Federal Office
for Information Security

Open Source Face Image Quality (OFIQ)

Implementation and Evaluation of Algorithms

Version 1.2

2024-11-01

Authors

Johannes Merkle[1], Christian Rathgeb[1], Benjamin Herdeanu[1], Benjamin Tams[1], Day-Parn Lou[1], André Dörsch[1], Maxim Schaubert[1], Jonas Dehen[1], Liming Chen[2], Xiangnan Yin[2], Di Huang[3], Anna Stratmann[4], Marcel Ginzler[4], Marcel Grimmer[5,6], Christoph Busch[5,6]

1) secunet Security Networks AG,
Kurfürstenstr. 58, 45138 Essen, Germany

2) Liris Laboratory UMR CNRS 5205, Ecole Centrale de Lyon,
36 Avenue Guy de Collongue, 69134 Ecully Cedex, France

3) School of Computer Science and Engineering, Beihang University,
Xueyuan Road, Haidian District, 100191Beijing, China

4) Federal Office for Information Security,
P.O. Box 20 03 63, 53133 Bonn, Germany

5) Hochschule Darmstadt, Fachbereich Informatik,
Schöfferstrasse 3, 64295 Darmstadt, Germany

6) Norwegian University of Science and Technology,
Teknologiveien 22, 2815 Gjøvik, Norway

Federal Office for Information Security
P.O. Box 20 03 63
53133 Bonn
E-Mail: OFIQ@bsi.bund.de
Internet: https://www.bsi.bund.de

Table of Contents

List of Figures

List of Tables

1 Executive Summary

This report summarises the development of Open Source Facial Image Quality (OFIQ), a library of open-source algorithms for face image quality assessment. OFIQ is the reference implementation for the international standard ISO/IEC 29794-5 and comprises various quality assessment algorithms:

- An algorithm providing a unified quality score which aims to predict the utility of the facial image for recognition purposes.
- Various quality components algorithms, each of which assessing the conformance of the facial image to a certain requirement, e.g. from ISO/IEC 39794-5:2019 [1]. Some of these quality components depend on the environment used for imaging acquisition (capture-related), while others primarily depend on aspects of the presentation of the face (subject-related).

Consequently, OFIQ outputs a vector of quality measures, each of which being an integer in the range [0,100], where higher values signify higher quality, i.e., higher utility for face recognition or higher conformance to the requirements.

The report documents the algorithms for each quality measure, the results of their evaluation, and the selection of the final algorithms for OFIQ.

2 Introduction

2.1 Motivation

Biometrics, including face recognition, have become increasingly important in various application scenarios including transnational deployments such as the Entry-Exit System (EES). The EES will be used in the European Union (EU) to automatically monitor the border-crossing of third-country nationals. In such large-scale biometric systems, it is of utmost importance to ensure high sample quality of biometric data. Poor sample quality results in unreliable biometric decisions and can cause recognition errors.

With recent technological advances, in particular the introduction of algorithms based on deep learning, face recognition error rates dropped massively. However, they remain significant, depending on several factors, such as the imaging process or the level of cooperation of the biometric capture subject. Accordingly, face image quality assessment algorithms are designed to calculate a (unified) quality score for a captured facial image, which indicates its utility for recognition purposes, see Figure 1.

QS = 95	QS = 85	QS = 62	QS = 42
A	b	c	d

Figure 1: Examples of face images of a single subject with example quality scores (QS).

The estimated quality scores can be used to ensure that only facial images of sufficient quality are fed into a face recognition system. Specifically, the quality scores are compared with a quality threshold value, and if a quality score exceeds the quality threshold, the corresponding facial image is accepted, otherwise it is rejected. In the latter case, a re-capture might be initiated, where actionable feedback is needed for users and operators. This is achieved by computing, in addition to the unified quality score, various quality components assessing the conformance of the facial image to specific requirements, some of which mainly depend on the systems and environment of image acquisition (capture-related) and while others primarily depend of aspects of the presentation of the face (subject-related).[1] Each quality component corresponds to a specific defect that is known to negatively affect the utility of facial images. For example, overexposure or non-frontal pose represent examples of capture- and subject-related defects, respectively. By assessing specific quality components, decisions of quality assessment algorithms become more transparent for users and operator, which is essential in biometric systems like the EES.

[1] A clear distinction between these two classes is not always possible. For instance, the sharpness of the facial image depends not only on the acquisition system but also on the subject (keeping still).

2.2 Objective and Approach

The objective of the project "Open Source Facial Image Quality" (OFIQ) is to develop, implement, and publish a library of open-source algorithms for the quality assessment of facial images used in face recognition. This library, named OFIQ, aims to be applicable for various biometric applications, particularly in border control scenarios.[2] OFIQ is written in the C/C++ programming language. OFIQ is the reference implementation for the international standard ISO/IEC 29794-5. Its source code is available from https://github.com/BSI-OFIQ/OFIQ-Project.

This report documents the implementation and evaluation of the candidate algorithms as well as the selection of the final algorithms for OFIQ and the current revision of ISO/IEC 29794-5.

For each quality component included or considered for ISO/IEC 29794-5, candidate algorithms were implemented and evaluated. Many of the candidate algorithms were taken from or contributed to the evolving drafts of ISO/IEC 29794-5:

- Working Draft 1 (ISO/IEC WD1 29794-5:2020) [2]
- Working Draft 4 (ISO/IEC WD4 29794-5:2022) [3]
- Working Draft 5 (ISO/IEC WD5 29794-5:2022) [4]
- Working Draft 6 (ISO/IEC WD6 29794-5:2023) [5]
- Committee Draft 1 (ISO/IEC CD1 29794-5:2023) [6]
- Committee Draft 2 (ISO/IEC CD2 29794-5:2023) [7]
- Committee Draft 3 (ISO/IEC CD3 29794-5:2023) [8]
- Draft International Standard (ISO/IEC DIS 29794-5:2024) [9]
- Final Draft International Standard (ISO/IEC FDIS 29794-5:2024) [10]

Additionally, the implemented quality measures and algorithms took into consideration the recommendations and requirements from other relevant standards, notably ISO/IEC 19794-5:2011 [11], ISO/IEC TR 29794-5:2010 [12], ISO/IEC 39794-5:2019 [1] and the BSI Technical Guideline TR-03121 Part 3 Volume 1 [13].

In order to become a universally usable, OFIQ needs to meet various requirements. Based on a thorough research of the current state-of-the-art [14], suitable candidate algorithms have been shortlisted, implemented, tested and improved, and, for each task, the best algorithm was selected. The results of this evaluation and the final algorithm selection are documented in Sections 4, 5, 6, and 7. Additionally, it was ensured that the licences of used algorithms allow commercial use, which is a key requirement for OFIQ.

Furthermore, OFIQ shall provide a good trade-off between state-of-the-art accuracy, runtime and resource consumption. To facilitate its use, OFIQ must be compatible with relevant platforms, including Windows, Linux, MacOS, Android and iOS.

[2] Face recognition in border control scenarios does not only comprise the comparison of the live image with the reference image but also supporting algorithms for attack detection, in particular morphing attack detection (MAD) and presentation attack detection (PAD).

2.3 Evaluation Methodology

The evaluation of the algorithms was conducted through several means:

- Eligible test sets with ground truth labels were collected and created, and the algorithms were then assessed with respect to their accuracy in predicting the ground truth. Depending on the nature of the ground truth labels (numeric or binary), different visualisation methods are employed: For numeric labels, e.g. for head pose angles, empirical cumulative distribution functions (ECDF) are presented, while for binary (2-class) labels, detection error trade-off (DET) curves are utilised. In cases where DET curves cannot be plotted, e.g. in case of perfect separation of the classes, score distributions are displayed. The evaluations with ground truth data are based on the "native" scores output by the specified algorithms and not on the quality component values resulting from the mapping to the integer interval [0,100] defined in ISO/IEC FDIS 29794-5:2024 [10].
- In order to determine the influence of the quality assessment algorithm on the accuracy of face recognition algorithms, error-versus-discard characteristic (EDC) curves are computed. The EDC evaluation aims to demonstrate the utility of the quality assessment algorithm in face recognition by evaluating the reduction of the False Non-Match Rate (FNMR) of face recognition algorithms when using the quality assessment algorithm to successively filter out the face images with lowest quality. This evaluation employs several open-source and commercial face recognition algorithms, along with a large subset of VGGFace2 [15] as test set, as described in Section 5.1. Since the EDC evaluation requires a higher-is-better semantic, outputs that do not have such a semantic were transformed appropriately.
- Whenever possible, the most promising algorithms were also submitted to the Specific Image Defect Detection (SIDD) track of the Face Analysis Technology Evaluation (FATE) Quality[3]. NIST evaluates these algorithms using their own sequestered test data. However, the quality components covered by FATE Quality and its SIDD track don't fully align with the quality components outlined in ISO/IEC 29794-5. Consequently, only algorithms pertaining to quality components covered by the FATE Quality SIDD track were submitted. The results from the evaluation in FATE Quality SIDD track [16] are also summarised in this report.

[3] Previously named FRVT Quality, see https://pages.nist.gov/frvt/html/frvt_quality.html

3 OFIQ Framework

An overview of the OFIQ framework[4] is depicted in Figure 2. OFIQ takes as input a single 2D face image. First different pre-processing steps (common computations) are performed. Subsequently, the unified quality and different capture- as well as subject-related quality components are assessed, resulting in an output vector consisting of the unified quality score and various quality component values.

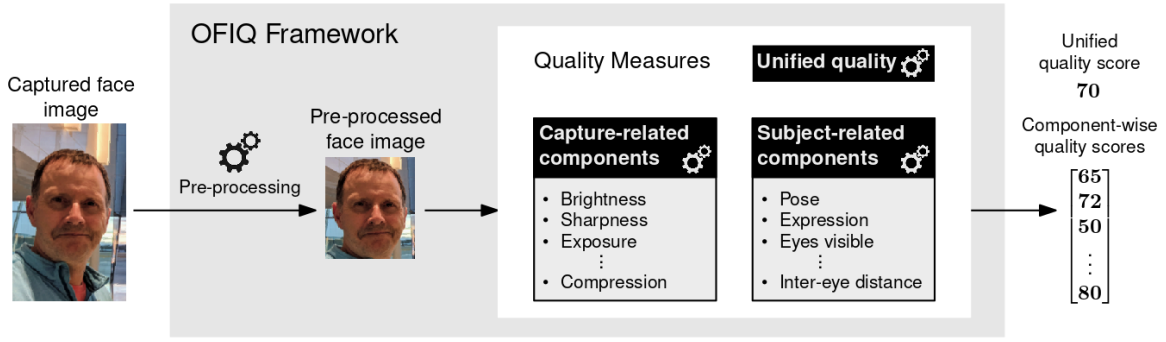

Figure 2: Overview of the OFIQ framework.

The assessed quality components are listed in Table 1, each of which is computed by a distinct algorithm.

Capture-related Quality Components	Subject-related Quality Components
• Background uniformity • Illumination uniformity • Moments of the luminance distribution (Brightness, Variance) • Over-exposure prevention • Under-exposure prevention • Dynamic range • Sharpness • No compression artifacts • Natural colour	• Single Face Present • Eyes open • Mouth closed • Eyes visible • Mouth occlusion prevention • Face occlusion prevention • Inter-eye distance • Head size • Crop of the face (leftward, rightward, upward, downward) • Head pose (yaw, pitch, roll) • Expression neutrality • No head coverings

Table 1: List of capture- and subject-related quality components in OFIQ.

The algorithm for the unified quality score and the algorithms for the individual quality components all output a native quality measure, which is a floating-point value without uniform value range or semantic. Each of these native quality measures is then mapped to the unified quality score or the quality component value, respectively, in the target range [0,100] having a higher-is-better semantics. This means that a unified quality score or a quality component of 100 refers to optimal quality.

The development of the algorithms for the capture-related quality components is documented in Section 5. The development of the algorithms for the capture-related quality components and for the subject-related quality components is documented in Section 6 and Section 7, respectively.

The following quality components had been considered during development of the OFIQ and the revision of ISO/IEC 29794-5 but discarded due to a lack of algorithms with satisfactory predictive performance.

[4]https://github.com/BSI-OFIQ/OFIQ-Project

- Radial Distortion Prevention. The implementation and evaluation of candidate algorithms for this quality component is documented in Section 6.9.
- Frontal Gaze. No candidate algorithms have been implemented and evaluated for OFIQ, but a high-level algorithm is specified in Annex D of ISO/IEC FDIS 29794-5:2024 [10].
- Shoulder Presentation. No candidate algorithms have been implemented and evaluated for OFIQ but a high-level algorithm is specified in Annex D of ISO/IEC FDIS 29794-5:2024 [10].
- Camera to Subject Distance. No candidate algorithms have been implemented and evaluated for OFIQ, but a high-level algorithm is specified in Annex D of ISO/IEC FDIS 29794-5:2024 [10].
- Motion Blur Prevention. The implementation and evaluation of candidate algorithms for this quality component is documented in Section 7.13.

4 Pre-Processing Algorithms

Many of the algorithms for the unified quality score and the different quality components rely on certain pre-processing steps. The algorithms used to perform these steps are described in this section.

4.1 Face Detection

Face detection algorithms aim to provide a bounding rectangle tightly enclosing the face in the image. Depending on the algorithm and the data used for training, this rectangle may include or exclude certain portions of the face, such as the forehead. However, since the primary objective of face detection is to confine subsequent computations (e.g., facial landmark estimation or head pose estimation) to the region where the face is situated, and theses computations exhibit a certain degree of tolerance regarding this region, it is not imperative to precisely define the location and size of the face bounding boxes to be output.

The bounding box is always upright, e.g. the edges of the box are parallel to the edges of the image, independent of the roll angle. An example of a face bounding box output by a face detection algorithm (in this case, RetinaFace) is shown in the following figure.

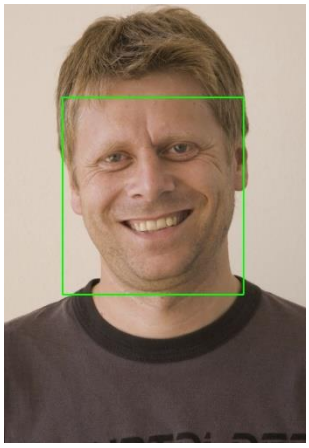

Figure 3: Example bounding box output by a face detector.

Note that most face detection algorithms, including those considered for OFIQ, have been trained on face images where the roll angle is typically below 90 degrees and, thus, may fail for faces with larger roll angles.

4.1.1 Selection and Prototyping of Candidate Algorithms

The following face detection algorithms were implemented during the prototyping phase:

- Dlib: The dlib library includes a face detector based on Histogram of Oriented Gradients (HOG) features combined with a linear SVM classifier.[5] While this algorithm cannot compete with state-of-the-art face detectors, its simplicity and widespread use make it a suitable baseline.
- MediaPipe: The MediaPipe library implements a BlazeFace face detector [17] in Tensorflow Lite.[6] Within the python package *mediapipe*, this face detector is integrated into a facial mesh (3D landmarks) computation but can also be invoked explicitly. MediaPipe provides two face detection models:
    - A short-range model of 224 KB size optimised for faces captured within 2 meters distance of the camera.
    - A full-range model of 665 KB size optimised for faces within 5 meters distance.
- SSD: This face detection algorithm, available on GitHub, utilises a Single-Shot-Detector with a ResNet10 model of 10 MB size.[7]

[5] http://dlib.net/face_detector.py.html
[6] https://google.github.io/MediaPipe/solutions/face_detection.html
[7] https://github.com/sr6033/face-detection-with-OpenCV-and-DNN

- RetinaFace: This face detection algorithm is based on the RetinaFace approach [18] with a MobileNet model of 1.8 MB size and has been published on GitHub.[8]

All face detector algorithms output a list of face bounding boxes, where each bounding box is specified by integer coordinates (*a,b*) of its upper left corner in the image and its dimensions (*w, h*), where *w* represents the width and *h* the height of the bounding box. Coordinate *a* denotes the horizontal position (in pixels) from the left image border and coordinate *b* the vertical position (in pixels) from the upper image border.

4.1.1.1 Dlib Face Detector

The algorithm takes as input an image and the following optional parameter:

- *do_upscaling*: This parameter specifies how many times the image is upscaled within dlib's face detector implementation to allow detection of smaller faces.

To initialise the algorithm, the face detector object *D* is created by invoking the function get_frontal_face_detector() of dlib. The algorithm takes as input an image *I* in BGR colour channel order with 8 bits per channel.

1. Convert *I* to RGB colour channel order.
2. Invoke the face detector object *D* with the image and the parameter *do_upscaling* as arguments to obtain a *dlib rectangles* object *A*, which is an array of *dlib rectangle* objects.
3. For each *dlib* rectangle object *R* in *A*, extract the coordinates (a_i,b_i) and dimension (w_i,h_i) of the corresponding face bounding box using the functions left(), top(), width() and height() of the class *rectangles*.
4. For all *i*, set the coordinates to (a_i, b_i, c_i, d_i), where $c_i = a_i + w_i$ and $d_i = b_i + h_i$.
5. Sort the indices *i* by the area $(c_i - a_i) \cdot (d_i - b_i)$ of the face bounding boxes.
6. Return the list of coordinates (a_i, b_i, c_i, d_i) for all face bounding boxes, i.e. for all rows *i*.

4.1.1.2 MediaPipe Face Detector

The algorithm takes as input an image and the following optional parameters:

- *use_short_range_model*: This parameter specifies which of the face detection models should be used.
- *min_detection_confidence*: The minimum value for the confidence (output by the CNN) of the detected faces, i.e. detected faces with a lower confidence are discarded. The default value is 0.5.

To initialise the algorithm, the face detector object *D* is created by invoking the function *FaceDetection*() of the class *solutions.face_detection* of MediaPipe with parameters *min_detection_confidence* and *model_selection*, where *model_selection* is set to 0 if *use_short_range_model* is true and 1 otherwise. The algorithm takes as input an image *I* in BGR colour channel order with 8 bits per channel.

1. Convert *I* to RGB colour channel order.
2. Determine the height *h* and the width *w* of *I*.
3. Invoke the function process of the face detector object *D* with the image *I* as input to obtain a detection result object *R*, which contains a list of detection objects representing the detected faces.
4. If *R* is None, return an empty list.
5. For all detected faces, retrieve the relative coordinates (x_i,y_i) and dimensions (u_i,v_i) of the corresponding face bounding box as elements *xmin*, *ymin*, *width*, *height* of the element *location_data.relative_bounding_box* of the corresponding detection object.

[8] https://github.com/WIKI2020/FacePose_pytorch

6. Compute the list of absolute coordinates (a_i, b_i, c_i, d_i) of all face bounding boxes as $a_i = w \cdot x_i, b_i = h \cdot y_i, c_i = w \cdot (x_i + u_i), d_i = h \cdot (y_i + v_i)$.
7. Sort the indices *i* by the area $(c_i - a_i) \cdot (d_i - b_i)$ of the face bounding boxes.
8. Return the list of coordinates (a_i, b_i, c_i, d_i) for all *i*.

4.1.1.3 SSD Face Detector

The algorithm takes as input an image *I* and the following optional parameters:

- *P:* The relative amount by which the image is padded to improve detection of faces only partially covered by the image region.
- *T*: The minimum value for the confidence (output by the CNN) of the detected faces. Detected faces with a lower confidence are discarded.
- *M*: The minimum width of the face bounding boxes relative to the width *w* of the input image. Detected faces with a bounding box width smaller than $M \cdot w$ are discarded.
- *E*: The maximum fraction of the width and height of the image, by which the bounding box is permitted to protrude (i.e., extend beyond) the image boundary.

In the evaluation, the parameters were initially set to

- $P = 0$
- $T = 0.6$
- $M = 1/8$
- $E = 0.2$

For the 2nd submission (secunet_002) to the NIST FATE Quality SIDD track, the parameters were changed to

- $P = 0$
- $T = 0.6$
- $M = 1/20$
- $E = 0.2$

For the 3rd and 4th submission (secunet_003 and secunet_004) to the NIST FATE Quality SIDD track, the parameters were changed to

- $P = 0.2$
- $T = 0.4$
- $M = 1/50$
- $E = 0$

To initialise the algorithm, the CNN model files of the face detector are loaded using the OpenCV function readNetFromCaffe(), which creates a neural network model object *M*. Note that the CNN model files are in Caffe format and comprise a prototext file and binary protocol buffer file. The algorithm takes as input an image *I* in BGR colour channel order with 8 bits per channel.

1. Determine the height *h* and the width *w* of *I*.
2. Pad *I* from the left and right side by $p_1 = \lfloor w \cdot P \rfloor$ and from the top and bottom side by $p_2 = \lfloor h \cdot P \rfloor$, and update $w \rightarrow w + p_1$ and $h \rightarrow h + p_2$ to the new width and height of *I*.
3. Resize the image *I* to size (height, width) 300×300 using OpenCV with bilinear interpolation.

4. Normalise I using mean $\mu = (104.0, 117.0, 123.0)$ and standard deviation $\sigma = (1,1,1)$ to obtain an array A, i.e. set

$$A_{i,j,k} = (I_{i,j,k} - \mu_k)$$

for all i, j, k.

5. Encode A as input tensor T of dimensions (1, 3, 300, 300) for the neural network model.
6. Run a forward pass through the neural network model using T as input to obtain the output tensor T' of dimensions (1, 1, N, 7), where N is the number of detected faces prior to filtering.
7. Cast T' as a matrix F of size (N, 7), where each row of F contains the data of the detected face bounding boxes:
    a. Column 3 specifies the confidence
    b. Columns 4-5 specify the relative (i.e., scaled to the interval [0,1]) x- and y-coordinates, respectively, of the upper left corner.
    c. Columns 6-7 specify the relative (i.e., scaled to the interval [0,1]) x- and y-coordinates, respectively, of the lower right corner.
8. Delete all rows i in F, where one of the following conditions is fulfilled:
    a. The confidence is lower than T, i.e.

$$F_{i,3} < T$$

    b. The relative width of the face bounding box is smaller than *M, i.e.*

$$F_{i,6} - F_{i,4} < M \cdot w$$

    c. The face bounding box protrudes the image boundary by more than a fraction E of the image width or height, respectively, i.e.

$$F_{i,4} \leq -E \text{ or } F_{i,6} \geq (1 + E) \text{ or } F_{i,5} \leq -E \text{ or } F_{i,7} \geq (1 + E).$$

9. If the resulting Table is empty (no rows remain), return an empty list.
10. For each row i, obtain the coordinates (a_i, b_i) and (c_i, d_i) of the upper left corner and the lower right corner of the face bounding box within the input image, respectively, as $a_i = \lfloor w \cdot F_{i,4} - p_1 \rceil$, $b_j = \lfloor h \cdot F_{i,5} - p_2 \rceil$, $c_i = \lfloor w \cdot F_{i,6} - p_1 \rceil$, $d_j = \lfloor h \cdot F_{i,7} - p_2 \rceil$, where the rounding function $\lfloor x \rceil$ is defined as $\lfloor x + 0.5 \rfloor$ for positive x and as $\lceil x - 0.5 \rceil$ for negative x.
Note, that the coordinates (a_i, b_i, c_i, d_i) may lie outside the region of the input image.
11. Sort the rows i of F by the area $(c_i - a_i) \cdot (d_i - b_i)$ of the face bounding boxes.
12. Return the list of coordinates (a_i, b_i, c_i, d_i) for all face bounding boxes, i.e. for all rows i.

4.1.1.4 RetinaFace Face Detector

The algorithm takes as input an image I and the following optional parameters:

- T: The minimum value for the confidence (output by the CNN) of the detected faces. Detected faces with a lower confidence are discarded. The default value in the original implementation on GitHub is 0.1, but when using an iteration over the input image size (see below), 0.6 was found to be optimal and used in the evaluation.
- M: The minimum width of the face bounding boxes relative to the width w of the input image. Detected faces with a bounding box width smaller than $M \cdot w$ are discarded. For all datasets considered in the evaluation, $M = 1/20$ was found to be eligible and used in the evaluation.

- m: The square root of the maximum number of pixels in the input image given to the CNN model. As in the original implementation, m = 500 was used.

To initialise the algorithm, the CNN model files of the face detector are loaded using the OpenCV function readNetFromCaffe(), which creates a neural network model object *M*. Note that the CNN model files are in Caffe format and comprise a prototext file and binary protocol buffer file. The algorithm takes as input an image *I* in BGR colour channel order with 8 bits per channel.

1. Determine the height h and the width w of I.
2. If $w \cdot h > m^2$, resize I to size (height, width) $\lfloor m \cdot \sqrt{w/h} \rfloor \times \lfloor m \cdot \sqrt{h/w} \rfloor$ with bilinear interpolation, and update w and h to the new width and height.
3. Normalise I using mean $\mu = (104, 117, 123)$ and standard deviation $\sigma =$ (1,1,1) to obtain the array A, i.e. set

   $$A_{i,j,k} = (I_{i,j,k} - \mu_k)$$

   for all i, j, k.
4. Encode A as input tensor T of dimensions (1, 3, w', h') for the neural network model.
5. Run a forward pass through the neural network model using T as input to obtain the output tensor T' of dimensions (1,1,N,7), where N is the number of detected faces prior to filtering.
6. Cast T' to a face bounding box table F of size (N,7), where each row of F contains the data of one of the detected face bounding boxes:
   a. Column 3 specifies the confidence
   b. Columns 4-5 specify the relative (i.e. scaled to the interval [0,1]) x- and y-coordinates, respectively, of the upper left corner.
   c. Columns 6-7 specify the relative (i.e. scaled to the interval [0,1]) x- and y-coordinates, respectively, of the lower right corner.
7. Delete all rows i in F, where one of the following conditions is fulfilled:
   a. The confidence is lower than T, i.e.,

      $$F_{i,3} < T$$

   b. The relative width of the face bounding box is not smaller than *M, i.e.,*

      $$F_{i,6} - F_{i,4} \geq M \cdot w$$

   c. The face bounding box protrudes the image boundary, i.e.,

      $$F_{i,4} \leq 0 \text{ or } F_{i,6} \geq 1 \text{ or } F_{i,5} \leq 0 \text{ or } F_{i,7} \geq 1.$$
8. If the resulting Table is empty, return an empty list.
9. Delete the first 3 columns of *F.*
10. Multiply columns 1 and 3 by w and columns 2 and 4 by h, and round the results to the next integer. After this step, the columns 1-4 contain the coordinates (a_i, b_i, c_i, d_i) where (a_i, b_i) are the absolute coordinates of the upper left corner and (c_i, d_i) are the absolute coordinates of the lower right corner of the corresponding face bounding box.
11. Sort the rows i of F by the area $(c_i - a_i) \cdot (d_i - b_i)$ of the face bounding boxes.
12. Return the list of coordinates (a_i, b_i, c_i, d_i) for all face bounding boxes, i.e. for all rows i.

