects in superconducting RF cavities for electron-accelerators*. PhD thesis, DESY, Hamburg, Germany, 2014.

[4] Lea Steder. Optische Inspektion von supraleitenden Cavities. Presented at the "DPG Frühjahrstagung 2013", 03 2013.

[5] Alceu Ferraz Costa, Gabriel Humpire-Mamani, and Agma Juci Machado Traina. An efficient algorithm for fractal analysis of textures. In *Graphics, Patterns and Images (SIBGRAPI), 2012 25th SIBGRAPI Conference on*, pages 39–46. IEEE, 2012.

[6] Paul Viola and Micheal J Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

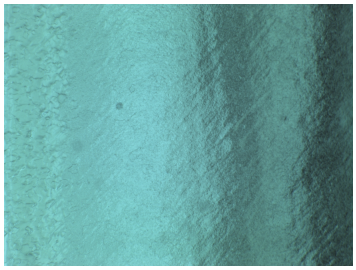
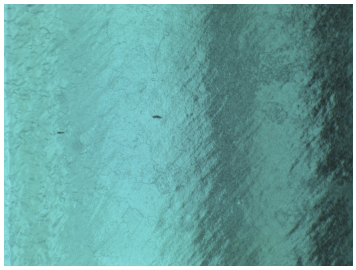
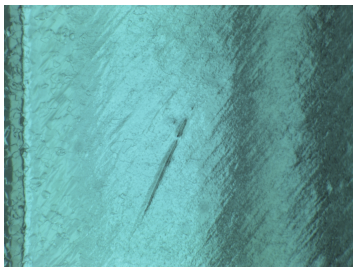
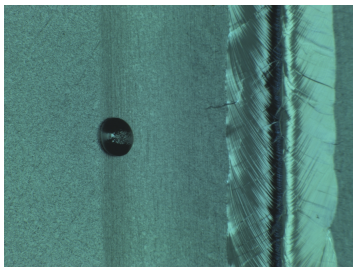
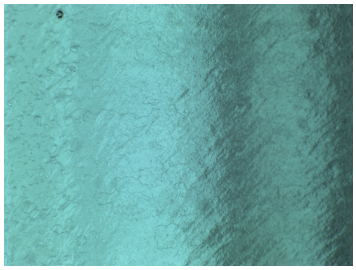
[7] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[8] Kara-Ali Aliev. Study of automated defect detection methods for OBACHT. 2013.

[9] *MATLAB version 8.4 (R2014b)*. The MathWorks Inc., Natick, Massachusetts, 2014.

A. Overview of defects

The following table shows an overview of encountered surface features:

Image	Description of feature
	A footprint
	Two scratches
	A big scratch
	A big droplet
	A cateye

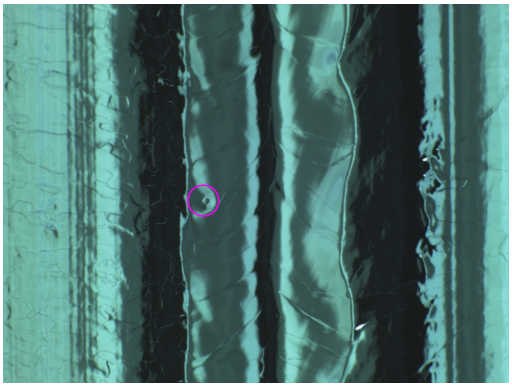


Figure 11: A welding seam with a detected cateye using [8]