4.1.2 Evaluation

Typical sets used for benchmarking face detection algorithms, such as WIDER FACE, contain extremely challenging images, many of which depict a large number of very small faces. A few examples are shown in Figure 4. These images are vastly different from those typically encountered in the biometric applications targeted by OFIQ and, therefore, these test sets are not suitable for evaluating the face detection algorithms intended for OFIQ.

Figure 4: Challenging (yet typical) images from WIDER FACE.

In order to evaluate the face detection algorithms, we conducted face detections on all images of the following datasets using each algorithm (with varying parameters):

- AFLW2000 [19]: This dataset comprises 2,000 images, the majority of which typically show only one face, with few exceptions. It covers a wide range of poses (including many full profile images), illumination conditions, expressions, cropped faces, and occlusions. Therefore, this dataset represents a significant challenge for face detection algorithms.

Figure 5: Challenging images in AFLW2000.

- VGGFace2 subset: We selected a random subset of 5,000 images from the VGGFace2 dataset [15], which contains in-the-wild images of celebrities. The images typically depict only one face with a wide range of resolution, illumination conditions, occlusions, but also distortions, occlusions by printed text, and artistic image processing (e.g., for covers of audio albums). However, most images show the face with limited deviation from a frontal pose, and only a small proportion are profile images. Therefore, this dataset is less challenging for face detection algorithms than AFLW2000 with respect to poses but comparable challenging in all other respects.

Figure 6: Challenging images in the VGGFace2 subset.

We computed the detection failure rate as the number of images, for which either the face detector didn't find any face or output a face bounding box that is considerable wrong. To this end, the images were cropped to the faces bounding boxes found by the face detectors and the cropped images were manually verified for correctness. The failure rates are listed in the following table.

Table 2: Detection failure rates of the considered face detectors on AFLW2000 and the VGGFac2 subset.

Algorithm	AFLW2000	VGGFace2 (Subset)
Dlib	24.6%	11.0%
MediaPipe Long Range	14.8%	4.3%
MediaPipe Short Range	0.52%	0.34%
SSD	0.24%	0.16%
RetinaFace	1.6%	0.16%

Obviously, dlib and MediaPipe‘s long range model perform significantly worse than the other algorithms. On AWLW2000, SSD has the lowest failure rate and MediaPipe’s short range model is second best. Conversely, on the VGGFace2 subset, RetinaFace is on par with SSD and performs better than MediaPipe’s short range model. However, it is worth noting that 97% of the failure cases of RetinaFace on AFLW2000 are full-profile images, compared to only 58% for SSD. From this, we infer that RetinaFace performs worse than SSD and MediaPipe’s short range model on images with extreme poses, while it performs equally well as SSD for images with limited poses.

The processing times per image (excluding the initial loading of the model) on a PC with a 3.7 GHz Intel Core i9-10900X CPU are given in the following Table. For dlib, the running time depends on the image dimension and, thus, the time for images of 300pixels height (and correspondingly scaled image width) is specified.

Table 3: Processing times of the face detection algorithms

Algorithm	Processing time per image
Dlib (300 pixels height)	105 ms
MediaPipe Long Range	2 ms
MediaPipe Short Range	5 ms
SSD	30 ms
RetinaFace	20 ms

The SSD face detector was included with different parameter settings (see Section 4.1.1.3) in the submissions secunet_001 - secunet_004 to the NIST FATE Quality SIDD track. The evaluation results of the measurement Total Number of Faces from [16] are shown in Figure 7.

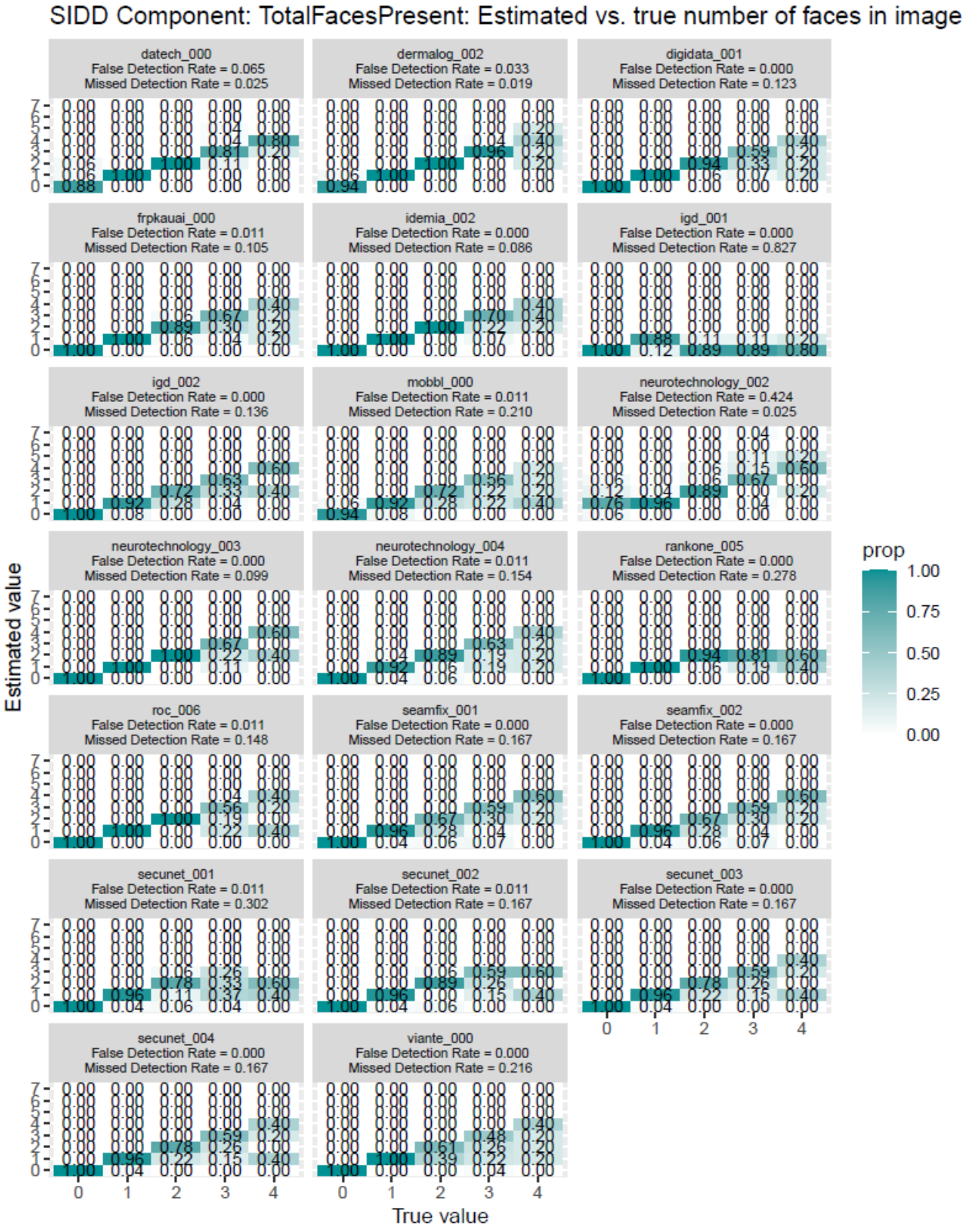

Figure 7: Results for Number of Detected Faces from the NIST FATE Quality SIDD report [16]. The submissions secunet_003 and secunet_004 are identical.

Obviously, the decrease of the minimal face width M to 1/50 in secunet_002 reduced the missed detection rate considerably, while the decrease of the confidence threshold T from 0.6 to 0.4 in combination with the padding of the input image by P=0.2 decreased the false detection rate to 0.

4.1.3 Final Algorithm Selection

For the Face Detection, the SSD face detector algorithm specified in Section 4.1.1.3 was selected and implemented in OFIQ using the following parameters:

- $P = 0.2$
- $T = 0.4$
- $M = 1/20$
- $E = 0$

4.2 Facial Landmark Estimation

Facial landmarks are supposed to pinpoint in the image the location of predefined key points, which typically represent salient points like corners or boundaries of the eyes, mouth, nose, eye brows and face boundary. Different (indexed) sets of key points can be defined, depending on the demands of the applications.

4.2.1 Selection and Prototyping of Candidate Algorithms

The following facial landmark estimation algorithms were implemented:

- Dlib. The dlib library provides a predictor for 68 2-dimensional facial landmarks baed on [20], utilizing an ensemble of regression trees in a 96 MB large model, which was trained on the iBUG 300-W face landmark dataset.[9]
- MediaPipe. The MediaPipe library provides a function to compute a face mesh, a large set of 3-dimensional facial landmarks, based on the approach of [21]. Theses landmarks do not only mark distinctive parts of the face but also encompass the areas between them (e.g., on the cheeks and chin) and the forehead.[10] This face mesh can be computed using one of two different CNN models, the *Face Landmarks Model* than outputs 468 3D-landmarks, and the Attention Mesh Model which "applies attention to semantically meaningful face regions" (specifically, the eyes and mouth) and outputs 478 3D-landmarks, with 10 of them marking the irises. In the Python package *mediapipe*, the face mesh computation is hard-wired with the MediaPipe face detector, thus, the prototype implementations used MediaPipe landmarks in combination with the MediaPipe face detector.
- ADNet. In commonly used benchmarks, ADNet [22] is one of the most accurate landmark estimation algorithms. As of September 2024, in the leader boards for the task "Face Alignment" on the web site Papers with Code[11], it ranks 5th on 300W and 9th on WFLW.

4.2.1.1 Dlib Facial Landmark Estimation

To initialise the algorithm, the facial landmark predictor object *P* is created by invoking the function shape_predictor (Modelpath) of dlib with Modelpath being the file path of the dlib shape predictor model. The algorithm takes as input an image *I* in RGB colour channel order with 8 bits per channel and the coordinates (a, b, c, d) of the first, i.e. the largest, face bounding box output by any of the face detection algorithms described in Section 4.1.

1. If the bounding box was computed with the SSD face detector, set $b = b - 0.18 \cdot (d - b)$ to remove the forehead region.
2. Convert (a, b, c, d) to a dlib rectangle object *R*.
3. Invoke the facial landmark predictor object *P* with *I* and *R* as arguments to obtain a dlib Full Object Detection object *O*, i.e. $D = P(I, R)$.
4. Invoke member function parts() of *D* to retrieve a dlib Points object *P* (which is a list of 68 dlib Point objects), i.e. *P=D.parts()*.
5. For each Point object P_i in *P*, obtain the coordinates (x_i, y_i) of the landmark within image *I* as $x_i = P_i.x$ and $y_i = P_i.y$.
6. Output all landmarks.

[9] http://dlib.net/face_landmark_detection.py.html
[10] https://google.github.io/MediaPipe/solutions/face_mesh
[11] https://paperswithcode.com/task/face-alignment

The semantic and location of the 68 landmarks output by dlib is shown in Figure 8

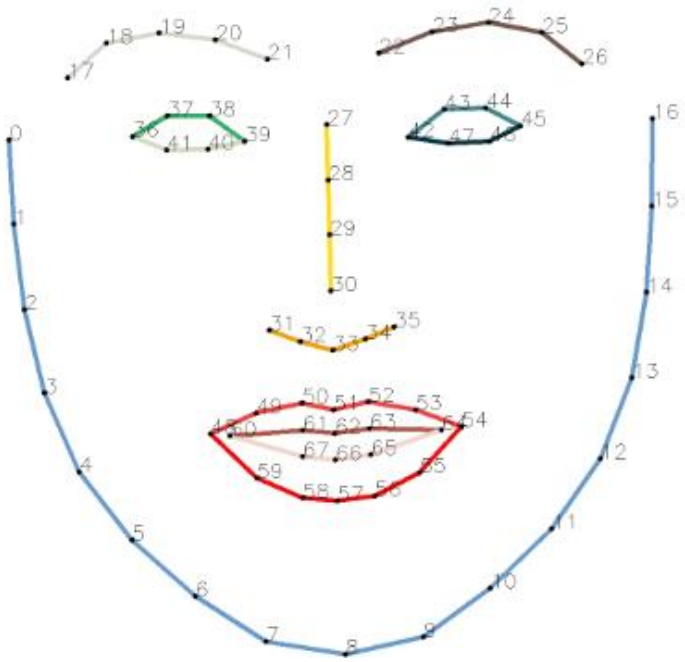

Figure 8: Semantic and location of the landmarks output by dlib.

4.2.1.2 MediaPipe Facial Landmark Estimation

Since the preliminary tests revealed that the landmarks output by the Attention Mesh Model were considerably more accurate than those of the basic Face Landmark Model, we only used the Attention Mesh Model, and reduced the landmarks to 2D by omitting the 3rd coordinate. In the Python package *mediapipe*, the model can be selected by setting the parameter "refine_landmarks" in the initialisation of the face mesh function.

The Python implementation of MediaPipe's landmarks estimation algorithm includes a face detection step and, thus, takes as input only the image.

To initialise the algorithm, the face mesh estimator object *P* is created by invoking the function face_mesh of the submodule MediaPipe.solutions, and *P* is then initialised by calling its function FaceMesh with the parameters static_image_mode=True, max_num_faces = 1, refine_landmarks=True and min_detection_confidence=0.5.

The algorithm takes as input an image *I* in RGB colour channel order with 8 bits per channel.

1. Compute the width *W* and height *H* of image *I*.
2. Call the function *process* of object *P* with *I* as argument to obtain a SolutionOutputs object *R*.
3. If *R* is None, return an error ("no face found").
4. Obtain the NormalisedLandmarkList object *L* as the first element in *R*. multi_face_landmarks.
5. Obtain the list of 478 NormalisedLandmark objects L_1, …, L_{478} as L. landmark.
6. For all I from 0 to 477, obtain the relative coordinates $(\hat{x}_i, \hat{y}_i)$ of the *i*-th landmark within image *I* as L_i.x and L_i.y.
7. Compute the absolute coordinates (x_i, y_i) of the *i*-th landmark within image *I* as $x_i = \lfloor \hat{x}_i \cdot W \rfloor$ and $y_i = \lfloor \hat{y}_i \cdot H \rfloor$.
8. Output all landmarks.

The semantic and location of the 478 landmarks output by MediaPipe are shown in https://github.com/google/mediapipe/tree/master/mediapipe/modules/face_geometry/data/canonical_face_model_uv_visualization.png.

4.2.1.3 ADNet Facial Landmark Estimation

Since the official implementation of ADNet including the trained models has been published under MIT license[12], it was decided to test it for OFIQ. In the repository, three models are available:

- A model trained on COFW computing 29 landmarks.
- A model trained on 300W computing 68 landmarks (same semantic as dlib, see Figure 8).
- A model trained on WFLW computing 98 landmarks (see Figure 9).

The WFLW model was selected (98 landmarks) and converted to ONNX. Unfortunately, the ONNX model cannot be executed with OpenCV's DNN module, thus, ONNXRuntime was used. The pre-processing (face detection, cropping, pixel value normalisation) was empirically adapted to use the SSD face detector.

To initialise the algorithm, an ONNXRuntime inference session is created by loading the ADNet model. The algorithm takes as input an image I in BGR colour channel order with 8 bits per channel and the coordinates (a, b, c, d) of the first, i.e. the largest, face bounding box output by the SSD face detection algorithm described in Section 7.

1. Determine the height h and the width w of the image.
2. Extend the face bounding box on both sides to square shape
   a. If $d - b > c - a$, update a, c to $\left\lfloor \frac{a+c}{2} - \frac{d-b}{2} \right\rfloor, \left\lceil \frac{a+c}{2} + \frac{d-b}{2} \right\rceil$.
   b. If $d - b < c - a$, update b, d to $\left\lfloor \frac{b+d}{2} - \frac{c-a}{2} \right\rfloor, \left\lceil \frac{b+d}{2} + \frac{c-a}{2} \right\rceil$.
3. If necessary, pad the image and update the face bounding box coordinates accordingly
   a. Set $p_{\text{left}} = \max(0, -a)$, $p_{\text{right}} = \max(0, c - w + 1)$, $p_{\text{top}} = \max(0, -b)$, and $p_{\text{bottom}} = \max(0, d - h + 1)$.[13]
   b. Pad the image from left, top, right, bottom side by $p_{\text{left}}, p_{\text{top}}, p_{\text{right}}, p_{\text{bottom}}$ pixels, respectively.
   c. Update a, b, c, d to $a + p_{\text{left}}, b + p_{\text{top}}, c + p_{\text{left}}, d + p_{\text{top}}$
4. Crop I to the rectangle defined by the corners (a, b) and (c, d).
5. Resize the image I to size 256×256 using OpenCV with bilinear interpolation.
6. Normalise I using mean $\mu = 127.5$ and standard deviation $\sigma = 127.5$ to obtain the array A, i.e. set
   $$A_{i,j,k} = (I_{i,j,k} - 127.5)/127.5$$
   for all i, j, k.
7. Encode A as input tensor T of dimensions (1, 3, 256, 256) for the neural network model.
8. Run a forward pass through the neural network model using T as input to obtain a list of 14 tensors $(T'_1, \ldots, T'_{14})$. The last tensor T'_{14} has dimensions (1, 98, 2).
9. Cast T'_{14} to an array A of dimensions (98,2).
10. De-Normalise A to obtain the array B specifying the landmark positions within the cropped and resized image by setting
    $$B_{i,j} = 127.5 \cdot (A_{i,j} + 1)$$

[12] https://github.com/huangyangyu/ADNet
[13] We assume that the pixel coordinates start with 0.

for all i, j.

11. Compute the list of landmarks (x_i, y_i) within the original input image I as $x_i = \lfloor B_{i,1}(d-b)/256 + a - p_{\text{left}} \rceil$ and $y_i = \lfloor B_{i,2}(d-b)/256 + b - p_{\text{top}} \rceil$, where the rounding function $\lfloor x \rceil$ is defined as $\lfloor x + 0.5 \rfloor$ for positive x and as $\lceil x - 0.5 \rceil$ for negative x.

12. Output all landmarks $(x_0, y_0), \ldots, (x_{97}, y_{97})$.

Note, that landmarks can lie outside the image region, i.e. may have coordinates smaller 0, x-coordinates larger or equal than w, or y-coordinates larger or equal than h.

The semantic and location of the 98 landmarks output by ADNet is shown in Figure 9. Note that the

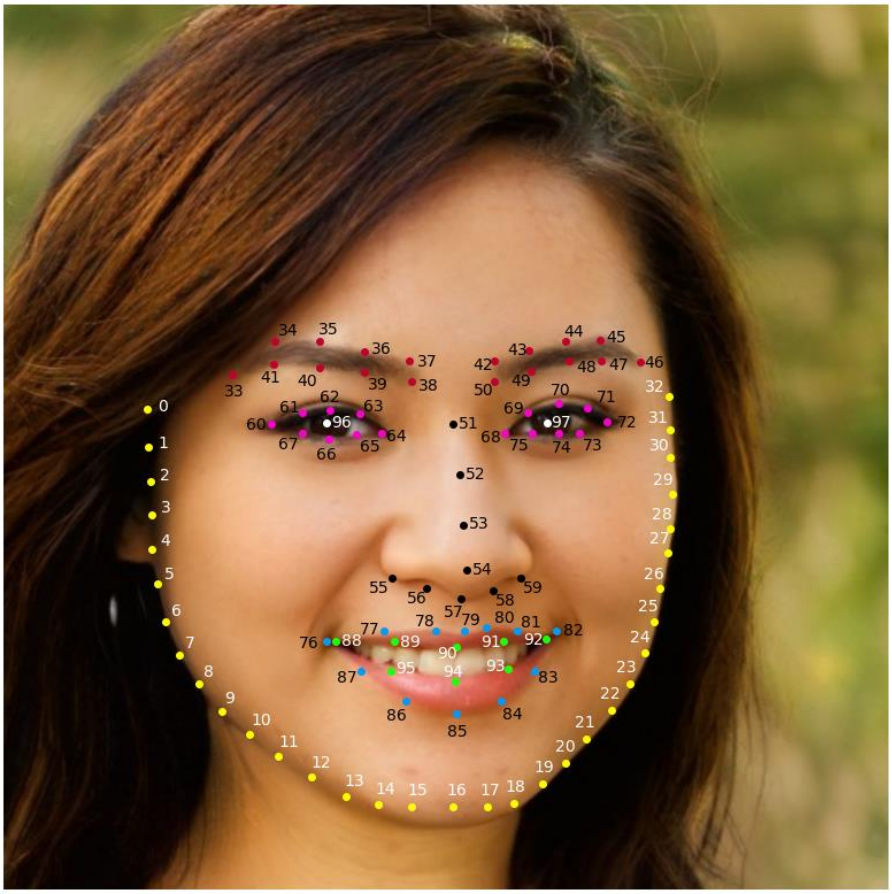

Figure 9: Semantic and location of the 98 landmarks output by ADNet.

landmarks 96 and 97 are not the centres of the eyes but the location of the pupil.

4.2.2 Evaluation

The accuracy of the estimated landmarks is not evaluated explicitly but only implicitly by the evaluation of quality component algorithms than primarily rely on landmarks, in particular, for the components Eyes Open and Mouth Closed.

The dlib landmarks estimation algorithm never fails to output landmarks if a valid bounding box (inside the image region and of non-zero area) is given as input along with the image. (However, for faces with a large yaw angle, the landmarks are typically completely wrong.) In contrast, the MediaPipe landmark estimation can fail, even in cases where the MediaPipe face detector succeeds. On AFLW2000 [19], MediaPipe fails for 7.9% of the images, which is a considerably higher failure rate than that of MediaPipe's face detection with the Short Range Model, but on the subset of VGGFace2 used to evaluate the face detection algorithms (See Section 4.1), it fails for only 0.6% of the images. As visible in Table 11 and Table 13, a drastic decrease of the error rates is observed for the use of ADNet, which proves the superiority of ADNet over mediapie and

dlib. The ADNet landmarks estimation algorithm never fails to output landmarks if a valid bounding box (of non-zero area) is given as input along with the image.

The processing times per image (excluding the initial loading of the model) on a PC with a 3.7 GHz Intel Core i9-10900X CPU are given in the following Table. There, ADNet was executed with ONNXRuntime.

Table 4: Processing times of the face detection algorithms

Algorithm	**Processing time per image**
Dlib	3 ms
MediaPipe	6 ms (incl. face detection)
ADNet	120 ms

4.2.3 Final Algorithm Selection

For the Facial Landmark Estimation, the ADNet algorithm specified in Section 4.2.1.3 was selected and implemented in OFIQ.

4.3 Face Segmentation

Several kinds of face segmentations are used in this report:

- Landmarked Region Segmentation: Segmentation of the inner face region using landmarks.
- Occlusion Segmentation: Segmentation of occluded and un-occluded regions of the face.
- Face Parsing: Segmentation of individual parts of the subject, in particular face skin, face parts (e.g. mouth, nose, eyes, eyebrows), ears, hair, neck, clothing, head coverings, glasses.

4.3.1 Landmarked Region Segmentation

The Landmarked Region Segmentation is intended to separate the *inner face region* as specified in Clause 3.39 of ISO/IEC 39794-5:2019 [1], as the face region from the chin to (and including) the eyebrows and between (but excluding) the ears. This area excludes the forehead above the eyebrows and is typically delineated by landmarks (MediaPipe also outputs landmarks on the forehead, but these are outside the inner face region considered here). An example of the inner face region is shown in Figure 10.

Note that, if the head pose has a considerable yaw angle, on one side of the face, parts of the boundary of this inner face region may not be visible (occluded by other parts of the face), while, on the other side of the face, the exact boundary of the inner face region may not be clearly distinguishable.

In the NIST FATE Quality SIDD track, the measure FaceOcclusion requires segmentation of a different face region: There, the face region extends to "the point between and above the eyes that lies on the line from the midpoint of the eyes to the chin, and extends above the eyes by 85% of the distance from the midpoint of the eyes to the chin" [23].[14] (This definition is similar to the definition of the face region in Clause D.1.4.2.8 of ISO/IEC 39794-5:2019 [1], which expands "from crown to chin and from the left ear to the right ear"). For this reason, the landmarked region segmentation algorithm was parametrised to support both face region definitions.

4.3.1.1 Selection and Prototyping of Candidate Algorithms

The landmarked region is computed with respect to the aligned image.

The algorithm takes as input the transformed landmarks L_1, …, L_n output by the alignment algorithm in Section 4.4.1 and a parameter $\alpha \geq 0$:

1. If MediaPipe landmarks are used, discard all landmarks on the forehead except those at the eyebrows. Precisely, discarded the following landmarks: 162, 21, 54, 103, 67, 109, 10, 338, 297, 332, 284, 251, 389, 139, 71, 68, 104, 69, 108, 151, 337, 299, 333, 298, 301, 368, 9.
2. If $\alpha > 0$ perform the following steps to approximate the region of the forehead:
    a. Calculate the centre of the subject's right eye as the midpoint $P_l = (L_i + L_j)/2$ of the landmarks L_i and L_j of its inner and outer canthi.
        - For dlib, use i=39 and j=36.
        - For MediaPipe, use i=133 and j=33.
        - For ADNet, use i=64 and j=60.
    b. Calculate the centre of the subject's left eye as the midpoint $P_l = (L_i + L_j)/2$ of the landmarks of its inner and outer canthi.

[14] In December 2023, a new measure Face Occlusion 2 was added, using the same definition of the face region as in Clause 6.6 of ISO/IEC FDIS 29794-5:2024 [10].

- For dlib, use i=42 and j=45.
- For MediaPipe, use i=362 and j=263.
- For ADNet, use i=68 and j=72

c. Compute the midpoint of the eyes as $P_c = (P_l + P_r)/2$.

d. Set C to the position of the landmark L_i typically marking the lowest end of the chin.

- For dlib, use i=8.
- For MediaPipe, use i=152.
- For ADNet, use i=16.

e. Extend line segment (C, P_c) beyond P_c (i.e. upward) by a fraction of α of its length and define the forehead top point P_f as the upper end of the extended line segment.

f. Select two landmarks L_{i_1}, L_{i_2} that the left and right side of the face contour roughly at the height of the eyes.

- For dlib, use i_1=0 and i_2=16.
- For MediaPipe, use i_1=127 and i_2=288.
- For ADNet, use i_1=0 and i_2=32.

g. Select two landmarks L_{i_3}, L_{i_4} that the left and right side of the face contour roughly at the height of the Mouth.

- For dlib, use i_1=4 and i_2=12.
- For MediaPipe, use i_1=356 and i_2=58.
- For ADNet, use i_1=7 and i_2=25.

h. Fit an ellipse E though the points $P_f, C, L_{i_1}, L_{i_2}, L_{i_3}, L_{i_4}$.

i. Define 36 points Y_1, ... Y_{36} bounding ellipse E, starting at the left intersection of the small semi-axis with the ellipse and setting a new landmark at each 10 degrees.

j. Discard ellipse points that are not on forehead: Remove all points Y_i, for which the projection of the line segment (C, Y_i) onto the line through C and P_c is shorter than the line segment (C, P_c).

k. Amend the landmarks by the remaining ellipse points $Y_{i_1}, \dots, Y_{i_m}$, i.e. set $L_{n+1} = Y_{i_1}, \dots, L_{n+m} = Y_{i_m}$.

3. Use the OpenCV function convexHull() to determine the landmarks $L_{j_1} \dots, L_{j_k}$ that define the convex hull of all landmarks.
4. Generate a segmentation map S as grayscale image (one channel), where all pixels inside the convex hull are set to 1 and all pixels outside are set to 0.
5. Output S.

For $\alpha = 0$, the algorithm outputs the inner face region used by the other quality component algorithms in this report and by the measure Face Occlusion 2 in NIST FATE Quality [23]. For $\alpha = 0.85$, the algorithm extends the face region upward as specified for the measure Face Occlusion in NIST FATE Quality [23]. These two segmentations are illustrated in Figure 10.

Figure 10: Example of the landmarked region segmentation (after alignment) with α=0 (left) and α=0.85 (right).

4.3.1.2 Evaluation

The landmarked region segmentation algorithm is an essential pre-processing for the face occlusion detection algorithm specified in Section 7.6.2.1 and, thus, its accuracy with different landmark estimation algorithms is implicitly evaluated by the evaluation of this face occlusion detection algorithm.

4.3.1.3 Final Algorithm Selection

For the Landmarked Region Segmentation, the algorithm specified in 4.3.1.1 was selected and implemented in OFIQ using the parameter $\alpha = 0$.

4.3.2 Face Parsing

For several quality components (Background Uniformity, No Head Coverings, No Occlusion of the Face), it is very useful to perform face parsing, i.e. segmentation of the face image into the distinguished classes representing different parts of the face (e.g., eyes, nose, mouth, ears, residual face region), or subject (hair, glasses, head coverings, clothing and background). The output of the face parsing is a segmentation map, i.e. an image, in which each colour value corresponds to one class.

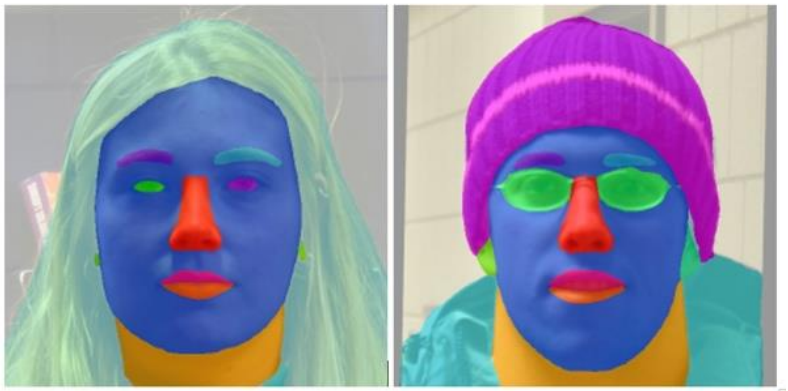

Figure 11: Example visualisations of outputs of the face parsing algorithm

Two different CNN models for face parsing were identified as eligible with respect to the distinguished image parts (e.g. including head coverings), model size and license:

1. The model published in the official implementation of [24].[15]
2. The model published in the Github repository face-parsing.PyTorch.[16]

However, in preliminary tests, the model from official implementation of [24] gave poor results. Thus, only the model from the Github repository face-parsing.PyTorch was implemented as a prototype and evaluated.

4.3.2.1 Selection and Prototyping of Candidate Algorithms

The algorithm is based on the CNN model the Github repository face-parsing.PyTorch, which generates a segmentation map distinguishing the following 19 classes: Left eye, right eye, left eyebrow, right eyebrow, nose, upper lip, lower lip, residual mouth region (between the lips), left ear, right ear, residual face region (face skin), earrings, glasses, neck, necklace, clothing, hair of head, head covering, background. Facial hair (beards, moustaches) and occlusions of the face by hands or objects (other than hair, glasses or head coverings) are not distinguished from the underlying face skin.

Since the face parsing model has been trained on aligned images, the input images are also aligned and then, cropped and scaled by the algorithm below to resemble the alignment used for the training set. However, the alignment applied in the algorithm below slightly deviates from the alignment used for training:

- In order to speed up the computation, the input image size was changed to 400x400 pixels as compared to 512x512 used for training. Only a small decrease of accuracy by this reduction was observed.
- In order to ensure that, in most cases, hair and head coverings are completely contained in the input image, the images aligned using the algorithm specified in Section 4.4.1 are cropped from both sides and from the bottom, but not at the top, resulting in a slightly larger extension of the scenery on the top as compared to the training set.

This deviation from the alignment of the face in the input image only results in a moderate decrease of accuracy of the segmentation, in particular for those image parts used for quality assessment (head coverings, background).

The CNN model has been converted to ONNX and is executed with ONNXRuntime (resulting in much smaller processing times as compared to OpenCV). The algorithm takes as input a face image I of dimension 616x616 output by the alignment algorithm specified in Section 4.4.1.

1. Convert I to RGB colour channel order.
2. Crop the image by 30 pixels from both sides and by 60 pixels from the bottom to obtain image I' of size (556x556).
3. Scale image I' to 400x400 pixels using bilinear interpolation to obtain image I''.
4. Normalise I with mean $\mu = (123.7, 116.3, 103.5)$ and standard deviation $\sigma = (58.4, 57.1, 57.4)$ to obtain the array A by setting

   $$A_{i,j,k} = (I_{i,j,k} - \mu_k)/\sigma_k$$

   for all i, j, k.[17]
5. Reshape A to an input tensor T of dimensions (1, 1, 400, 400)
6. Run a forward pass through the face parsing model using T as input to obtain an output tensor T' of dimensions (19, 400, 400)

[15] https://github.com/switchablenorms/CelebAMask-HQ/tree/master/face_parsing
[16] https://github.com/zllrunning/face-parsing.PyTorch
[17] This is equivalent, up to rounding, to first dividing the pixel values by 255 and then perform normalisation with mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225).

7. Compute the argmax on first axis of T' to obtain the segmentation map S of size 400x400.
8. Return the segmentation map S.

4.3.2.2 Evaluation

An evaluation of the face parsing algorithm is implicitly performed by the evaluation of the algorithms that rely on it, specifically, those described in Sections 6.1.2 and 7.12.2.

4.3.2.3 Final Algorithm Selection

For the Face Parsing, the algorithm specified in Section 4.3.2.1 was selected and implemented in OFIQ.

4.3.3 Face Occlusion Segmentation

The face occlusion segmentation outputs a segmentation map that marks the parts of the face that are not occluded.

Figure 12: Example visualisations of outputs of the face occlusion segmentation algorithm

4.3.3.1 Selection and Prototyping of Candidate Algorithms

The CNN model used is taken from the repository FaceExtraction[18], which is the official implementation of the method described in [25] and is also based on [26], [27], and [28]. According to [25], the occlusion labels of the data set FaceOcc used for training were defined as follows:

- Any objects in front of the face are considered as occlusion. This includes hands, protective faces masks, any kind of head coverings, scarves, band-aid or bandage, mobile phones.
- Beards, moustaches and eye brows are not considered as occlusion
- Hair of the head occluding facial skin is considered as occlusion
- Opaque lenses of glasses are considered as occlusion
- Transparent lenses of glasses are not considered as occlusion except strong specular reflections
- Frames of eyeglasses are considered as occlusion
- A tongue poked out of the mouth is considered as occlusion
- Makeup is only considered as occlusion if there is a clearly visible border to face skin, which is only the case if the makeup has a clearly unnatural colour
- Self-occlusion of the face by extreme poses are not considered as occlusion
- Shadows on the face are not considered as occlusion

[18] https://github.com/face3d0725/FaceExtraction

Since the CNN model had been trained on aligned images, the input images are also aligned and, then, cropped and scaled by the algorithm below to resemble the alignment used for the training set. However, in order to speed up the computation, the images are scaled to size 224x224 as compared to 256x256 used for the training. This deviation only results in a moderate decrease of accuracy of the segmentation.

The CNN model was converted to ONNX and is executed with ONNXRuntime (resulting in much smaller processing times as compared to OpenCV).

The algorithm takes as input the face image *I* of dimension 616x616 output by the alignment algorithm specified in Section 4.4.1.

1. Convert *I* to RGB colour channel order.
2. Crop image *I* from all sides by 96 pixels to obtain an image of size 424x424.
3. Scale image *I* to 160x160 pixels using bilinear interpolation to obtain image *I''*.
4. Normalise *I* using mean $\mu = 0$ and standard deviation $\sigma = 255$ to obtain the array *A*, i.e. set

   $$A_{i,j,k} = I_{i,j,k}/255$$

   for all i, j, k.
5. Encode *A* as input tensor T of dimensions (1, 1, 224, 224)
6. Run a forward pass through the face parsing model using T as input to obtain an output tensor T' of dimensions (1, 1, 224, 224)
7. Create a bit mask *M* of size 224x224 with

   $$M_{i,j} = 1 \text{ if } T'_{1,1,i,j} > 0 \text{ and } M_{i,j} = 0 \text{ else}$$

   for all i, j, k.
8. Resize *M* to size 424x424 using nearest neighbour interpolation.
9. Pad *M* from all sides by 96 pixels setting the padded regions to 0 (indicating occlusion) to obtain a segmentation map of 616x616 pixels corresponding to the face image given as input.
10. Return the segmentation map *M*.

4.3.3.2 Evaluation

An evaluation of the face occlusion segmentation algorithm was implicitly performed by the evaluation of the algorithms for Face Occlusion Prevention, Mouth Occlusion Prevention and Eyes Visible described in Sections 7.6.3, 7.5.3 and 7.4.3.

4.3.3.3 Final Algorithm Selection

For the Face Occlusion Segmentation, the algorithm specified in Section 4.3.3.1 was selected and implemented in OFIQ.

4.4 Face Alignment

Several CNNs used in the prototyping phase require that the face image has been aligned. This alignment ensures that, for all images, the face and its parts are roughly located at the same positions. For this purpose, target locations for distinguished key points of the face, precisely, centres of the eyes, nose tip and mouth corners, are defined and an affine transformation composed of translation, rotation and scaling[19] is estimated to approximately map the current locations of these key points to the target locations. This affine transformation is then applied to the image and the image is cropped and/or padded to obtain the required image dimensions.

For the quality components Background Uniformity and No Head Coverings, it is important that, in most cases, the hair and head covering are fully visible in the aligned image. Furthermore, the face must not be too small to ensure that the segmentation still works well. The target coordinates P' and image dimension (616x616) are selected so that this is typically fulfilled.

The CNNs used for segmentation output a segmentation map for the input image. In order to compute the segmentation map for the original image from the segmentation map of the aligned input image, the transformation matrix is returned along with the transformed image and the transformed landmarks.

4.4.1 Selection and Prototyping of Candidate Algorithms

The algorithm takes as input the image I in RGB colour channel order and the facial landmarks $L = (L_1, \ldots, L_N)$ computed using one of the algorithms in Section 4.2.1:

Figure 13: Example of a facial image after alignment

1. Compute the position P_{re} of the subject's right eye's centre as the mean of the landmarks L_i and L_j of the right eye's inner and outer corners: $P_{re} = (L_i + L_j)/2$, where i=36 and j=39 for dlib, and i=33 and j=133 for MediaPipe.
2. Compute the position P_{le} of the subject's left eye's centre as the mean of the landmarks L_i and L_j of the right eye's inner and outer corners: $P_{re} = (L_i + L_j)/2$, where i=42 and j=45 for dlib, and i=362 and j=263 for MediaPipe.
3. Compute the position P_{no} of the subject's nose tip as $P_{no} = L_i$, where i=30 for dlib and i=4 for MediaPipe.
4. Compute the positions P_{rm} and P_{lm} of the subject's right and left mouth corner as $P_{rm} = L_i$ and $P_{lm} = L_j$ where i=48 and j=54 for dlib and i=61 and j=291 for MediaPipe.
5. Construct the array $P = [P_{re}, P_{le}, P_{no}, P_{rm}, P_{lm}]$ and the target point array P' = [[251, 272], [364, 272], [308, 336], [262, 402], [355, 402]].
6. Invoke the OpenCV function estimateAffinePartial2D() with parameter method= LMedS to compute an affine transformation matrix M limited to translation, rotation and scaling, so that $M(P) \approx T$ is

[19] These affine transformations are preserving angles and, thus, do not introduce shearing.

optimised. This function first applies a Least Median of Squares regression [29] and refines the transformation further using the Levenberg-Marquardt algorithm [30].

7. Apply the OpenCV function warpAffine() to image *I* with parameters M=*M*, dsize=(616,616), borderMode = BORDER_CONSTANT and borderValue=0 to obtain an image of dimension 616x616 resulting from the application of *M* to *I* and padding with black colour (where necessary).
8. Apply the OpenCV function transform() to the landmarks *L* with parameter M=*M* to obtain the accordingly transformed landmarks *L'*.
9. Output *I'*, *M* and *L'*.

An example output is shown in Figure 13.

4.4.2 Evaluation

Since the results of different methods for face alignment differ only very slightly,[20] no evaluation is necessary.

4.4.3 Final Algorithm Selection

For the Face Alignment, the algorithm specified in Section 4.4.1 was selected and implemented in OFIQ.

[20] Different optimisation algorithms can be applied to determine the affine transformation.

5 Unified Quality Score

The unified quality score aims to predict the utility of the facial image for recognition purposes.

5.1 Data Selection

Following the testing methodology of ISO/IEC 29794-1 [31], a Unified Quality Score shall predict the recognition performance of a face recognition attempt. Therefore, the Unified Quality Score is supposed to depend on all quality criteria which are relevant for face recognition performance, and thus, we should ideally use a dataset that covers a wide range for all these criteria. Since this requirement is typically fulfilled for in-the-wild face image datasets, these are frequently used for the evaluation of unified quality scores. We use a subset of 486.000 images from the VGGFace2 dataset [15], which covers a large variety of quality levels and quality issues (see Figure 14).

Figure 14: Example images from the VGGFace2 test set with various kinds of quality issues

For evaluation using EDC curves, we use a subset of approx. 483,144 images of all 9,131 subjects.

5.2 Selection and Prototyping of Candidate Algorithms

5.2.1 MagFace-based Algorithms

MagFace [32] is an approach for training CNNs on face recognition and face image quality assessment (FIQA) simultaneously, so that the magnitude of the computed embeddings can be used as an estimate for the quality of the corresponding face image. Four trained models published in the official MagFace repository[21] were tested for computing a Unified Quality Score:

- MagFace100: the iResNet100 model trained on MS1MV2 with MagFace loss and PyTorch's DistributedDataParallel (DDP) parallelisation method
- MagFace50: iResNet50 model, trained on MS1MV2 with MagFace loss and DDP parallelisation.
- MagFace50_FP16: iResNet50 model, trained on MS1MV2 with MagFace loss without DDP parallelisation using floating point 16 arithmetic.
- MagFace18: iREsNet18 model, trained on CASIA-WebFace with MagFace loss without DDP parallelisation.

The corresponding algorithms take as input the aligned image I' in BGR colour channel order output by the alignment algorithm in Section 4.4.1, and perform the following steps:

1. Resize I' to 192x192 pixels using bilinear interpolation.
2. Crop I' by 40 pixels from left and right, by 33 pixels from the top and by 47 pixels from the bottom.
3. Normalise I' using mean $\mu = 0$ and standard deviation $\sigma = 255$ to obtain the array A, i.e. set, for all i, j, k:

$$A_{i,j,k} = I_{i,j,k}/255$$

4. Encode A as input tensor T of dimensions (1, 1, 112, 112).
5. Run a forward pass through the iResNet100 model using T as input to obtain an output tensor T' of dimensions (1, 1, 512)
6. Cast T' to a vector F of length 512.
7. Set q to the Euclidean norm of F, i.e. $q = \|F\|_2$.
8. Output q.

The output value has a higher-is-better semantic.

5.2.2 AdaFace-based Algorithms

AdaFace [33] is a methodology for training CNNs on face recognition. Its authors show that the magnitude of the embeddings correlates to face image quality, and thus, AdaFace seems to be eligible as a face image quality assessment (FIQA) algorithm, even though the authors don't propose this use case. Therefore, three models from the original AdaFace repository[22] were tested for computation of a Unified Quality Score:

- AdaFace100: ResNet100 model, trained on WebFace12M.
- AdaFace50: ResNet50 model, trained on WebFace4M.
- AdaFace18: ResNet18 model, trained on WebFace4M.

[21] https://github.com/IrvingMeng/MagFace
[22] https://github.com/mk-minchul/AdaFace

The processing differs from that of the MagFace algorithms only in the values for mean and standard deviation used for normalisation of the input tensor. The algorithms take as input the aligned image I' in BGR colour channel order output by the alignment algorithm in Section 4.4.1, and perform the following steps:

1. Resize I' to 192x192 pixels using bilinear interpolation.
2. Crop I' by 40 pixels from left and right, by 33 pixels from the top and by 47 pixels from the bottom.
3. Normalise I' using mean $\mu = 0$ and standard deviation $\sigma = 127.5$ to obtain the array A, i.e. set for all i, j, k:
$$A_{i,j,k} = \big(I_{i,j,k} - 127.5\big)/127.5$$
4. Encode A as input tensor T of dimensions (1, 1, 112, 112).
5. Run a forward pass through the iResNet100 model using T as input to obtain an output tensor T' of dimensions (1, 1, 512)
6. Cast T' to a vector F of length 512.
7. Set q to the Euclidean norm of F, i.e. $q = \|F\|_2$.
8. Output q.

For AdaFace50 and AdaFace18, the output has a higher-is-better semantic. For AdaFace100, however, for unknown reasons, the output has a higher-is-worse semantic.

5.3 Evaluation

Since the Unified Quality Score doesn't predict any specific attribute of the image, no ground truth data is available. On the other hand, the unified quality score is supposed to assess the utility of the image for face recognition and other applications of face biometrics. One established method to evaluate this utility is the Error-Versus-Discard Characteristics (EDC) curve as defined in ISO/IEC 29794-1 [31]. This method is used to analyse how well the unified quality score of an image predicts the comparison scores that can be obtained when performing a biometric comparison with another image of the same person (mated comparison).

From a subset of VGGFace2 containing 483,144 images, we construct the same number of mated (genuine) comparisons by comparing each image with the subsequent image of the same subject in lexicographic order, and the last image of a subject with its first one. From these mated comparisons, 3,299 comparisons of identical or almost (up to scaling, recompression, and slight cropping) identical images, resulting in Euclidean distances of ArcFace features below 0.3, have been discarded, resulting in a final list of 479,843 mated comparisons.

The following face recognition systems are employed:

- ArcFace ResNet100: an open-source face feature extractor based on the iResNet100 model, trained on MS1MV2. For face detection, RetinaFace was applied, using MediaPipe as fallback in case no face is detected. For this system the processing failed in 0.14% of mated comparisons.
- MagFace ResNet100: the open-source face feature extractor used for the unified quality score algorithm specified in Section 5.2.1. For face detection, SSD was used with RetinaFace as fallback if no face is detected. In 0.04% of all mated comparisons, errors occurred.
- Cognitec (Version 9.3.2.0): a commercial off-the-shelf face recognition system. In 3.07% of the mated comparisons, errors occurred for this system when performing mated comparisons.
- Paravision (Version 1.0.6): a commercial off-the-shelf face recognition system. This system failed to produce scores in 0.22% of all mated comparisons.

The resulting EDC curves are plotted in Figure 15 and Figure 17. As expected, the MagFace18 algorithm performs considerably worse as compared to the other MagFace algorithms, and the MagFace50 algorithm performs slightly worse than MagFace100. Surprisingly, MagFace50_FP16 performs best, with a very small advantage over MagFace100. On the other hand, MagFace50_FP16 runs much faster (3.5 times and 4.8 times, respectively) than MagFace50 and MagFace100.

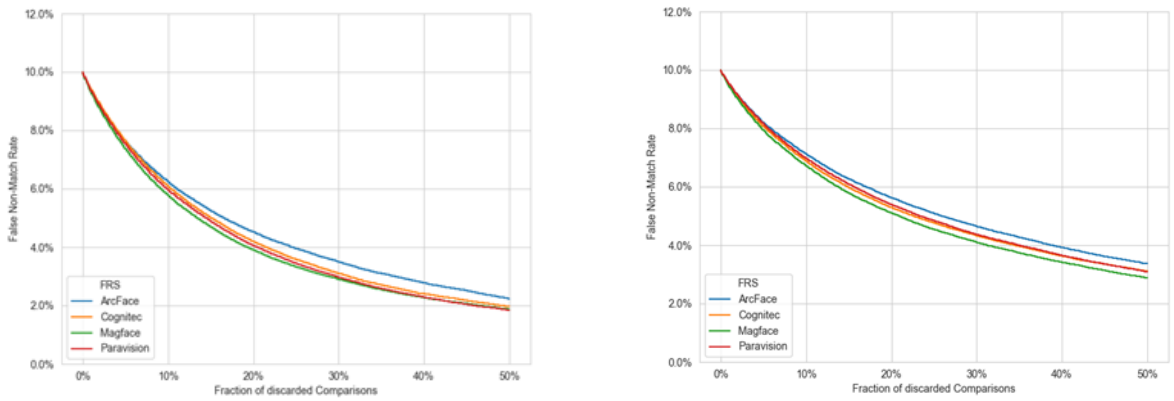

Figure 15: EDC curves for the Unified Quality Score using the MagFace50_FP16 (left) and MagFace18 (right) on the VGGFace2 test set.

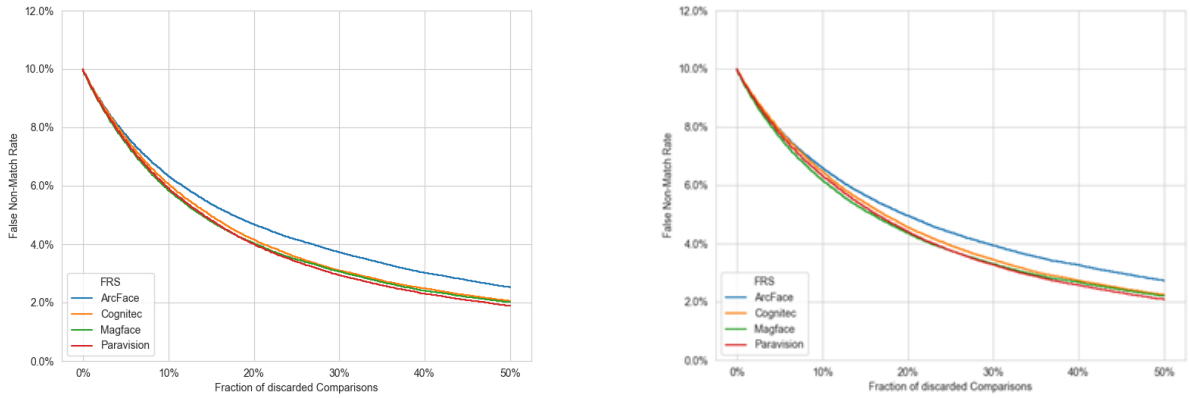

Figure 17: EDC curves for the Unified Quality Score using the MagFace100 (left) and MagFace50 (right) on the VGGFace2 test set.

The EDC curves of the AdaFace algorithms are shown in Figure 16 and Figure 18. While the curves of AdaFace100 show an almost linear reduction of the FNMR, the decline is steeper for discard rates below 20% in the EDC curves of AdaFac50. Surprisingly, the algorithm using the smallest model, AdaFace18, shows the strongest reduction of the FNMR among the AdaFace algorithms. Nevertheless, compared to the MagFace algorithms, the AdaFace18 performs worse than most MagFace algorithms.

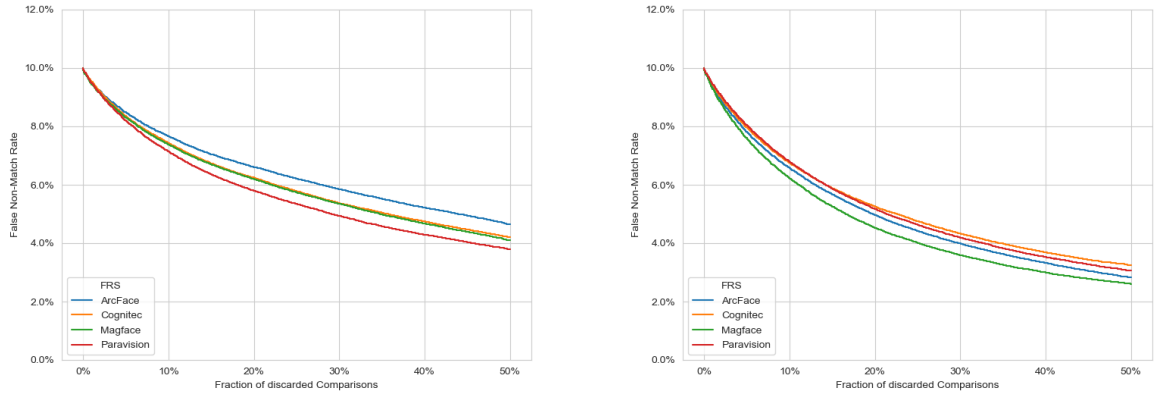

Figure 16: EDC curves for the Unified Quality Score using the AdaFace50 (left) and AdaFace18 (right) on the VGGFace2 test set.

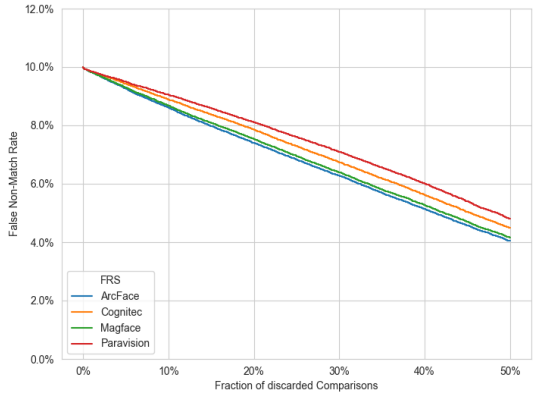

Figure 18: EDC curves for the Unified Quality Score using the AdaFace100 on the VGGFace2 test set.

For the ease of comparison, the mean FNMR values (taken over all 4 face recognition algorithms) and processing times (per image) of all algorithms are listed in Table 5.

Table 5: Mean FNMR (taken over the four face recognition algorithms) on the VGGFace2 test set achieved by the individual algorithms for Unified Quality Score at various discard rates (DR), and their mean processing times per image on a 3.7 GHz Intel Core i9-10900X CPU.

Algorithm	DR = 1%	DR = 5%	DR = 10%	DR = 20%	DR = 30%	Running Time
MagFace100	9.41%	7.61%	6.05%	4.22%	3.21%	177 ms
MagFace50	9.45%	7.82%	6.41%	4.58%	3.50%	128 ms
MagFace50_FP16	**9.37%**	**7.56%**	**6.03%**	**4.18%**	**3.13%**	37 ms
MagFace18	9.51%	8.11%	6.93%	5.36%	4.37%	15 ms
AdaFace100	9.84%	9.37%	8.81%	7.73%	6.63%	211 ms
AdaFace50	9.52%	8.34%	7.41%	6.21%	5.38%	18 ms
AdaFace18	9.41%	7.87%	6.60%	4.99%	4.03%	10 ms

The algorithms based on MagFace50_FP16 and MagFace100 were submitted to NIST FATE Quality SIDD track as part of the submissions secunet_003 and secunet_004, respectively [16]. Here, the EDC curves were computed from the average error rates of 15 different face recognition algorithms, on a test set consisting of high-quality reference photos and webcam like probe photos, with quality assessment applied only to the latter, and for baseline FNMR values of 5%, 2%, 1% and 0.5%. Again, MagFace50_FP16 showed slightly better performance than MagFace100, and even showed the highest drop of FNMR of all submitted algorithms (see Figure 19).

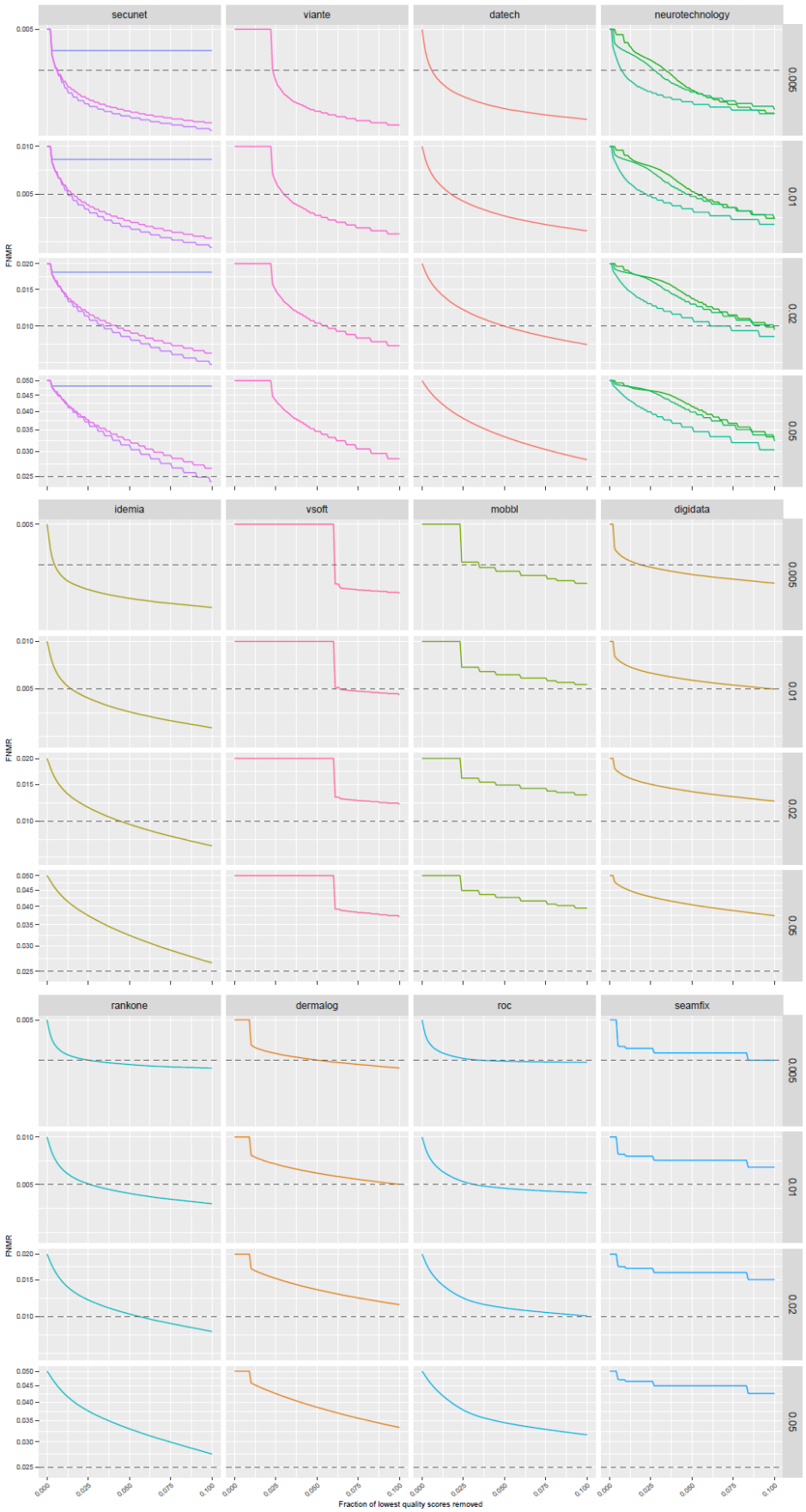

Figure 19: EDC curves for the unified quality score compiled from the NIST FATE Quality SIDD track [16]. In the upper left diagrams (labelled "secunet"), the lower curves are MagFace50_FP16 (secunet_003), and the intermediate curves are MagFace100 (secunet_004).

5.4 Final Algorithm Selection

For the Unified Quality Score, the algorithm MagFace50_FP16 specified in Section 5.2.1 was selected and implemented in OFIQ. The ONNX model was modified to output only the norm of the embedding (which is the native quality measure) to avoid issues with regulations on software that can be used for identification.

For the mapping of the native quality measure q to the uniform quality score Q target range [0,100], the function

$$Q = \text{ROUND}(100\ \text{SIGMOID}(q, 23, 2.6))$$

is used. Here and throughout the document, SIGMOID denotes the sigmoid function

$$\text{SIGMOID}(x, x_0, w) = \frac{1}{1 + e^{\frac{x_0 - x}{w}}}$$

with location parameter x_o and width w, and ROUND is defined as

- $\text{ROUND}(x) = \lfloor x + {}^1/_2 \rfloor$ for real non-negative numbers x
- $\text{ROUND}(x) = \lceil x - {}^1/_2 \rceil$ for negative numbers x

:

6 Capture-related Quality Components

Capture-related quality components mainly depend on the systems and environment used for image acquisition, e.g. on the capture device, on the lighting and the background during image capture, the distance between capture device and subject, and on the processing of the captured image by subsequent systems.

6.1 Background Uniformity

6.1.1 Data Selection

A test set was compiled from the FRGCv2 database [34] by selecting face images with uniform and non-uniform background. A total number of 24,614 face images with a uniform background and 14,713 with a non-uniform background have been chosen. Example images are shown in Figure 20.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for the evaluation of the Unified Quality Score (see Section 5.1).

Figure 20: Examples of face images with uniform (left) and non-uniform (right) background.

6.1.2 Selection and Prototyping of Candidate Algorithms

6.1.2.1 Luminance Entropy Algorithm

The first candidate algorithm for background uniformity is based on the algorithm from ISO/IEC WD5 29794-5:2022 [4], which measures the entropy of the luminance on the background of the face image. In comparison to the algorithm from [4], several improvements were implemented:

- The background is segmented via face parsing (cf. Section 4.3.3.2) instead of using landmarks (as in [4]). The background segmentation derived from the face parsing map is eroded to exclude the regions of the border between fore- and background.
- Image regions that have been padded during alignment (and are, thus, of black colour) are excluded from the background mask.
- Since background uniformity is mainly relevant for reference images for ID documents, the image and the segmentation map are cropped to meet the geometric face portrait requirements defined in ISO/IEC 39794-5:2019.
- Since parts of the clothing are sometimes incorrectly labelled as background, the image and the segmentation map are further cropped from the bottom to remove the region below the mouth.

The resulting algorithm takes as input the aligned image I and the transformation matrix T output by the algorithm in Section 4.4.1 and the face parsing segmentation map S (having the same dimensions as I) output by the algorithm in Section 4.3.2.1, and performs the following steps:

1. Create a completely black (0) grey scale image A of dimensions (w,h).

2. Apply the transformation T to A and crop and/or pad the image to dimension 616x616 to obtain a padding mask P. For padding use white colour (255).
3. Crop both I and P by 62 pixels from both sides and by 210 pixels from the bottom
4. Resize both I and P to size (354,292).
5. Crop the segmentation map S by 23 pixels from both sides and by 108 pixels from the bottom.
6. Compute the background mask B with B_{ij}=1, if S_{ij}=0 and P_{ij}=0, and B_{ij}=0 otherwise.
7. Apply to B the OpenCV function erode() with kernel size 4.
8. If $B_{i,j} = 0$ for all i and j, output an error Code (Failure to Assess).
9. Compute the luminance image L for image I as specified in the algorithm in Section 6.3.2.
10. Compute the normalised histogram $h_0, \cdots, h_{255}$ of the luminance values in the background B, where h_{i} denotes the frequency of the luminance value i in the background region $B(i,j)$=1.
11. Compute the entropy E of H, i.e. $E = \sum_{i=0}^{255} -(h_i + \varepsilon) \cdot \log_2(h_i + \varepsilon)$ where $\varepsilon = 10^{-7}$ is chosen in the implementation to avoid taking logarithms of zero values.
12. Output E as the native score for the background uniformity measure.

An example of the background segmentation is shown in Figure 21.

Figure 21: Example of the background segmentation used in the Luminance Entropy algorithm.

6.1.2.2 Luminance Gradient Algorithms

Furthermore, an algorithm based on the mean squared lengths of the luminance gradients was implemented. The squaring of the gradients is supposed to give more emphasis on higher values.

As an option, the background image resulting from applying the background segmentation mask to the accordingly resized image is scaled down by a factor 5 to increase gradients of softly blended regions of different colour. This option results in an additional variant of the algorithm which was coined *Luminance Gradient with Post-Scaling* algorithm.

The segmentation of the background was performed as in the Luminance Entropy algorithms (see Section 6.1.2.1 and Figure 21.

The resulting algorithms take as input the aligned image I and the transformation matrix T output by in Section 4.4.1, the face parsing segmentation map S (having the same dimensions as I) output by the algorithm in Section 4.3.2.1, and the input dimensions (w,h) of the original image (prior to alignment), and performs the following steps:

1. Crop and scale I and compute a corresponding background mask B as in the Gradient Entropy algorithm (Section 6.1.2.1).
2. If $B_{i,j} = 0$ for all i and j, output an error Code (Failure to Assess).
3. Only for *Luminance Gradient with Post-Scaling*: Resize L to width 98 and height 81 using Nearest Neighbour interpolation.
4. Compute the luminance image L for image I as specified in the algorithm in Section 6.3.2.
5. Apply horizontal and vertical Sobel filters with kernel size 3, S_3^{x} and S_3^{y} respectively, to L.
6. Compute ~~the array G of the~~ magnitudes of the gradients of L as $G_{i,j} = \sqrt{S_3^{\mathrm{x}}(L)_{i,j}{}^2 + S_3^{\mathrm{y}}(L)_{i,j}{}^2}$.
7. Compute the mean length of the gradients on the background, e.g.
8. $m = \left(\sum_{(i.j):B_{i,j}=1} G_{i,j}\right) / \sum_{(i.j)} B_{i,j}$
9. Output m as the native score for the background uniformity measure.

6.1.3 Evaluation

DET curves of the implemented algorithms on the test set FRGC are plotted in Figure 22 using the SSD face detector with ADNet landmark detection for alignment. The best detection performance, with EERs of 2.7%-2.8%, is achieved by the algorithms using luminance gradients (Luminance Gradient and Luminance Gradient with Post-Scaling).

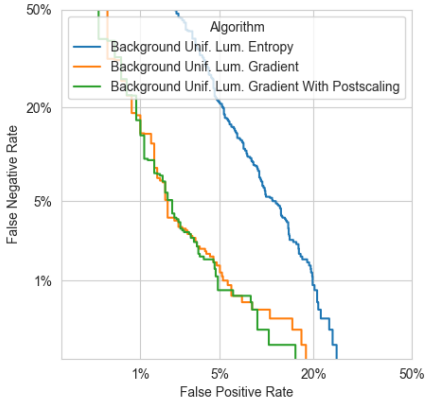

Figure 22: DET curve for the quality component Background Uniformity on the FRGC test set.

The corresponding EDC curves are plotted in Figure 23 and Figure 24. Interestingly, the Luminance Entropy algorithm shows a slight reduction in FNMR values for discard rates of up to 40% for all face recognition algorithms. In contrast, the algorithms using luminance gradients, which obtained the best EERs, only achieve a reduction in FNMR values for the commercial algorithms.

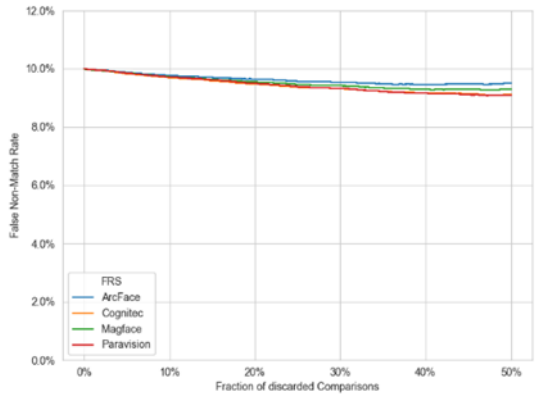

Figure 23: EDC curves for the quality component Background Uniformity using the Luminance Entropy algorithm on the VGGFace2 test set.

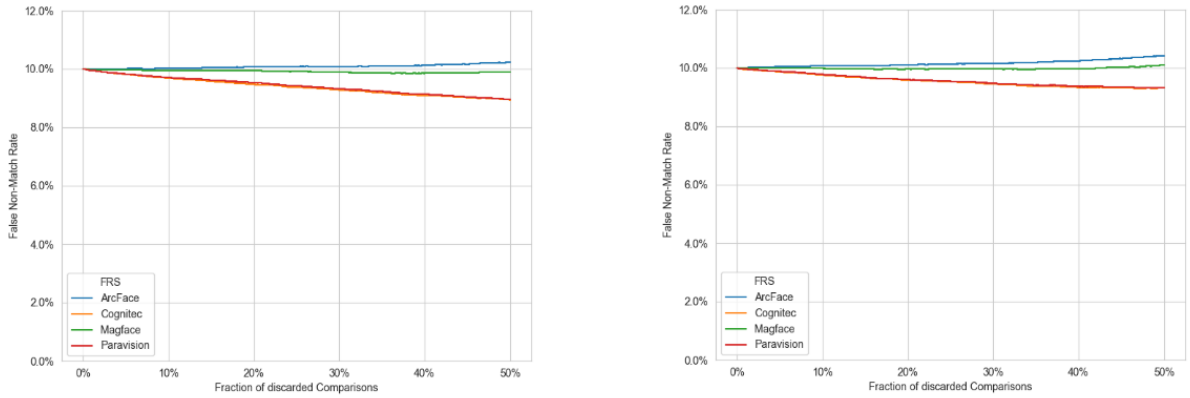

Figure 24: EDC curves for the quality component Background Uniformity using the algorithms Luminance Gradient (left) and Luminance Gradient with Post-Scaling (right) on the VGGFace2 test set.

The algorithms Luminance Gradient and Luminance Gradient with Post-Scaling were included in the submissions secunet_003 and secunet_004, respectively, to the NIST FATE Quality SIDD track [16]. There, Luminance Gradient showed a higher correlation with the ground truth than Luminance Gradient with Post-Scaling and Luminance Entropy (submission secunet_002), albeit its correlation was clearly lower than those of several commercial algorithms (see Figure 25).

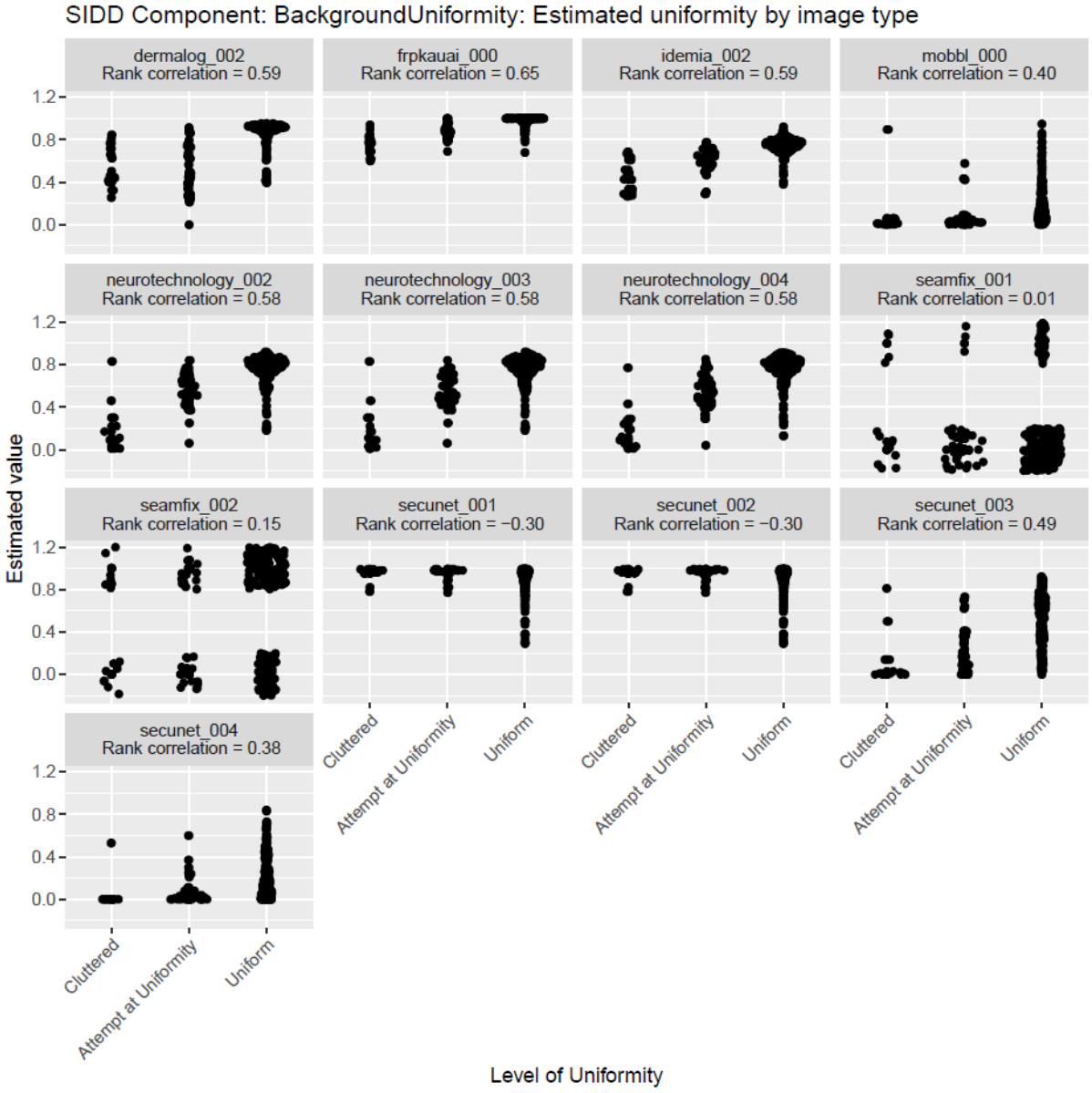

Figure 25: Distribution of the outputs for the SIDD component Background Uniformity per type of background in NIST FATE Quality SIDD report [16]. The plots for the algorithms Luminance Gradient and Luminance Gradient with Post-Scaling are labelled as secunet_003 and secunet_004, respectively.

6.1.4 Final Algorithm Selection

For the quality component Background Uniformity, the algorithm Luminance Gradient (without post-scaling) specified in Section 6.1.2.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \min(100, \max(0, \text{ROUND}(190(1 - \text{SIGMOID}(q, 10, 100)))))$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

6.2 Illumination Uniformity

6.2.1 Data Selection

Two test sets were compiled for this quality component:

1. From the ARFace database [35], 255 face images with uniform illumination (indexes 1 and 14) and 510 face images with non-uniform illumination (indexes 5,6,18, and 19) were selected. The latter class of images consist of 255 images captured each with light sources places on the left and right side of the face. Example images are shown in Figure 26.

Figure 26: Example images of the ARFace test set for Illumination Uniformity.

2. From the CMU Multi-PIE database [36] all frontal images with the value “16” at the end of the file name were selected as non-uniform illumination images (168 images), whereas 28 images are used for uniform-illumination (indexes 01-03,11-13) at the end of the file name.

Figure 27: Example images of the CMU Multi-PIE test set for Illumination Uniformity.

Table 6: Number of images in the selected test sets containing illumination variations.

Test Sets	Non-uniform illumination	Uniform illumination
ARFace	510	255
CMU Multi-PIE	168	28

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.2.2 Selection and Prototyping of Candidate Algorithms

6.2.2.1 WD1 Algorithm

The first candidate algorithm to estimate illumination uniformity was based on the algorithm defined in ISO/IEC WD1 29794-5:2020 [2]. Since, the specification in [2] does not define how the face region is segmented and split in a left and right side, the landmarked region segmentation (see Figure 10) was used, and the split was achieved using a vertical line defined by landmarks.

The algorithm takes as input the aligned image *I'* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1, and performs the following steps:

1. Convert *I'* to grey scale (single colour channel) using the OpenCV function cvtColor().
2. Using the landmarks *L'*, compute the landmarked region segmentation mask *R* using the algorithm specified in Section 4.3.1.1 with parameter α=0.
3. Split the landmarked region *R* in a left side R_L and a right side R_R as follows:
   a. Compute the left and right eye centres, (X_L, Y_L) and (X_R, Y_R), respectively, as in the algorithm in Section 7.7.2.
   b. Compute the eyes' midpoint M_e as the mean (centre of gravity) of the left and right eye centres, i.e. as $M_e = \big((X_L+X_R)/2, (Y_L+Y_R)/2\big)$.
   c. Compute the line *L* through M_e that is orthogonal to the line through (X_L, Y_L) and (X_R, Y_R).
   d. Set R_R to the part of *R* on the right side of *L*, and R_L to the remaining part of *R*.
4. Compute the mean μ_L and standard deviation σ_L of the pixel intensity values of *I'* in the left side R_L of the landmarked region.
5. Compute the mean μ_R and standard deviation σ_R of the pixel intensity values of *I'* in the right side R_R of the landmarked region.
6. Compute the aggregated standard deviation $\sigma = \sqrt{\frac{{\sigma_L}^2}{N_L} + \frac{{\sigma_R}^2}{N_R}}$, where N_L and N_R are the number of pixels in the left and right side of the landmarked region, respectively.
7. Compute and output the non-uniformity measure as $D = (\mu_L + \mu_R)/\sigma$.

The implementation of the algorithm was adapted from an implementation of [37].[23]

6.2.2.2 WD4 Algorithm

The second candidate algorithm to estimate illumination uniformity was based on the algorithm defined in ISO/IEC WD4 29794-5:2022 [3], which measures the intersection of the luminance histograms on the left and right side of the face. The algorithm was improved by using the landmarked region segmentation (see Figure 10) instead of the face bounding box and by taking the aligned face as input.

The algorithm takes as input the aligned image *I'* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1, and performs the following steps:

1. Compute the luminance of the image *I'* as in the algorithm in Section 6.3.2.
2. Using the landmarks *L'*, compute the landmarked region segmentation mask *R* using the algorithm specified in Section 4.3.1.1 with parameter α=0.
3. Using the landmarks of the eyes, split the landmarked region *R* in a left side R_L and a right side R_R as in the algorithm in Section 6.2.2.1.

[23] Taken from https://share.nbl.nislab.no/g03-03-sample-quality/face-image-quality.

4. Compute the normalised histogram $\left(h_0^{\mathrm{L}}, \ldots, h_{255}^{\mathrm{L}}\right)$ of the luminance values in the left side R_{L} of the landmarked region, where h_i^{L} is the proportion of pixels in R_{L} with luminance value *i*.
5. Compute the normalised histogram $\left(h_0^{\mathrm{R}}, \ldots, h_{255}^{\mathrm{R}}\right)$ of the luminance values in the right side R_{R} of the landmarked region, where h_i^{R} is the proportion of pixels in R_{R} with luminance value *i*.
6. Compute and output the intersection of the normalised histograms as

$$D = \sum_{i=0}^{255} \min\left(h_i^{\mathrm{L}}, h_i^{\mathrm{R}}\right)$$

6.2.2.3 WD5 Algorithm

In addition, the algorithm specified in ISO/IEC WD5 29794-5:2023 [4] (which is still contained in ISO/IEC DIS 29794-5:2024 [9]) also was implemented, which considers the luminance only in measurement zones *L'* and *R'* according to ISO/IEC 39794-5:2019 [1], see Figure 28.

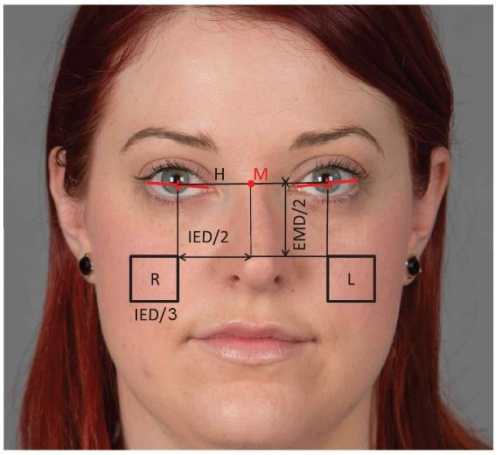

Figure 28: Definition of the measurement zones L' *and* R' *in ISO/IEC 39794-5:2019*

The algorithm takes as input the aligned image *I'* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1, and performs the following steps:

1. Compute the luminance of the image *I'* as in the algorithm in Section 6.3.2.
2. Using the landmarks *L'*, compute the left and right measurement zones, Z_{L} and Z_{R}, respectively, as in the algorithm in Section 6.8.2.1
3. Compute the normalised histogram $\left(h_0^{\mathrm{L}}, \ldots, h_{255}^{\mathrm{L}}\right)$ of the luminance values in the left measurement zone *L'*, where h_i^{L} is the proportion of pixels in Z_{L} with luminance value *i*.
4. Compute the normalised histogram $\left(h_0^{\mathrm{R}}, \ldots, h_{255}^{\mathrm{R}}\right)$ of the luminance values in the right measurement zone *R'*, where h_i^{R} is the proportion of pixels in Z_{R} with luminance value *i.*
5. Compute and output the intersection of the normalised histograms as

$$D = \sum_{i=0}^{255} \min\left(h_i^{\mathrm{L}}, h_i^{\mathrm{R}}\right)$$

6.2.3 Evaluation

The DET curves of the algorithms on the ARFace test set and the CMU Multi-PIE test set are shown in Figure 29. On ARFace, the WD5 algorithm (using the measurement zones) is much more accurate than the fixed WD4 algorithm (using the left and right half of the landmarked region). On Multi-PIE, the fixed WD4 algorithm performs slightly better than the WD5 algorithm. The WD1 algorithm shows a very poor detection performance.

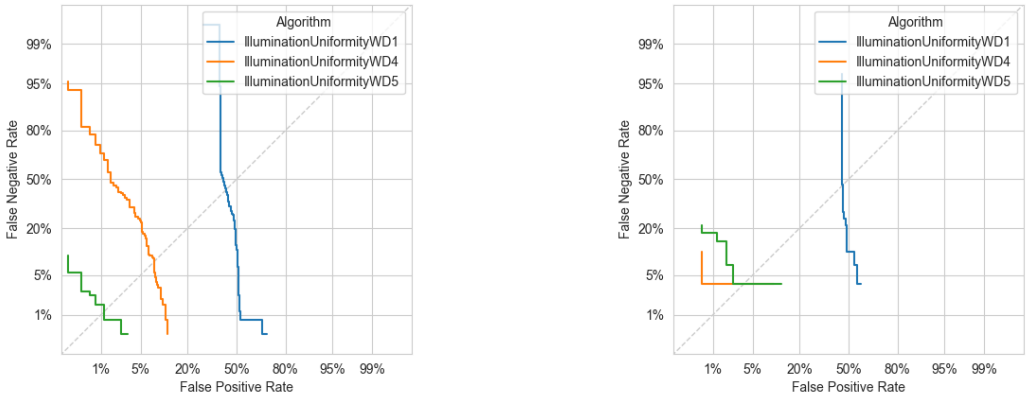

Figure 29: DET curves for the quality component Illumination Uniformity with SSD face detector and ADNet landmark detection on the ARFace (left) and CMU Multi-PIE (right) test sets.

The EDC curves for the tested algorithms on the VGGFace2 test set are shown in Figure 31, Figure 32 and Figure 30. Similar to the detection results, best performance is obtained for the WD5 algorithm. However, the reduction in terms of FNMRs is only moderate, e.g. approximately 0.5% for discarding 10% of images with non-uniform illumination using the WD5 algorithm.

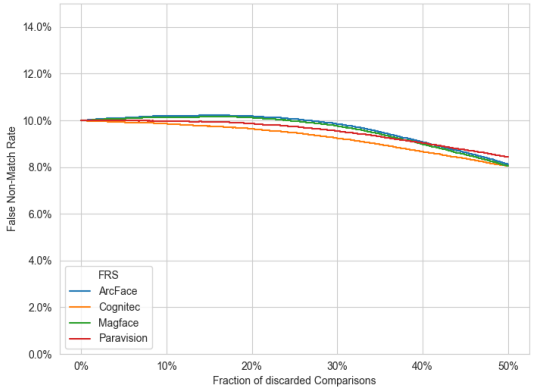

Figure 30: EDC curves for the quality component Illumination Uniformity using the fixed WD4 algorithm on the VGGFace2 test set.

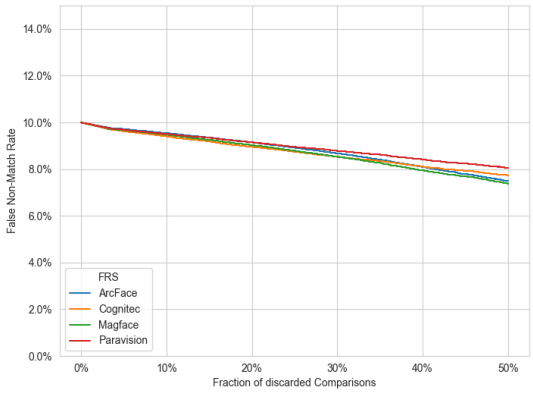

Figure 31: EDC curves for the quality component Illumination Uniformity using the WD5 algorithm on the VGGFace2 test set.

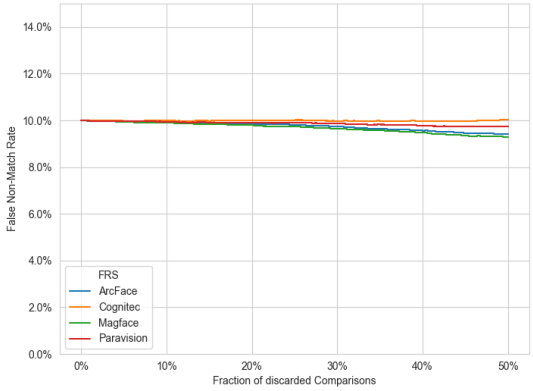

Figure 32: EDC curves for the quality component Illumination Uniformity using the WD1 algorithm on the VGGFace2 test set.

6.2.4 Final Algorithm Selection

For the quality component Illumination Uniformity, the WD5 algorithm specified in Section 6.2.2.3 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \mathrm{ROUND}(100\ q^{0.3})$$

is used, where ROUND is defined as described in Section 5.4.

6.3 Moments of Luminance Distribution

6.3.1 Data Selection

Since this quality component computes an exact quality measure, no evaluation with respect to ground truth is conducted.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.3.2 Selection and Prototyping of Candidate Algorithms

The algorithms defined in ISO/IEC CD3 29794-5:2023 [8] were implemented to compute the moments of the luminance distribution. The face region is segmented as the landmarked region (see Figure 10). Since the computations share the first steps, they are presented here in one common algorithm.

The algorithm takes as input the aligned image *I'* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1.

1. Using the landmarks *L'*, compute the landmarked region segmentation mask *R* using the algorithm specified in Section 4.3.1.1 with parameter α=0.
2. Compute the luminance from the RGB data of *I'* by the following procedure:
   a. Normalise and do gamma inversion of the (*RGB*) values using

      R_L = ColourConvert($R/255$), G_L = ColourConvert($G/255$), B_L = ColourConvert(B/255)

      where

      $$\text{ColourConvert}(V) = \begin{cases} ((V + 0.055)/1.055)^{2.4} & \text{if } V > 0.04045 \\ V/12.92 & \text{if } V \geq 0.04045 \end{cases}$$

   b. Compute a normalised luminance as

      $$Y = 0.2126R_L + 0.7152G_L + 0.0722B_L$$

   c. Scale the normalised luminance to an integer in the integer interval [0,255] by setting

      $$Y' = \lfloor 256 \cdot Y \rfloor$$

3. Compute the normalised luminance histogram $h_0, \cdots, h_{255}$ in the landmarked region *R,* where h_i is the proportion of pixels in *R* with luminance value *i*.
4. Compute and output the Luminance Mean as

   $$\bar{h} = \sum_{i=0}^{255} h_i \frac{i}{255}$$

5. Compute and output the Luminance Variance as

   $$VAR = \sum_{i=0}^{255} h_i \left| \frac{i}{255} - \bar{h} \right|^2$$

6. Compute the sample standard deviation σ of the luminance histogram as $\sigma = \sqrt{VAR}$.
7. Compute and output the Luminance Skewness as

   $$SKEW = \sum_{i=0}^{255} h_i \frac{\left| \frac{i}{255} - \bar{h} \right|^3}{\sigma^3},$$

8. Compute and output the Luminance Kurtosis as

$$KURT = \sum_{i=0}^{255} h_i \frac{\left|\frac{i}{255} - \bar{h}\right|^4}{\sigma^4}$$

The native quality measure values output by the above algorithm do not have a higher-is-better semantic and, except for Skewness, not even a lower-is-better semantic. Therefore, the native values are mapped to the corresponding quality component (QC) values, which are integers in the range [0,100], using the following mapping function, where SIGMOID is defined as described in Section 5.4.

$$QC_{\text{Brightness}} = \left\lceil 100 \cdot \text{SIGMOID}\left(\bar{h}, 0.2, 0.05\right)\left(1 - \text{SIGMOID}\left(\bar{h}, 0.8, 0.05\right)\right)\right\rceil$$

$$QC_{\text{Variance}} = \left\lceil 100 \cdot \sin\left(\pi \cdot \frac{60 \cdot VAR}{60 \cdot VAR + 1}\right) + 0.5\right\rceil$$

$$QC_{\text{Skewness}} = \left\lceil 100 \cdot \left(1 - \text{SIGMOID}(SKEW, 2, 0.35)\right) + 0.5\right\rceil$$

$$QC_{\text{Kurtosis}} = \left\lceil 100 \cdot \text{SIGMOID}(KURT, 2, 0.5)\left(1 - \text{SIGMOID}(KURT, 10, 2)\right) + 0.5\right\rceil$$

These scalar values have a higher-is-better semantic and were used for evaluation using EDC curves.

6.3.3 Evaluation

The EDC curves on the VGGFace2 data set are plotted in Figure 33 and Figure 34. All four quality components show a slight reduction of FNMR for all face recognition algorithms. The least reduction is observed for the Luminance Kurtosis, while the strongest reduction is observed for the Luminance Variance and Luminance Mean. The drop of the FNMR is also quite steady, which shows that the revised definitions of the functions mapping the native quality value to the final value are reasonable.

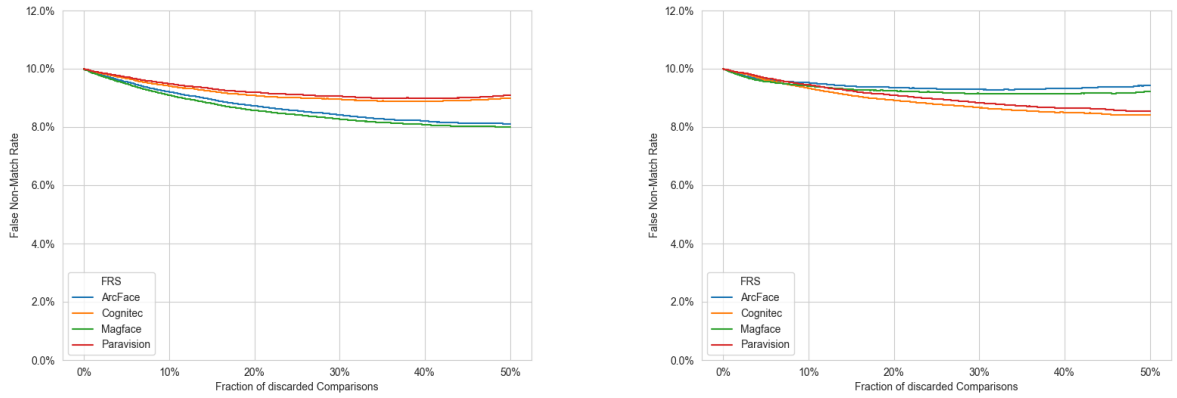

Figure 34: EDC curves of the quality components Luminance Skewness (left) and Luminance Kurtosis (right) on the VGGFace2 test set.

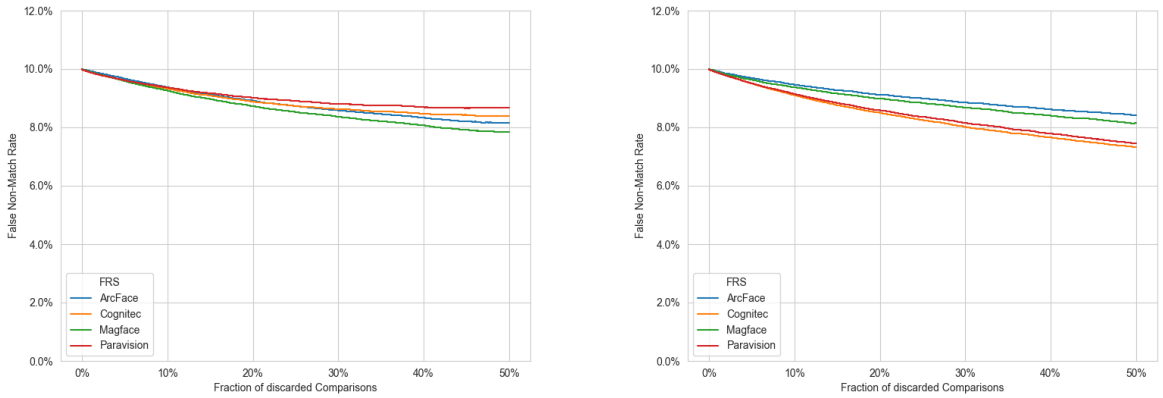

Figure 33: EDC curves of the quality components Luminance Mean (brightness, left) and Luminance Variance (right) on the VGGFace2 test set.

6.3.4 Final Algorithm Selection

For the quality components Luminance Mean and Luminance Variance, the algorithm specified in Section 6.3.2 was selected and implemented in OFIQ. The quality components Luminance Skewness and Luminance Kurtosis were discarded in ISO/IEC DIS 29794-5:2024 [9] and in OFIQ due to their limited impact on recognition performance.

For the mapping of the native quality measure $\bar{h}$ and VAR to the corresponding quality component values Q in the target range [0,100], the functions specified in Section 6.3.2 are used.

6.4 Over- and Under-Exposure Prevention

6.4.1 Data Selection

The first tests set were compiled by selecting face images with normal exposure as well as under- and over-exposed images from the CAS-PEAL-R1 face database [38], see Figure 35.[24] It is worth noting that in this test set, some of the normal images are partially overexposed (in particular in the nose area). As a consequence, evaluation results on this test set might be slightly worsened.

Figure 35: Example images with variations in exposure, from left to right: normal, over- and under-exposed face image.

For the second test set, images with normal exposure (indexes 1 and 14) and over-exposed mages (indexes 7 and 20) were selected from the ARFace database [35]; however, many images labelled as over-exposed only show hot spots on the cheeks and some image with normal exposure show bright reflections on the lenses of spectacles (see Figure 36).

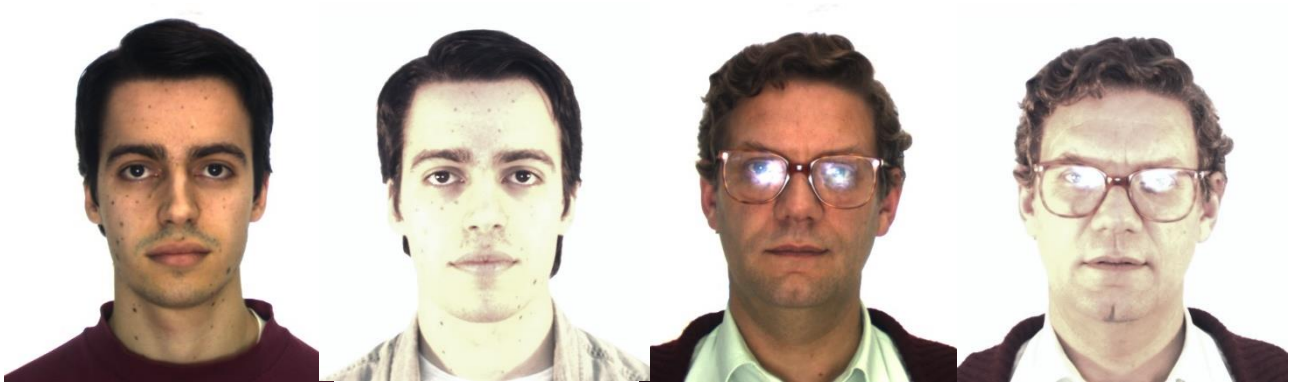

Figure 36: Images from ARFace with normal exposure (left), and over-exposure (middle and right).

Table 7 lists the number of normal, over- and under-exposed face images contained in each of the test sets.

Table 7: Number of images in compiled test sets containing exposure variations.

Test Set	Normal	Over-exposed	Under-exposed
CAS-PEAL-R1	1,040	28	98

[24] The differences in exposure result from different colours of the background, which was white in under-exposed images, blue in properly exposed images and dark in over-exposed images.

Test Set	Normal	Over-exposed	Under-exposed
ARFace	255	255	-

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.4.2 Selection and Prototyping of Candidate Algorithms

6.4.2.1 WD5 Algorithms

The first candidate algorithms for assessing over- and under-exposure are based on ISO/IEC WD5 29794-5:2022 [4]. The algorithms determine the proportion of pixels in the face region with luminance above (over-exposure) or below (under-exposure) threshold. The algorithms were improved by using the landmarked region segmentation (see Figure 10) instead of the face bounding box and by taking the aligned face as input.

The algorithms take as input the aligned image *I'* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1, and perform the following steps:

1. Using the landmarks L', compute the landmarked region segmentation mask R using the algorithm specified in Section 4.3.1.1 with parameter α=0.
2. Compute the luminance from the RGB data of I' as in the algorithm in Section 6.3.2.
3. Compute the normalised luminance histogram $h_0, \cdots, h_{255}$ in the landmarked region R, where h_i is the proportion of pixels in R with luminance value i.
4. Compute and output:
   a. Over-Exposure: The proportion of pixels for which luminance is on range [247;255] using
   $$v = \sum_{i=247}^{255} h_i$$
   b. Under-Exposure: The proportion of pixels for which luminance is on range [0;7] using
   $$v = \sum_{i=0}^{7} h_i$$

6.4.2.2 CD1 Algorithms

In ISO/IEC CD1 29794-5:2023 [6], the algorithms were improved by the following means:

- The face region is restricted to the points that are not farer from the eyes' midpoint than the nose tip (upper face region). This restriction aims to avoid that beards impair the accuracy of the assessment.
- The upper face region is further restricted to the region that is not labelled as occluded in the segmentation map computed by the face occlusion segmentation algorithm (Section 4.3.1.3). This restriction aims to prevent that hair in the face region, frames of glasses or other occlusions (e.g. face masks) impair the accuracy of the assessment.
- For under-exposure, the threshold for the luminance values is increased from 7 to 25.

The resulting candidate algorithms take as input the aligned image *I'* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1, the map *S* output by the face occlusion segmentation algorithm in Section 4.3.3.1, and perform the following steps:

1. Using the landmarks *L'*, compute the landmarked region segmentation mask *R* using the algorithm specified in Section 4.3.1.1 with parameter α=0.
2. Using the landmarks L', restrict *R* to the upper face region *U* as follows:
   a. Compute the left eye centre $(X_{\mathrm{L}}, Y_{\mathrm{L}})$ and the right eye centre $(X_{\mathrm{R}}, Y_{\mathrm{R}})$ as in the algorithm in Section 7.7.2.
   b. Compute the eyes' midpoint M_e as the mean (centre of gravity) of the left and right eye centres, i.e. as $M_{\mathrm{e}} = \big((X_{\mathrm{L}}+X_{\mathrm{R}})/2, (Y_{\mathrm{L}}+Y_{\mathrm{R}})/2\big)$.
   c. Compute the distance $D = \|M_{\mathrm{e}} - N\|$, with *N* being the landmark marking the nose tip:
      i. For dlib, $N = L'_{30}$
      ii. For mediapipe, $N = L'_{1}$
      iii. For ADNet, $N = L'_{54}$
   d. Set $U = \big\{\big((x,y) \in R \big| \|(x,y) - N\|_2 \leq D\big)\big\}$
3. Restrict *U* to its intersection with the region marked in *S* as un-occluded, i.e. set
$$X = U \cap S = \{(x,y) \in U | S(x,y) = 1\}$$
4. Compute the luminance from the RGB data of *I'* as in the algorithm in Section 6.3.2.
5. Compute the normalised luminance histogram $h_0, \cdots, h_{255}$ in the region *X,* where h_{i} is the proportion of pixels in *R* with luminance value *i*.
6. Compute and output:
   a. Over-Exposure: The proportion of pixels for which luminance is on range [247;255] using
$$v = \sum_{i=247}^{255} h_i$$
   b. Under-Exposure: The proportion of pixels for which luminance is on range [0;25] using
$$v = \sum_{i=0}^{25} h_i$$

An example of the segmentation steps performed by the CD1 algorithms is shown in Figure 37.

Figure 37: Example of the segmentation steps performed by the CD1 algorithms for Over- and Under-Exposure.

6.4.2.3 DIS Algorithms

In ISO/IEC DIS 29794-5:2024 [9], the algorithms were changed again by the following means:

- The face region is **not** restricted to the upper face region. Instead, the face region is restricted to the region that is not labelled as occluded in the segmentation map.

The resulting candidate algorithms take as input the aligned image *I'* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1, and perform the following steps:

1. Using the landmarks *L'*, compute the landmarked region segmentation mask *R* using the algorithm specified in Section 4.3.1.1 with parameter α=0.
2. Using the landmarks L', restrict *R* to the upper face region *U* as in the CD1 algorithm (Section 6.4.2.2).
3. Compute the luminance from the RGB data of *I'* as in the algorithm in Section 6.3.2.
4. Compute the normalised luminance histogram $h_0, \cdots, h_{255}$ in the region *U,* where h_i is the proportion of pixels in *R* with luminance value *i*.
5. Compute and output:
   a. Over-Exposure: The proportion of pixels for which luminance is on range [247;255] using

$$v = \sum_{i=247}^{255} h_i$$

   b. Under-Exposure: The proportion of pixels for which luminance is on range [0;25] using

$$v = \sum_{i=0}^{25} h_i$$

An example of the segmentation steps performed by the DIS algorithms is shown in Figure 38.

Figure 38: Example of the segmentation steps performed by the DIS algorithms for Over- and Under-Exposure

6.4.3 Evaluation

6.4.3.1 Over-Exposure

The DET curves for the algorithms for over-exposure are shown in Figure 39. On the ARFace test set, the CD1 algorithm assigns a raw score of 0 to more than 75% of the over-exposed images, resulting in a DET curve outside the plot's area. The DIS algorithm shows a considerably better performance as compared to the WD5 algorithm, which can be explained by bright reflections on spectacles, which are excluded in the DIS algorithm by applying the occlusion segmentation.

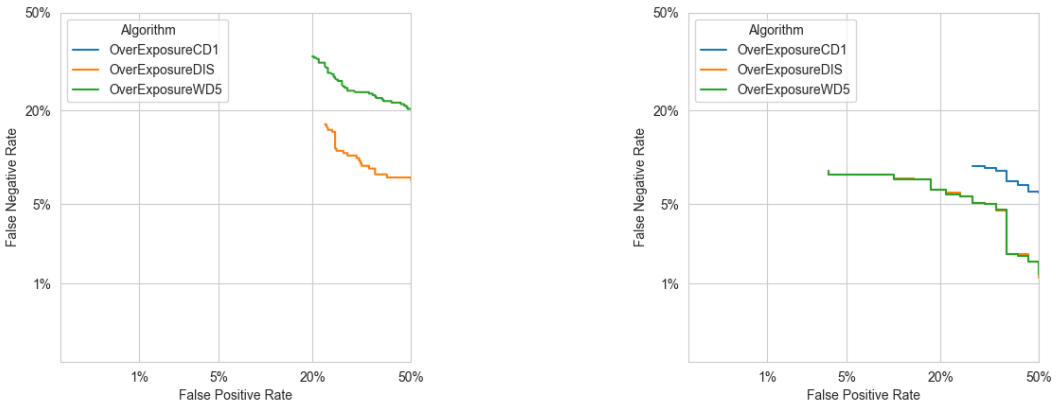

Figure 39: DET curves of the algorithms for the quality component Over-Exposure Prevention on ARFace test set (left) and the CAS-PEAL-R1 test set (right).

On CAS-PEAL-R1, the DET curves of the WD5 algorithm and the DIS algorithm are almost indistinguishable, due to the absence of significant occlusions in over-exposed images and reflections on spectacles. As on ARFace, the CD1 algorithm performs considerably worse. Obviously, the restriction to the upper face region in the CD1 algorithm doesn't work very well for over-exposure.

The corresponding EDC curves for the CD1 algorithm and the DIS algorithm for over-exposure are plotted in Figure 40. For both algorithms, only marginal reductions in terms of FNMR are achieved when discarding over-exposed face images assessed as over-exposed by these algorithms.

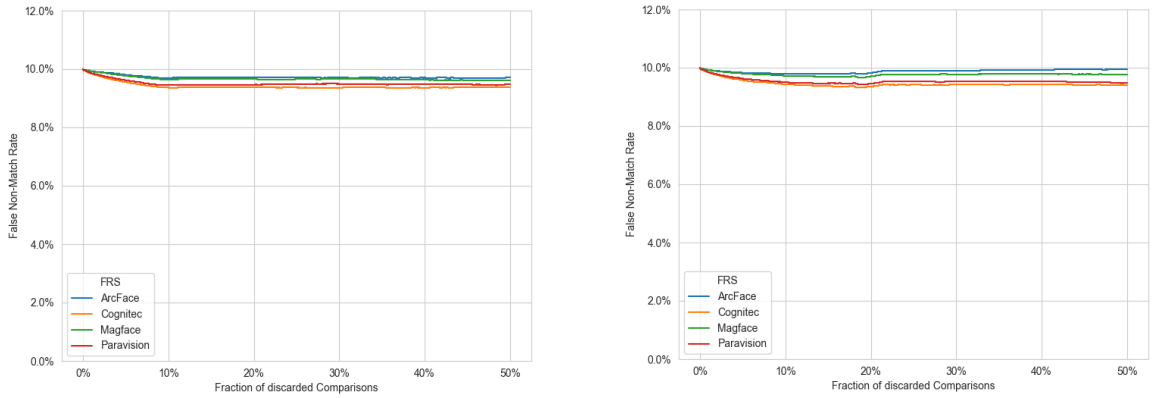

Figure 40: EDC curves for the quality component Over-Exposure Prevention using the CD1 algorithm (left) and the DIS algorithm (right).

6.4.3.2 Under-Exposure

All algorithms for under-exposure achieve a perfect separation on the CAS-PEAL-R1 test set. For the WD5 algorithm and the DIS algorithm, the scores are separated by a larger margin as compared to the CD1 algorithm, as visible in the histograms shown in Figure 41 and Figure 42.

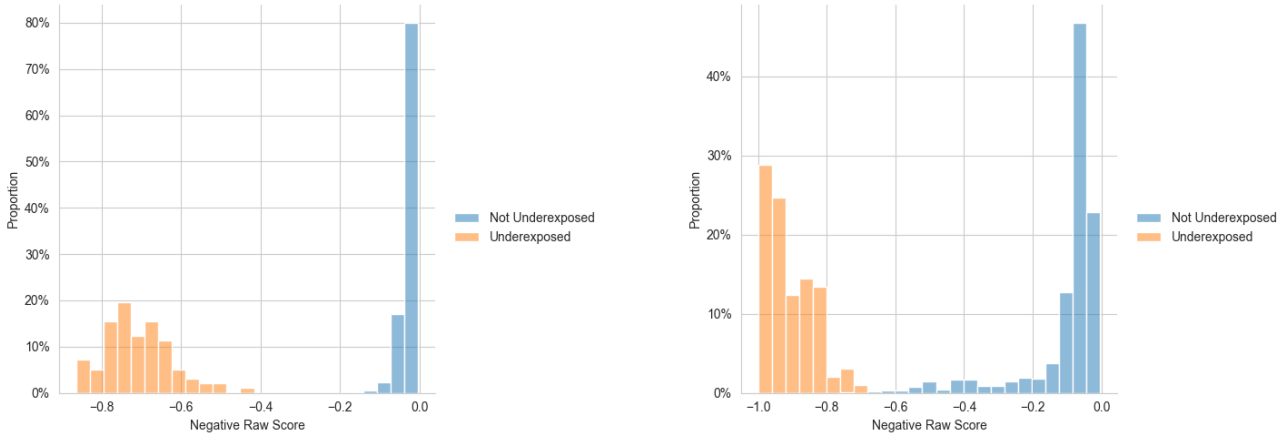

Figure 41: Histograms of the native scores of the WD5 algorithm (left) and the CD1 algorithm (right) for the the quality component Under-Exposure Prevention on the CAS-PEAL-R1 test set.

Thus, it can be concluded that the restriction of the segmentation to the upper face region is disadvantageous on the used test set. Since the purpose of this restriction was to avoid dark beards from being included in the histogram but the test set does not contain such beards, the utility of this restriction in cases of dark beards remains open. However, on images without dark beards (like those in the test set), the restriction seems to reduce the accuracy of the algorithm, thus, a better method to exclude dark beards (e.g., using a face parsing method that also labels beards) may be eligible for future improvements.

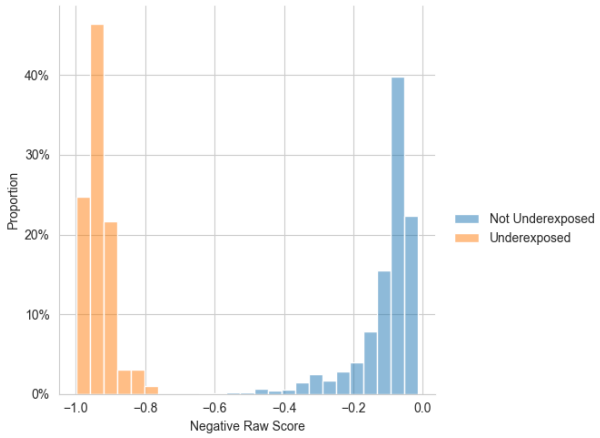

Figure 42: Histograms of the native scores of the DIS algorithm for the the quality component Under-Exposure Prevention on the CAS-PEAL-R1 test set.

The EDC curves for the CD1 algorithm and the DIS algorithm for under-exposure are shown in Figure 43. In contrast to the over-exposure quality component, noticeable reductions in terms of FNMR are obtained for discarding under-exposed face images. However, overall the FNMR is reduced by less than 1% for discarding 10% of images exhibiting the lowest quality according to this quality component.

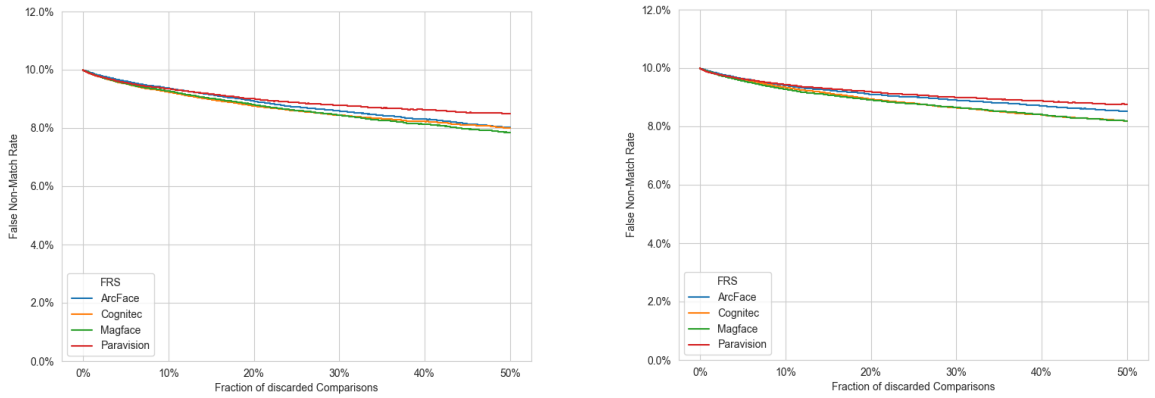

Figure 43: EDC curves for the quality component Under-Exposure Prevention using the CD1 algorithm (left) and the DIS algorithm (right).

The algorithms for over- and under-exposure were included in the submission secunet_005 to NIST FATE Quality SIDD track [16]. However, the submitted implementations of the CD1 algorithms were faulty.

As visible in Figure 44, the DIS algorithm for Over-Exposure achieves very high rank correlations with the ground truth values.

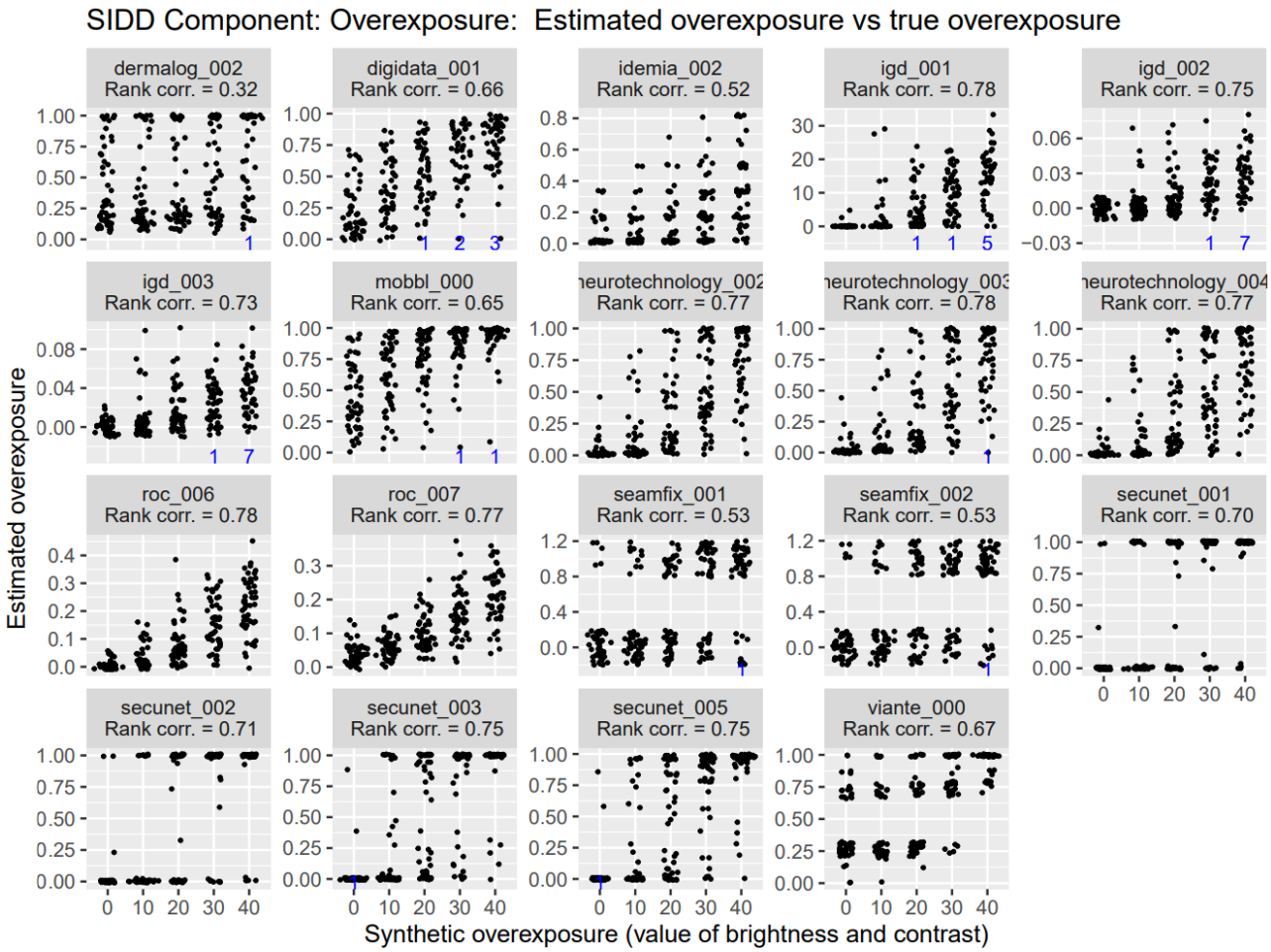

Figure 44: Distribution of the outputs for the SIDD component Over-Exposure per amount of over-exposure in NIST FATE Quality SIDD report [16]. The plot for the DIS algorithm is labelled as secunet_005, while secunet_003 denotes a faulty implementation of the CD1 algorithm.

In contrast, the DIS algorithm for Under-Exposure shows only a relatively low rank correlation with the ground truth data (see Figure 45). However, in the example images of the test set shown in the NIST FATE Quality SIDD report [16] (reproduced in Figure 46) the reduction of brightness and contrast by 16 leaves all textures perfectly visible so that, from the perspective of biometric recognition, the image with label (-16,-16) should not be considered as defective. Even in the example image with brightness and contrast reduced by -32, most textures are still visible, albeit with low contrast. Therefore, it depends on the application scenario, if a classification of such an image as “not under-exposed” is a serious issue. The DIS algorithm has not been designed to detect gradual reductions of the brightness.

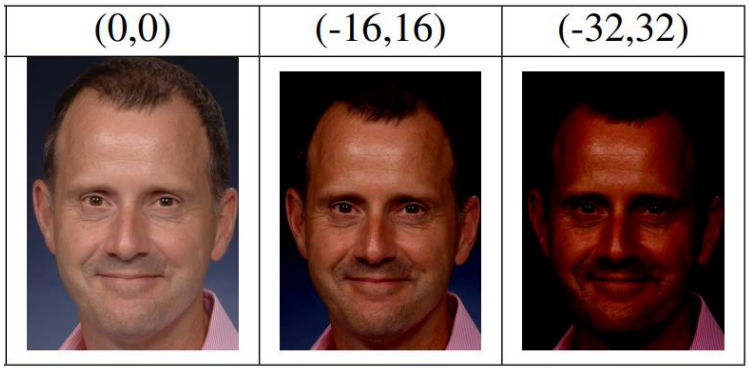

Figure 46: Example images of the NIST FATE Quality SIDD test set for under-exposure with the adjustments of brightness and contrast [16]].

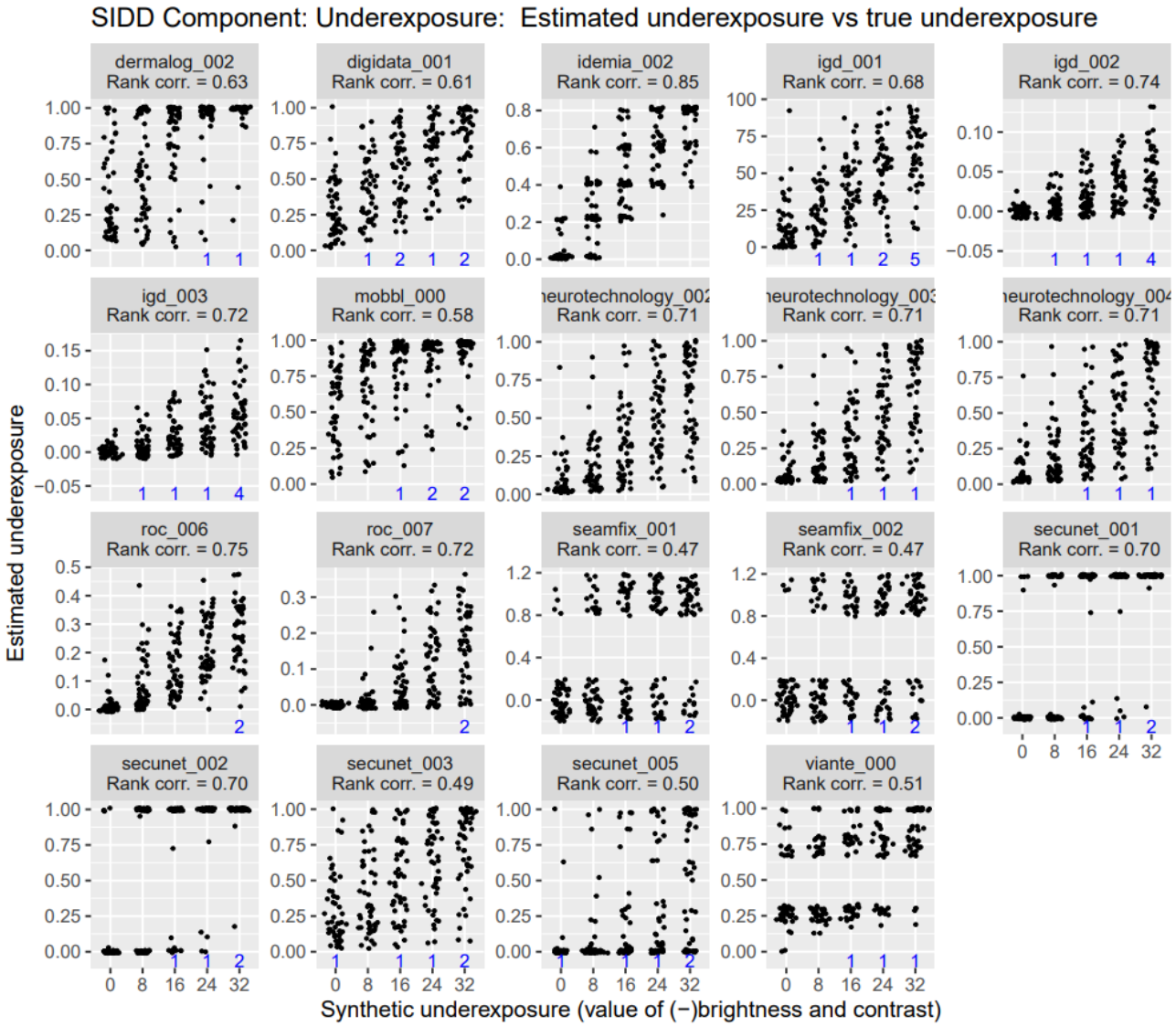

Figure 45: Distribution of the outputs for the SIDD component Under-Exposure per amount of under-exposure in the NIST FATE Quality SIDD report [16]. The plot for the DIS algorithm is labelled as secunet_005, while secunet_003 denotes a faulty implementation of the CD1 algorithm.

6.4.4 Final Algorithm Selection

For the quality components Over-Exposure Prevention and Under-Exposure Prevention, the DIS algorithms specified in Section 6.4.2.3 were selected and implemented in OFIQ.

For the mapping of the native quality measures q_{o} and q_{u} for over- and under-exposure, respectively, to the corresponding quality component values Q_{o} and Q_{u} in the target range [0,100], the functions

$$Q_{\mathrm{o}} = \mathrm{ROUND}(1/(q_{\mathrm{o}} + 0.01))$$

and

$$Q_{\mathrm{u}} = \min\left(100, \max\left(0, \mathrm{ROUND}\left(120, (0.832 - \mathrm{SIGMOID}(q_{\mathrm{o}}, 0.92, 0.05)\right)\right)\right)$$

are used, where ROUND and SIGMOID are defined as described in Section 5.4.

6.5 Dynamic Range

6.5.1 Data Selection

Since this quality component computes an exact metric, no evaluation with respect to ground truth is conducted.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.5.2 Selection and Prototyping of Candidate Algorithms

The algorithm defined in ISO/IEC CD3 29794-5:2023 [8] is used to estimate dynamic range.

The algorithms take as input the image I (un-aligned) and the landmarks L output by one of the algorithms in Section 4.2.1, and perform the following steps:

1. Using the landmarks L, compute the landmarked region segmentation mask R using the algorithm specified in Section 4.3.1.1 with parameter α=0.
2. Compute the luminance from the RGB data of I as in the algorithm in Section 6.3.2.
3. Compute the normalised luminance histogram $h_0, \cdots, h_{255}$ in the landmarked region R, where h_i is the proportion of pixels in R with luminance value i.
4. Compute and output the entropy H of the normalised luminance histogram is computed as

$$H = -\sum_{i=0, h_i \neq 0}^{M} h_i \log_2 h_i$$

6.5.3 Evaluation

EDC curves of the algorithm specified in Section 6.5.2 are depicted in Figure 47. A moderate drop in terms of FNMR can be observed.

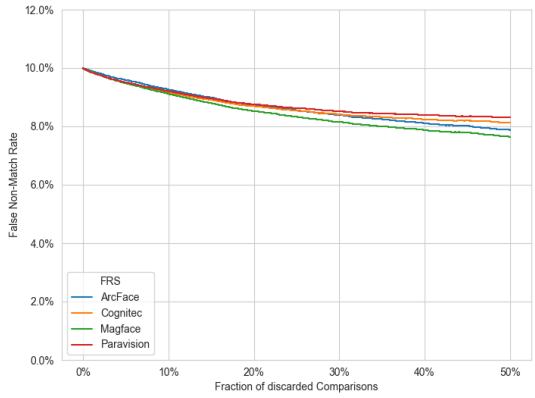

Figure 47:EDC curves for the quality component Dynamic Range using the algorithm based on ISO/IEC CD3 29794-5:2023 [8] with SSD face detector and ADNet landmark detection.

6.5.4 Final Algorithm Selection

For the quality component Dynamic Range, the algorithm specified in Section 6.5.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \min(100, \text{ROUND}(12.5\ q))$$

is used, where ROUND is defined as described in Section 5.4.

6.6 Sharpness

6.6.1 Data Selection

6.6.1.1 Training Set

For training of the machine learning-based algorithms (see Section 6.6.2.2), the following data set was compiled:

- 696 sharp and 254 blurry images chosen from the FRGCv2 database [34]
- 1,483 sharp and 1,313 blurry test images captured from volunteering employees at secunet using secunet's easykiosk and easygate systems.

This resulted in a training set of 2,179 sharp images and 1,567 blurry images that were assigned to the corresponding binary labels for classification.

6.6.1.2 Test Sets

The first test set was compiled by manually selecting 262 sharp and 280 out-of-focus face images from the FRGCv2 database [34] and setting binary class labels ("sharp", "blurred") accordingly. None of these images were included in the training set. Example images are shown in Figure 49.

Figure 49: Example images from the FRGC test set: sharp (left) and three unsharp.

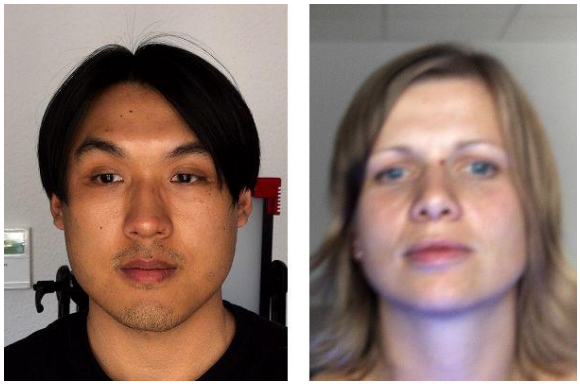

Figure 48: Example images (cropped to the facial region) of the internal test set: sharp (left) and out-of-focus (right).

For a second test set *internal*, 240 images were captured from volunteering employees at secunet using secunet's easykiosk and easygate systems, 116 images being out-of-focus and 124 images being sharp. None of these images were included in the training set. Example images of this internal test set are shown in Figure 48.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.6.2 Selection and Prototyping of Candidate Algorithms

All candidate algorithms are based on general image processing methods. In a pre-processing step, a face image is masked to the face region to ensure that only the inner facial region (without forehead) is assessed. Then the image is cropped to the minimum square bounding box of the face region and resized to 250x250 pixels. Finally, a greyscale conversion is applied. Examples of pre-processed face images are shown as part of Figure 50.

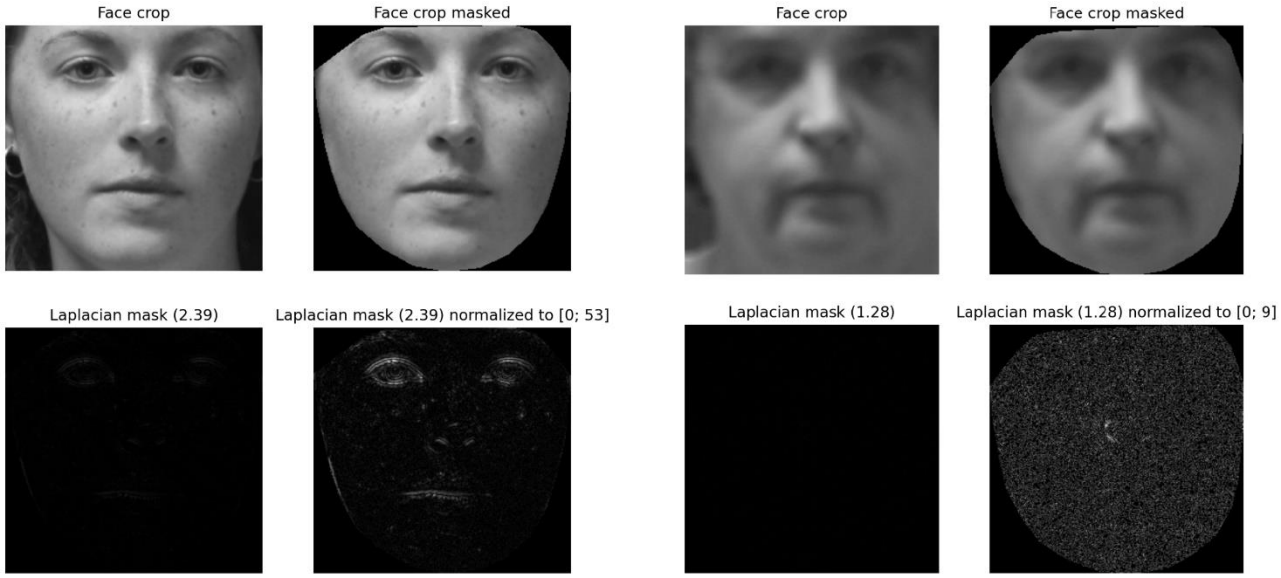

Figure 50: Example of pre-processing, the application of the Laplacian operator and resulting scores for a sharp (left) and an unsharp image (right).

6.6.2.1 Algorithms based on Single Features

For the detection of out-of-focus face images different features are applied:

1. **Laplacian:** First, a Gaussian blur filter G_3 with 3x3 kernel is applied to the image I to remove noise. Then, a Laplacian filter L_1 with kernel size 1 is applied using the OpenCV function Laplacian(). Examples are depicted in Figure 50. The mean of all absolute values on the face region R, i.e. $\mu\left(|L_1(G_3(I))| \,\middle|\, R_{i,j} = 1\right)$, is returned as score.
2. **Difference to mean-filtered image:** This method is defined in ISO/IEC WD5 29794-5:2022 [4]. A 3x3 mean filter M_3 (OpenCV function cv2.blur) is applied to the image and the resulting image is subtracted from the original one. The mean of this difference in the landmarked region R, i.e., $\mu\left(I_{i,j} - M_3(I)_{i,j} \,\middle|\, R_{i,j} = 1\right)$ is returned as score.

The resulting algorithm takes as input the image I (un-aligned) in BGR colour channel order with 8 bits per channel and the landmarks output by any of the landmark extraction algorithms in Section 4.2.1 . The following steps are executed:

1. Compute the landmarked region segmentation mask *R* using the algorithm specified in Section 4.3.1.1 with parameter α=0.
2. Crop *I* and *R* to the minimal upright bounding box of the landmarked region, i.e. of all pixels *i,j* with $R_{i,j}$=1.
3. Scale *I* and *R* so that the longer side is 250, i.e. by a factor q, where $q = 250/\max(w, h)$ and *w, h* are the width and height of the cropped image, respectively.
4. Convert *I* to grey scale.
5. Compute the score *x* on *I* restricted to *R* according to the method (see above), i.e. either
   - Laplacian or
   - Difference to mean-filtered image.

   There, the mean is computed on the landmarked region only.
6. Return the score *x*.

6.6.2.2 Machine Learning based Algorithms

In addition, an algorithm is implemented based on a classifier using the following features:

- **Sobel filters.** For each of the kernel sizes *s*=1,3,5,7,9, the isotropic Sobel filter S_s^{xy} is computed on the pre-processed image using the OpenCV function Sobel() with parameters dx=1, dy=1 and ksize=*s*. For each *s*, the arithmetic mean and standard deviation of the absolute values of the outputs in the landmarked region *R* are computed as feature, i.e. $\mu\left(\left|S_s^{\mathrm{xy}}(I_{i,j})\right| \,\middle|\, R_{i,j} = 1\right)$ and $\sigma\left(\left|S_s^{\mathrm{xy}}(I_{i,j})\right| \,\middle|\, R_{i,j} = 1\right)$ for *s*=1,3,5,7,9.
- **Laplacian filters.** First, a Gaussian blur filter G_3 with 3x3 kernel is applied to the pre-processed image to remove noise. Then, for each of the kernel sizes *s*=1,3,5,7,9, the Laplacian Filter L_s with kernel size *s* is computed on the blurred image using the OpenCV function Laplacian() with parameter ksize=*s*. For each *s*, the arithmetic mean and standard deviation of the absolute values of the outputs in the landmarked region *R* are computed as feature, i.e. $\mu\left(|L_s(G_3(I))| \,\middle|\, R_{i,j} = 1\right)$ and $\sigma\left(|L_s(G_3(I))| \,\middle|\, R_{i,j} = 1\right)$ for *s*=1,3,5,7,9.
- **Difference to mean-filtered image.** This feature is based on the method defined in ISO/IEC WD5 29794-5:2022 [3]. For each of the kernel sizes *s*=3,5,7 a mean filter M_s (OpenCV function cv2.blur) is applied to the image and the resulting image is subtracted from the original one. The mean and standard deviation of the absolute values of this difference in the landmarked region *R*, i.e., $\mu\left(\left|I_{i,j} - M_s(I)_{i,j}\right| \,\middle|\, R_{i,j} = 1\right)$ and $\sigma\left(\left|I_{i,j} - M_s(I)_{i,j}\right| \,\middle|\, R_{i,j} = 1\right)$, are returned as feature value.

This results in a features vector of length 26. These feature vectors were extracted for the training set described in Section 6.6.1.

Using OpenCV's machine learning module, three different classifiers were trained on 80% of the data (training split), and optimised with respect to the hyperparameters on 20% of the data (validation split):

- An SVM (class *ml::SVM* of OpenCV). As optimal hyperparameters, the SVM type *C-Support Vector classification*, a polynomial kernel with degree = 3.43, EPS-regression with C = 2.5 and gamma = 0.00225 were determined. TermCriteria was not set but left to the default (3, 1000, 1.19E-07).
- An AdaBoost classifier (class *ml::Boost* of OpenCV). As optimal hyperparameters, BoostType = 3 ("Gentle"), WeakCount = 100, WeightTrimRate = 0.9, MaxDepth = 25, and MinSampleCount = 12 were determined.
- A Random Forest (class *ml::RTree* of OpenCV). As optimal hyperparameters, MinSampleCount = 2, MaxDepth = 25, and TermCriteria = (3, 50, 0.1) were determined.

As evaluation metric, the accuracy was used. For each of the three classifiers, an algorithm was implemented.

The resulting algorithm takes as input the image *I* (un-aligned) in BGR colour channel order with 8 bits per channel and the landmarks output by any of the landmark extraction algorithms in Section 4.2.1 . For initialisation, the trained classifier model (SVM, AdaBoost or Random Forest, respectively) are loaded from file. In case of the SVM classifier, the scaling parameters for the features and the labels (computed on the training set) are also loaded. Then the following steps are executed:

1. Compute the landmarked region segmentation mask *R* using the algorithm specified in Section 4.3.1.1 with parameter α=0.
2. Crop *I* and *R* to the minimal upright bounding box of the landmarked region, i.e. of all pixels *i,j* with $R_{i,j}$=1.
3. Scale *I* and *R* so that the longer side is 250, i.e. by a factor *q*, where $q = 250/\max(w, h)$ and *w, h* are the width and height of the cropped image, respectively.
4. Convert *I* to grey scale.
5. Compute the features on *I* restricted to *R* as described above to obtain a 26-dimensional feature vector *f*; the mean and standard deviation of the filter outputs and mean difference values is computed on the landmarked region only.
6. If the SVM classifier is used, scale the feature vector f using the scaling parameters.
7. Feed the feature vector f into the classifier model (SVM, AdaBoost or Random Forest, respectively) to obtain a score *x*.
8. Return *x*.

6.6.3 Evaluation

The DET curves of all algorithms implemented on the FRGC test set and the internal test set are shown in Figure 51. On FRGC, the best error rates are achieved by the WD5 algorithm (using difference from the mean-filtered image) and the Laplacian filter, closely followed by the Random Forest Classifier algorithm (*ClassifierRTree*). Clearly, the AdaBoost Classifier algorithm (*ClassifierBoost*) has the highest error rates. On the internal test set, however, the trained classifiers (SVM, RandomForest and AdaBoost) perform better than the algorithms based on single features (Mean-Diff and Laplace). In contrast to the results on the FRGC test set, the SVM classifier gives the best accuracy.

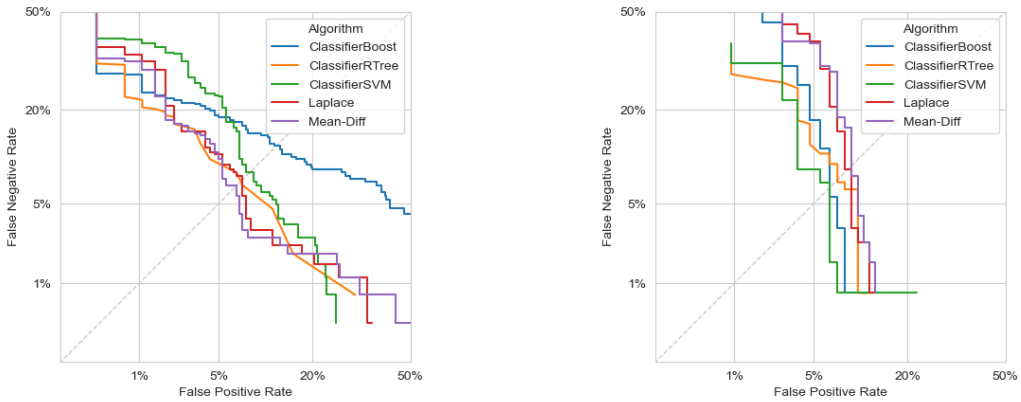

Figure 51: DET curves of the algorithms for the quality component Sharpness on the FRGC test set.

EDC curves of the implemented algorithms are plotted in Figure 52, Figure 53 and Figure 54. It can be observed that the algorithms based on machine learning classifiers (AdaBoost, Random Forest and SVM)

show only moderate to no reduction in FNMR across relevant discard rates. In contrast, the algorithms based on the single features, i.e., the difference from the mean-filtered image (WD5 algorithm) and the Laplacian filter, achieve a decrease in FNMR from 10% to about 8% for a discard rate of 10%.

However, it is not reasonable that this finding is caused by a better predictive accuracy w.r.t. image sharpness: The VGGFace2 data set only contains very few images where the lack of sharpness results in a visible lack of texture details in the images of dimensions 112x112 given as input to the open source face recognition algorithms;[25] consequently, a good predictive performance for image sharpness can result in a clear decline of the FNMR only for the first percentages discard rate. On the other hand, a visual inspection of the images obtaining very low scores by the WD5 algorithm and the algorithm based on the Laplacian filter shows that many of these are under-exposed or have a very low dynamic range.

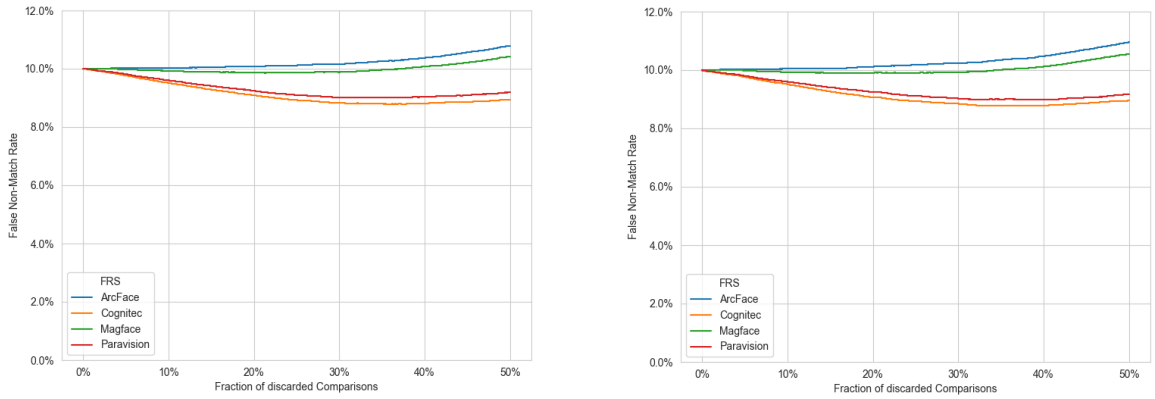

Figure 53: EDC curves for the quality component Sharpness using the AdaBoost Classifier algorithm (left) and the Random Forest Classifier algorithm (right) on the VGGFace2 test set.

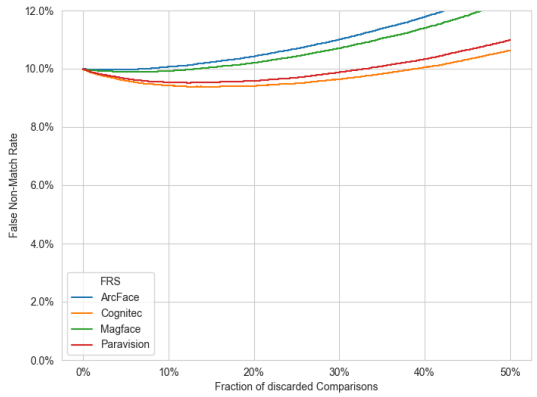

Figure 54: EDC curves for the quality component Sharpness using the SVM Classifier algorithm on the VGGFace2 test set.

[25] Both ArcFace and MagFace are used with the alignment and cropping of the algorithm in Section 3.2.

The algorithms based on Random Forest and AdaBoost were submitted to NIST FATE Quality SIDD track [16]. The results are illustrated in Figure 55. The test set contained face images with a large range of IED values synthetically blurred with different amounts of Gaussian blur. The results show a very high negative correlation (-0.82) of the outputs with ground truth values for the Random Forest Classifier algorithm. For the AdaBoost Classifier algorithm, the negative correlation is still reasonable high (-0.70), but smaller than the correlation for the algorithm based on the difference to the mean-filtered image (-0.78). However, since the test images were only synthetically blurred, these results should be taken with care.

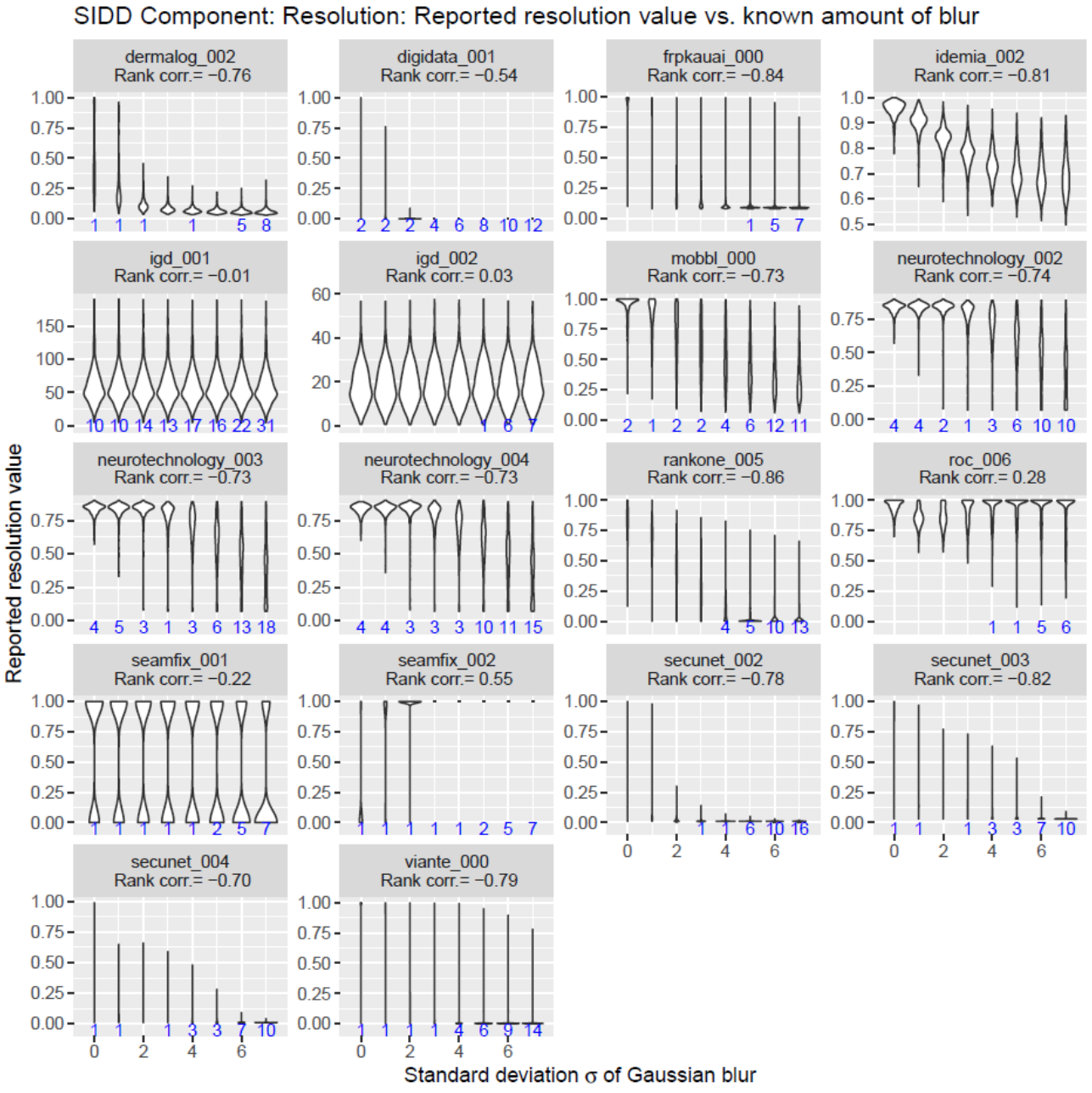

Figure 55: Distribution of the outputs (y-axis) per amount of Gaussian blur (x-axis) and rank correlation of the amount of blur with the outputs for the algorithms for the SIDD component Resolution in the NIST FATE Quality SIDD report [16]. The algorithm based on the difference to the mean-filtered image is labelled as secunet_002, the algorithms based on Random Forest and AdaBoost as secunet_003 and secunet_04, respectively.

6.6.4 Final Algorithm Selection

For the quality component Sharpness, the algorithm based on the Random Forest Classifier specified in Section 6.6.2.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(115\ \text{SIGMOID}(q, -20, 15) - 14)$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

6.7 No Compression Artefacts

6.7.1 Data Selection

6.7.1.1 Training Data

A first large training data set comprising rotated and upright compression artefacts of JPEG and JPEG2000 has been compiled based on 48,000 images from FERET [39] and FRGCv2 [34] as follows.

1. All 48,000 images are aligned as with the algorithm specified in Section 4.4.1, but with deviating parameters to ensure that the aligned image has a higher resolution. Precisely, in step 5, the target point array is set to P' = [[311, 337], [451, 337], [382, 416], [325, 498], [440, 498]] and, in step 7, the target image dimension is set to 763x763.
2. Each aligned image is scaled to an IED randomly chosen from the set [60, 70, …, 130, 140, 200].
3. With 50% probability, the scaled images are then rotated around the image centre by an angle randomly chosen from the integer interval [-8, 8].
4. All scaled and potentially rotated images are then compressed as follows:[26]
    - As JPEG using ImageMagick[27] with every even compression quality value in the integer range [20,64].
    - As JPEG2000 using ImageMagick with every odd compression quality value in the integer range [31, 65].
    - As JPEG2000 using IrfanView[28] with every even compression quality value in the integer range [20, 64].
5. For images that had been rotated before compression, the compressed images are rotated back by the negative angle around the image centre.
6. Finally, all images are scaled to dimensions 616x616 (as by the alignment algorithm in Section 4.4.1) and are then cropped from all sides by 184 pixels, resulting in images of dimensions 248x248 showing the inner face region.

The reason of limiting the compression quality value to 65 is that, for ImageMagick, images compressed with that value are already visually indistinguishable from uncompressed images. The purpose of scaling and rotation prior to compression is to ensure that the blocking artefacts visible in the images given as input to the algorithm specified in Section 4.4.1 (which are aligned, cropped to the face and scaled to fixed dimensions) are of varying sizes and not always parallel to the image axes.

This results in a training set of approximately 2.6 million images. As labels the difference between the image and the corresponding uncompressed image (i.e. the source image) are computed. The differences are measured by both Peak Signal to Noise Ratio (PSNR) and Structural Similarity (SSIM) [40], resulting in two sets of labels. The PSNR values were mapped to the range [0,1] by setting $p' = 1 - 1/p$, and then performing a min-max normalisation for all p'.

[26] The images from FRGCv2 are actually already shipped in JPEG format, but from the file sizes, we estimate that compression quality 98 or 99 had been used.
[27] https://imagemagick.org/index.php
[28] https://www.irfanview.de/

6.7.1.2 Test Data

Two tests sets were built from 400 sharp source images with good illumination from Flickr Faces HQ (FFHQ) [18]. These images are already aligned with an IED of approximately 260. These test sets were used for classification only.

For the first test set these source images were processed in the following way:

- Each image is scaled to an inter-eye-distance randomly chosen from the set {60,90,120,140}.
- With 50% probability, the scaled images are then rotated around the image centre by an angle randomly chosen from the integer interval [-15, 15].
- All scaled and potentially rotated images are then compressed using IrfanView as follows:
    - As JPEG with a random compression quality from the set {20,30,40,50,60,70}.
    - As JPEG2000 with a random compression quality from the set {20,30,40,50,60,70}.
- For images that had been rotated before compression, the compressed images are rotated back by the negative angle around the image centre.
- The uncompressed images are augmented by horizontal flipping

This results in a test set coined **Flickr-Rotated** consisting of 2.400 images, 800 being uncompressed and 1.600 being compressed. As labels, we assign “uncompressed” and “compressed”. This test set was split into two test sets **Flickr-Rotated JPG** and **Flickr-Rotated JP2** by restricting the compressed images correspondingly. Each of these sub sets contains 800 compressed images and 800 uncompressed images.

For a second test set the source images were processed in a similar way as for the first test set but without applying rotations:

- Each image is scaled to an inter-eye-distance randomly chosen from the set {60,90,120,140}.
- All scaled and potentially rotated images are then compressed using IrfanView as follows:
    - As JPEG with a random compression quality from the set {20,30,40,50,60,70}.
    - As JPEG2000 with a random compression quality from the set {20,30,40,50,60,70}.
- The uncompressed images are augmented by horizontal flipping

This results in a test set coined **Flickr-Upright** consisting of 2.400 images, 800 being uncompressed and 1.600 being compressed. As labels, we assign “uncompressed” and “compressed”. This test set was split into two sub sets **Flickr- Upright JPG** and **Flickr- Upright JP2** by restricting the compressed images correspondingly. Each of these sub sets contains 800 compressed images and 800 uncompressed images.

Figure 56: Sample images (cropped) from Flickr-Rotated: uncompressed (left), scaled to IED=120 and JPEG-compressed with quality 70 (middle), and scaled to IED=60 and JPEG2000-compressed with quality 60 (right).

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.7.2 Selection and Prototyping of Candidate Algorithms

6.7.2.1 Algorithms based on a single CNN (PSNR and SSIM)

Using the first training set described in Section 6.7.1.1, two different CNNs were trained to predict the PSNR and SSIM, respectively, of the input image to the uncompressed source image. The approach and design choices are described in detail in [41], but the training set differs from that used there. Using the models M_{PSNR} and M_{SSIM} trained on PSNR and SSIM, respectively, corresponding algorithms "PSNR" and "SSIM" were implemented that take as input an aligned image *I* output by the algorithm in Section 4.4.1 in RGB colour channel order with 8 bits per channel and perform the following steps:

1. Crop *I* by 184 pixels from all sides.
2. Normalise *I* with mean $\mu = (123.7, 116.3, 103.5)$ and standard deviation $\sigma = (58.4, 57.1, 57.4)$ to obtain array *A* by setting, for all i, j, k,
$$A_{i,j,k} = (I_{i,j,k} - \mu_k)/\sigma_k$$
3. Reshape A as input tensor T of dimensions (1, 3, 248, 248) for the neural network model.
4. Run a forward pass through the corresponding CNN model (M_{PSNR} or M_{SSIM}) using T as input to obtain a single output value tensor T' of dimension (1,1).
5. Output $T'_{1,1}$.

6.7.2.2 Algorithm using two CNNs

Furthermore, two additional CNN models M_{JPEG} and M_{JP2000} were trained using subsets of the training set described in Section 6.7.1.1 covering only one of the two compression algorithms: The model M_{JPEG} was trained on the JPEG compressed images and the uncompressed images, while M_{P2000} was trained on the JPEG2000 compressed images and the uncompressed images. Both models were trained to predict the PSNR. Apart from the training set, the approach and design choices were the same as for M_{PSNR} and in [41]. Using the models M_{JPEG} and M_{JP2000}, an algorithm "TwoCNNs" was implemented that takes as input an input image *I* in RGB colour channel order and performs the following steps:

1. Crop *I* by 184 pixels from all sides.
2. Normalise *I* with mean $\mu = (123.7, 116.3, 103.5)$ and standard deviation $\sigma = (58.4, 57.1, 57.4)$ to obtain array *A* by setting, for all i, j, k,
$$A_{i,j,k} = (I_{i,j,k} - \mu_k)/\sigma_k$$
3. Reshape *A* as input tensor T of dimensions (1, 3, 248, 248) for the neural network model.
4. Run a forward pass through the CNN model M_{JPEG} using T as input to obtain a single output value tensor T^1 of dimension (1,1).
5. Run a forward pass through the CNN model M_{JP2000} using T as input to obtain a single output value tensor T^2 of dimension (1,1).
6. Output $\min(0.46 \cdot T^1_{1,1}, T^2_{1,1})$.

The scaling parameter 0.46 in the last step is needed to ensure that the outputs of model M_{JPEG} are of the same magnitude as those of M_{P2000}; the parameter was determined by optimisation of the EER of a

classification on the ECVP test set used in [41], restricted to JPEG compressions below 97, to Image Magick JPEG2000 compressions between 36 and 44, and to IrfanView JPEG2000 compressions below 95.[29]

6.7.3 Evaluation

The DET curves of the implemented algorithms (using the SSD face detector with ADNet landmark estimation for alignment) for the subsets of Flickr-Rotated and Flickr-Upright are shown in Figure 57 and Figure 58. It can be observed that, on all 4 test sets, the PSNR CNN algorithm performs best and the SSIM CNN worst. The PSNR CNN algorithm achieves very a low error rates for JPEG-compressions (EER of 1.4% for Flickr-Rotated JPG and 0.9% for Flickr-Upright JPG), and higher error rates for JPEG2000 compressions (EER of 4.3% for Flickr-Rotated JPG and 2.1% for Flickr-Upright JP2). Nevertheless, the PSNR CNN algorithm outperforms the TwoCNNs algorithm on JPEG2000 images, too.

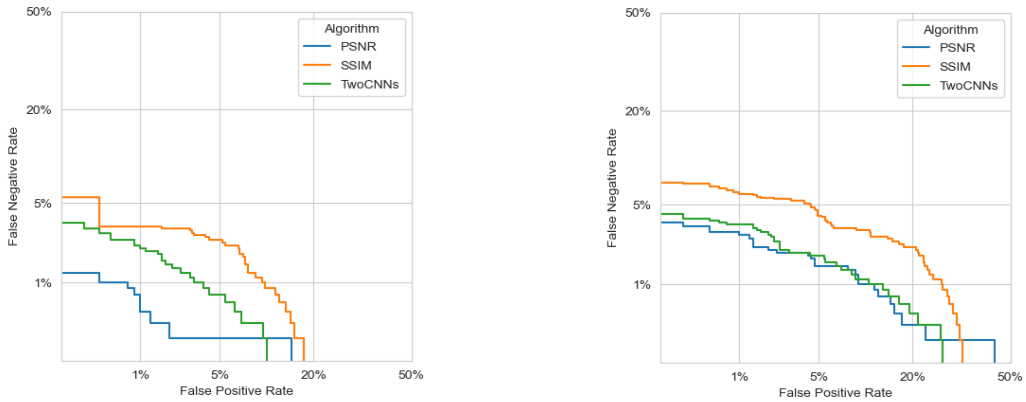

Figure 57: DET curves of the algorithms for the quality component No Compression Artefacts on Flick-Upright JPG (left) and Flick-Upright JP2 (right).

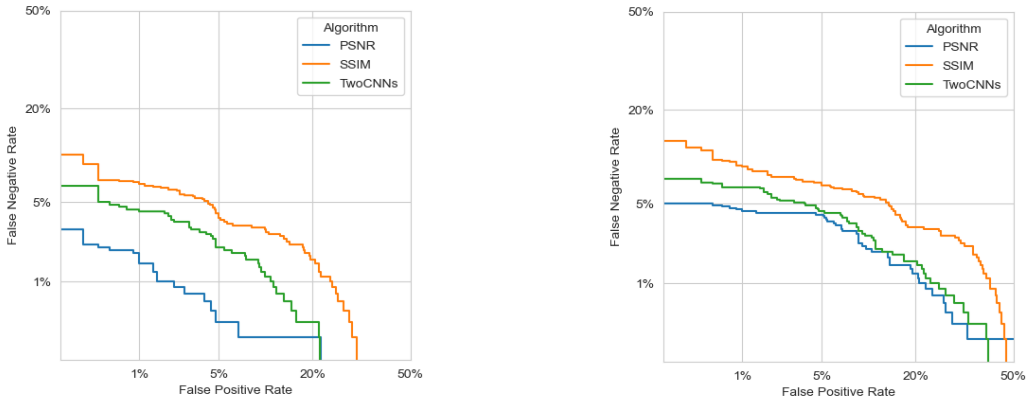

Figure 58: DET curves of the algorithms for the quality component No Compression Artefacts on Flick-Rotated JPG (left) and Flick-Rotated JP2 (right).

[29] The high-quality compressions very excluded because they are too difficult to classify, while the very low ImageMagick JPEG2000 compressions were excluded because the resulting levels of artefacts are hardly relevant for biometric applications.

The EDC Corves of the algorithms are shown in Figure 59 and Figure 60. Most decline of the FNMR is observed for the PSNR algorithm.

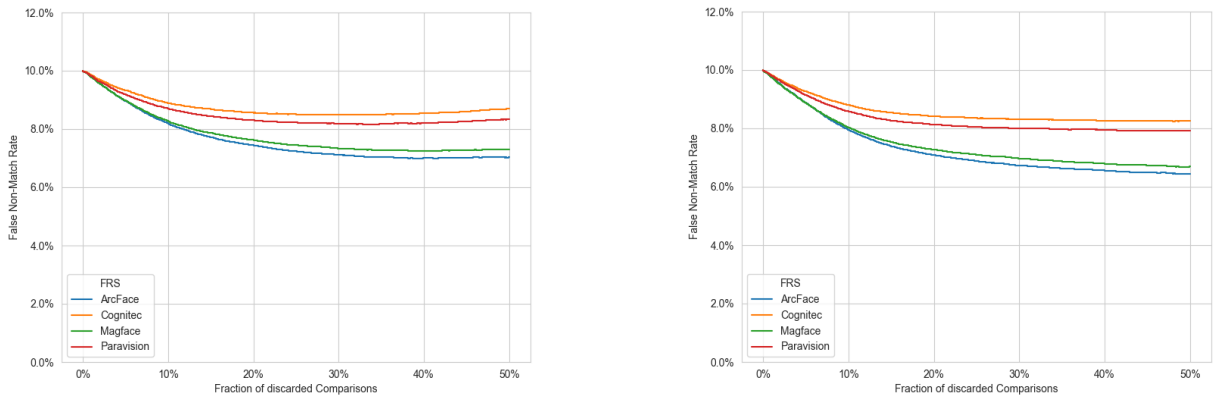

Figure 59: EDC curves for the quality component No Compression Artefacts using the algorithms PSNR (left) and the SSIM (right) on the VGGFace2 test set.

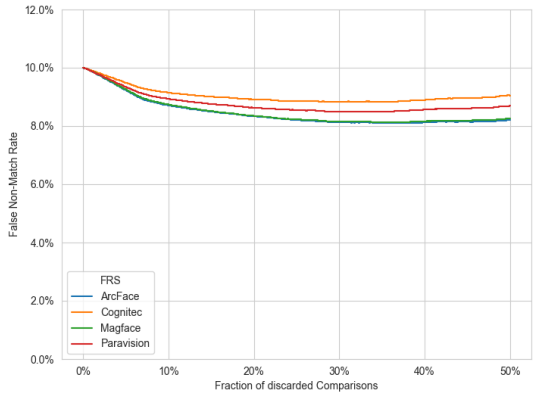

Figure 60: EDC curves the quality component No Compression Artefacts using the algorithm 2CNN on the VGGFace2 test set.

The algorithms implemented were submitted to the NIST FATE Quality SIDD track [16]. The results are shown in Figure 61. The algorithms were used for the SIDD quality component Compression Artefacts. The algorithms secunet_003 and secunet_004 represent the algorithms SSIM and PSNR, respectively, while secunet_005 is the 2CNN algorithm. Results are summarised in Figure 61. There, *d* represents the amount of applied compression, so that a perfect prediction results in a rank correlation of -1. Clearly, the 2CNN algorithm performs worst with a correlation coefficient of -0.63. The algorithms SSIM and PSNR show higher correlation (-0.86 and -0.89, respectively), but still perform worse than most other algorithms. However, it is important to note that this test only considered JPEG compression without any scaling or rotation after compression, while the proposed methods are trained to detect both JPEG and JPEG 2000 artefacts, and even after such post-processing.

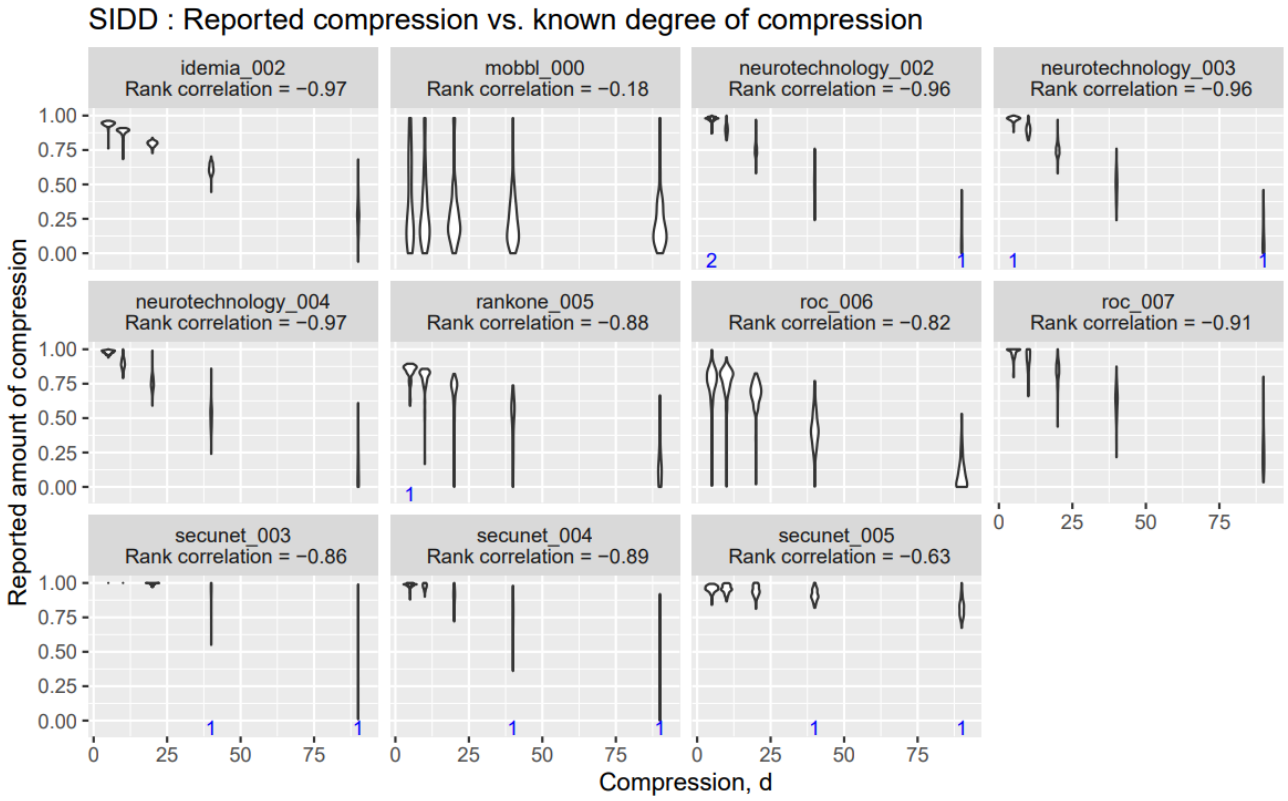

Figure 61: Outputs of the algorithms for the SIDD quality component compression in the NIST FATE Quality SIDD report [16]. The algorithms SSIM, PSNR and 2CNN are labelled as secunet_003, secunet_004 and secunet_005, respectively.

6.7.4 Final Algorithm Selection

For the quality component No Compression Artefacts, the algorithm PSNR (single CNN trained on PSNR labels) specified in Section 6.7.2.1 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \max(0, \min(100, \text{ROUND}(103\ \text{SIGMOID}(q, 0.3308, 0.092) - 0.0278))).$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

6.8 Natural Colour

6.8.1 Data Selection

Face images of unnatural colour were semi-synthetically generated based on a subset of 912 frontal images with natural colour of the FERET database [39] (containing images with minor variations in colour and saturation).

For the first test set, colour casts were generated by blending the following colours onto randomly chosen images: blue, red, yellow, cyan, green, magenta, and none. The latter colour refers to a blending on all colour channels (in contrast to the predefined colours that modify only one or two specific colour channels). The blending was implemented through the Python function *colorize()*, where the alpha values of the RGB channels of the aforementioned colours are randomly set to either 10, 20,30, or 40. Example images with colour casts are shown in Figure 62. Note that the FERET database already contains slight variations with respect to colour cast. Therefore, more severe colour casts were simulated. In total, 1,935 images with colour casts are generated, see Table 8.

Figure 62: Examples of generated colour casts, from left to right: original, blue, magenta, none.

Table 8: Overview of images in the test set with colour casts.

Colour Cast	**10%**	**20%**	**30%**	**40%**
Blue	-	95	96	94
Red	-	86	88	87
Green	-	96	87	92
Magenta		93	90	91
Yellow	88	93	97	-
Cyan	95	95	94	-
None	278			

As a second test set, face images of unnatural colour are synthetically created by increasing the saturation from the above-mentioned set of images with natural colour from FERET. To this end, the color() function of the ImageEnhance module of the Python Imaging Library is employed. Saturation factors of 1.3, 1.5, 1.7, 2.0, and 2.3 are applied to 145 randomly chosen images resulting in 725 images. Example images with unnaturally high saturation values are shown in Figure 63.

Figure 63: Examples of images with increased saturation values, from left to right: no saturation and saturation values of 1.3, 1.7, 2.3.

Another test set was set up from the ARFace database [35] by selecting 255 images with original labels 7 and 20 ("all side lights on") as samples with unnatural colour and 255 images with original labels 1 and 14 ("Neutral expression") as samples with natural colours. Example images are depicted in Figure 64.

Figure 64: Example images with unnatural colour from the ARFace test set.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.8.2 Selection and Prototyping of Candidate Algorithms

6.8.2.1 WD5 algorithm

The first candidate algorithm for Natural Colour was chosen from ISO/IEC WD5 29794-5:2022 [4]. It is based on the insight from [42] that, in the CIELAB colour space, typical skin colour values L*a*b* fit between 5-25 for a* and 5-35 for b*. In order to avoid spectacles and beards, the colour values are evaluated on measurement zones *L'* and *R'* on the cheeks specified in ISO/IEC 39794-5 Clause D.1.4.2.6 [1] (see Figure 28). In deviation to the specification of [4], the aligned image is used because determination of the measurement zones is easier without significant roll angle. Furthermore, the face image is segmented to the landmarked region to exclude pixels in the background.

The algorithm takes as input the aligned image *I'* and the transformed landmarks *L'* output by the alignment algorithm in Section 4.4.1 when taking as input the landmarks computed with the ADNet landmark estimation algorithm (Section 4.2.1.3), and performs the following steps:

2. Using the landmarks *L'*, compute the landmarked region segmentation mask S using the algorithm in Section 4.3.1.1
3. Apply the mask *S* to *I'*, i.e., set all pixels outside the landmarked region to black colour.
4. Using the landmarks *L'*, compute the left and right measurement zones, *L'* and *R'*,[30] respectively as follows:
   a. Compute the left eye centre $(X_{\mathrm{L}}, Y_{\mathrm{L}})$, the right eye centre $(X_{\mathrm{R}}, Y_{\mathrm{R}})$ and the inter-eye distance D_{IED} as in the algorithm in Section 7.7.2 with yaw angle $\alpha = 0$.[31]
   b. Compute the eyes' midpoint M_e as the mean (centre of gravity) of the left and right eye centres, i.e. as $M_{\mathrm{e}} = \big((X_{\mathrm{L}}+X_{\mathrm{R}})/2, (Y_{\mathrm{L}}+Y_{\mathrm{R}})/2\big)$.
   c. Compute the mouth's midpoint M_{m} as the mean (centre of gravity) of the landmarks marking the left and right mouth corner (90 and 94 for ADNet).
   d. Compute the Eye-Mouth Distance $D_{\mathrm{EMD}} = \|M_{\mathrm{m}} - M_{\mathrm{e}}\|_2$.
   e. Set *L'* to be the square with corners
   $$(X_{\mathrm{L}} - 0.3 \cdot D_{\mathrm{IED}}, Y_{\mathrm{L}} + 0.5 \cdot D_{\mathrm{EMD}}) \text{ and } (X_{\mathrm{L}}, Y_{\mathrm{L}} + 0.5 \cdot D_{\mathrm{EMD}} + 0.3 \cdot D_{\mathrm{IED}})$$
   f. Set R' to be the square with corners
   $$(X_{\mathrm{R}}, Y_{\mathrm{R}} + 0.5 \cdot D_{\mathrm{EMD}}) \text{ and } (X_{\mathrm{R}} + 0.3 \cdot D_{\mathrm{IED}}, Y_{\mathrm{R}} + 0.5 \cdot D_{\mathrm{EMD}} + 0.3 \cdot D_{\mathrm{IED}})$$
5. For the colour channels R, G and B, compute the mean values $(\mu_{\mathrm{R}}, \mu_{\mathrm{G}}, \mu_{\mathrm{B}})$ of all pixels of *I'* in the union of *L'* and R'.
6. Convert $(\mu_{\mathrm{R}}, \mu_{\mathrm{G}}, \mu_{\mathrm{B}})$ to CIELAB colour space using the following procedure.
   a. Normalise and do gamma inversion of the (*RGB*) values using

   $\mathrm{R_L}$ = ColourConvert($R/255$), $\mathrm{G_L}$ = ColourConvert($G/255$), $\mathrm{B_L}$ = ColourConvert($B/255$)

   where
   $$\text{ColourConvert}(V) = \begin{cases} \big((V + 0.055)/1.055\big)^{2.4} & \text{if } V > 0.04045 \\ V/12.92 & \text{if } V \geq 0.04045 \end{cases}$$
   b. Transform to XYZ space by setting
   $$X = 0.43605 \cdot R_{\mathrm{L}} + 0.38508 \cdot G_{\mathrm{L}} + 0.14309 \cdot B_{\mathrm{L}}$$
   $$Y = 0.22249 \cdot R_{\mathrm{L}} + 0.71689 \cdot G_{\mathrm{L}} + 0.06062 \cdot B_{\mathrm{L}}$$
   $$Z = 0.01393 \cdot R_{\mathrm{L}} + 0.09710 \cdot G_{\mathrm{L}} + 0.71419 \cdot B_{\mathrm{L}}$$
   c. Convert XYZ to Lab by normalising by CIE standard illuminant D50 which simulates warm daylight at sunrise or sunset with correlated colour temperature of 5003K (also known as horizon light).
   $$x_r = \frac{X}{0.964221} \qquad y_r = Y \qquad z_r = \frac{Z}{0.825211}$$
   d. Flatten the values
   $$F_x = cubic(x_r) \qquad F_y = cubic(y_r) \qquad F_z = cubic(z_r)$$
   e. where
   $$cubic(x) = \begin{cases} (kx + 16)/116 & if \quad x \leq \varepsilon \\ \sqrt[3]{x} & if \ x > \varepsilon \end{cases}$$

[30] Here, „left" and „right" is seen from the observer of the image, not from the depicted subject.
[31] The planar IED is used, because the width of the measurement zones should decrease with the yaw angle.

with $\varepsilon = 216/24389$ and $k = 24389/27$.

f. The CIELAB values are

$$L^* = 116\ F_y - 16 \qquad a^* = 500(F_x - F_y) \qquad b^* = 200(F_y - F_z)$$

7. If either a* or b* or both are less than zero then output D = 100
8. Else, compute and output $D = \sqrt{(a^* - 11.45)^2 + (b^* - 15.91)^2}$

6.8.2.2 CD1 Algorithm

The WD5 algorithm was improved in ISO/IEC CD1 29794-5:2023 [6] and implemented as second candidate algorithm. In deviation to the specification of [6], the aligned image and transformed landmarks output by the alignment algorithm in Section 4.4.1 are used as input, because the computation of the left and right measurement zones, L and R, respectively, specified in ISO/IEC 39794-5 Clause D.1.4.2.6 assumes an image without significant roll angle.

The algorithm takes as input the aligned image *I'* and the transformed landmarks *L'* output by the alignment algorithm in Section 4.4.1 when taking as input the landmarks computed with the ADNet landmark estimation algorithm (Section 4.2.1.3), and performs the following steps:

1. Using the landmarks *L'*, compute the landmarked region segmentation mask *R* using the algorithm in Section 4.3.1.1
2. Apply the mask *S* to *I'*, i.e., set all pixels outside the landmarked region to black colour.
3. Using the landmarks *L'*, compute the left and right measurement zones, Z_{L} and Z_{R} as in the WD5 algorithm (Section 6.8.2.1).
4. For the colour channels R, G and B, compute the mean values $(\mu_{\mathrm{R}}, \mu_{\mathrm{G}}, \mu_{\mathrm{B}})$ of all pixels in the union of Z_{L} and Z_{R}.
5. Convert $(\mu_{\mathrm{R}}, \mu_{\mathrm{G}}, \mu_{\mathrm{B}})$ to CIELAB colour space
6. If either a* or b* or both are less than zero then D = 100 and stop
7. Compute and output

$$D = \sqrt{\max(\max(0, 5 - a^*), \max(0, a^* - 25))^2 + \max(\max(0, 5 - b^*), \max(0, b^* - 35))^2}$$

where D is a measure of distance in (a*, b*) space from the skin tone plateau.

6.8.3 Evaluation

For the algorithms specified in WD5 and CD1, DET curves are shown in Figure 65.

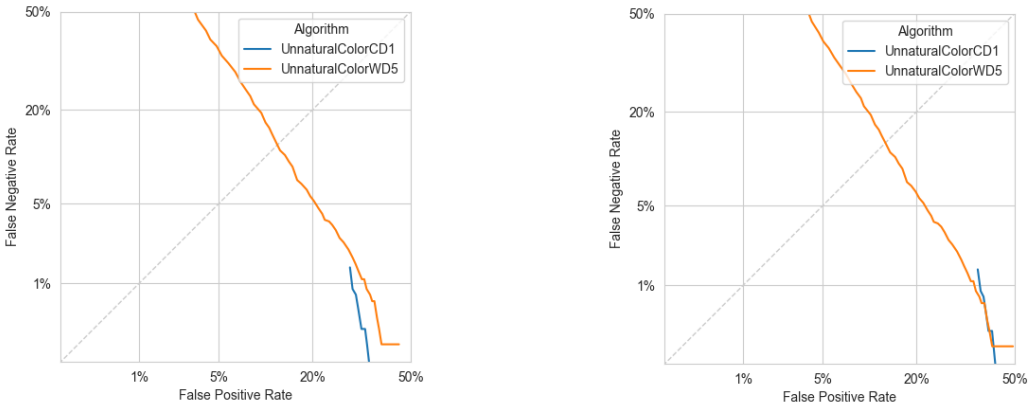

Figure 65: DET curves of the algorithms from WD5 and CD1 for the quality component Natural Colour on the FERET test set with colour cast (left) and unnatural saturation (right).

DET curves of the CD1 algorithm cover only high FPR values, which is caused by a large number of images with unnatural colour that obtain the optimal score of 0, resulting in a quality component value of 100. Some example images with their quality component value for both algorithms are given in Table 9. Therefore, it is recommended to use the algorithm of WD5 instead of CD1.

Table 9: Examples of images falsely assigned to the optimum QC value by the CD1 algorithm.

				
QC value WD5	51	39	19	39
QC value CD1	100	100	100	100

However, it should be noted that in the FERET dataset there are also images that have not been modified (label: natural) and still have a slight (yellow) colour cast due to coloured lighting. In this respect, the used test data is not optimal.

On the ARFace dataset, both algorithms achieve a clear separation of images of natural and unnatural colour. The corresponding score histograms are depicted in Figure 66. A negligible overlap of scores is observed for the WD5 algorithm while the CD1 algorithm yields a perfect separation of images with natural and unnatural colour.

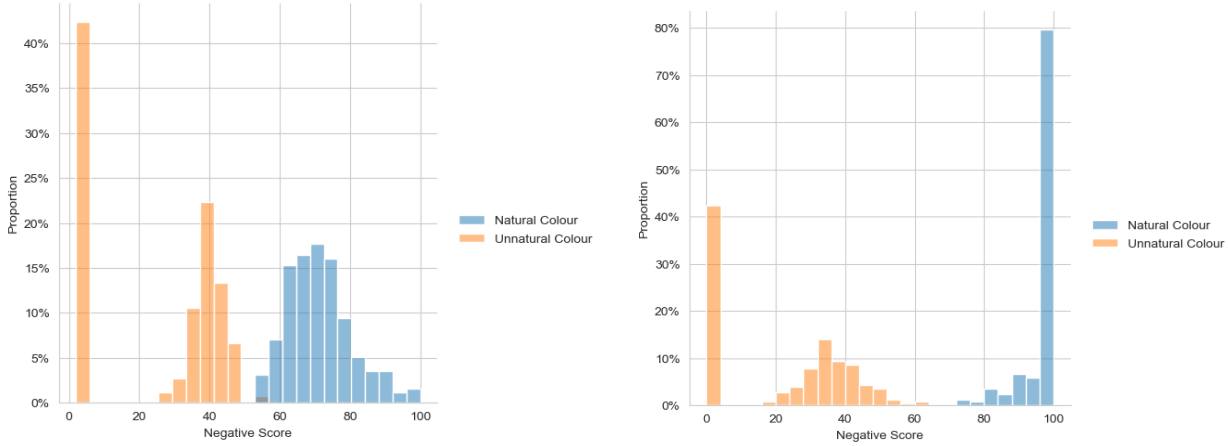

Figure 66: Score histograms of the algorithms from WD5 (left) and CD1 (right) for the quality component Natural Colour on the ARFace images.

The EDC curves of both algorithms (using ADNet for landmark estimation) on the VGGFace2 test set are shown in Figure 67 and Figure 68. Only a marginal decrease of FNMR is observed for both algorithms.

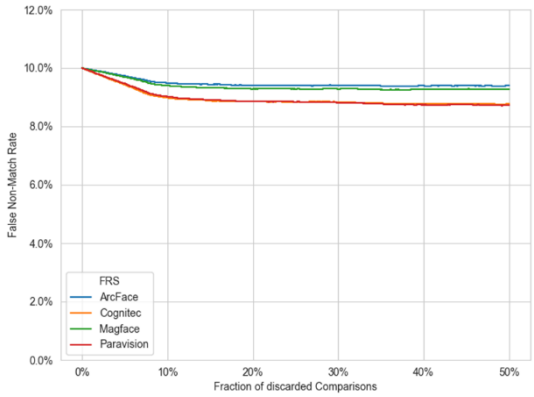

Figure 67: EDC curve for the quality component Natural Colour using the WD5 algorithm on the VGGFace2 test set.

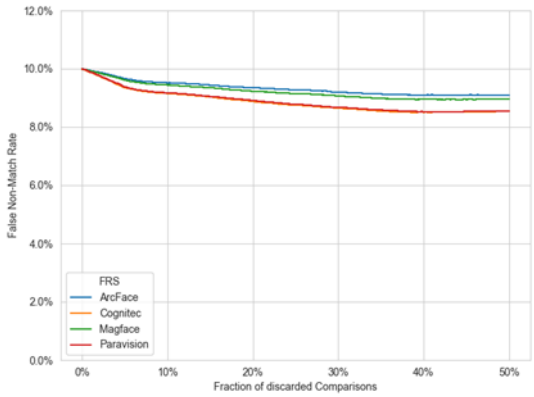

Figure 68: EDC Curve for the quality component Natural Colour using the CD1 algorithm on the VGGFace2 test set.

6.8.4 Final Algorithm Selection

For the quality component Natural Colour, the CD1 algorithm specified in Section 6.8.2.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \mathrm{ROUND}(200(1 - \mathrm{SIGMOID}(q, 0, 10)))$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

6.9 Radial Distortion Prevention

6.9.1 Data Selection

The only data set of face images with radial distortion (and corresponding labels) publicly available has been published with [43].[32] However, in these images the faces are not in the centre of the image and are thus, non-isotropically distorted, which is not a realistic scenario for face images submitted to biometric applications. The data set presented in [44] was requested by e-mail, but no answer was received.

Therefore, a new small data set was collected: 42 pairs of selfie images taken by 36 voluntaries, where each pair consisted of one image captured from a short distance (typically 15-20 cm), consequently, exhibiting significant radial distortion, and a second one captured from a full arm length distance, resulting in no perceivable distortion.

Figure 69: Example images from the collected data set with (left) and without (right) radial distortion

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

6.9.2 Selection and Prototyping of Candidate Algorithms

While several free software implementations of methods for distortion correction in photographs are available,[33, 34, 35] they don't seem to be eligible for the use case of user-submitted reference face images: The solution from [43] is specialised on wide-angle distortions of faces visible in the outer image region. In contrast, the example images shown in [45] and [46] indicate that these methods have been trained on images with prominent straight lines and equally spaced structures, which make estimation of distortions much easier. In face images, such background structures are often not be visible and, thus, it is questionable, if these methods perform well in such cases.

[32] https://people.csail.mit.edu/yichangshih/wide_angle_portrait/webpage/additional-results/index.html
[33] https://github.com/yzhq97/distortion-free-wide-angle.pytorch
[34] https://github.com/xiaoyu258/GeoProj
[35] https://github.com/ByronHsu/FEGAN

Figure 70: Example images from [45] (top) and [46] (bottom) exhibiting prominent straight lines and equally spaced structures.

Recently, a CNN-based algorithm for fish eye effect detection in facial images was developed by NTNU [47]. This algorithm was selected for development. The python code and model files were provided to secunet for use in OFIQ. According to [47], the model ModEpFcDiv .6-.9 performed best on the most challenging data set, which are the images collected by the authors (called "own dataset" or "collected dataset" in [47]) cropped to the face region. This model was trained on face images roughly cropped to the face region, and revealed, according to the analysis in [47], less overfitting on straight or regular structures visible in the background. Thus, this model was converted to ONNX and used for the prototype implementation.

The resulting algorithm takes as input an aligned input image I in BGR colour channel order output by the algorithm in Section 4.4.1:

1. Convert I to RGB colour channel order.
2. Crop I by 170 pixels from the left and right, respectively, by 195 from the top and by 116 from the bottom.
3. Resize I to dimension (244,244) using bilinear interpolation.
4. Normalise I with mean $\mu = (123.7, 116.3, 103.5)$ and standard deviation $\sigma = (58.4, 57.1, 57.4)$ to obtain array A by setting, for all i, j, k,

$$A_{i,j,k} = (I_{i,j,k} - \mu_k)/\sigma_k$$

5. Reshape A as input tensor T_1 of dimensions (1, 3, 244, 244) for the neural network model.
6. Feed the tensor T_1 to the CNN model ModEpFcDiv .6-.9 to obtain an output tensor T'_1 of dimensions (1,1).
7. Cast T_1 to a scalar value s.
8. Output s.

6.9.3 Evaluation

The DET curve of the NTNU algorithm on the internal test set (see Section 6.9.1) are shown in Figure 71. Obviously, its predictive performance is hardly better than coin flipping.

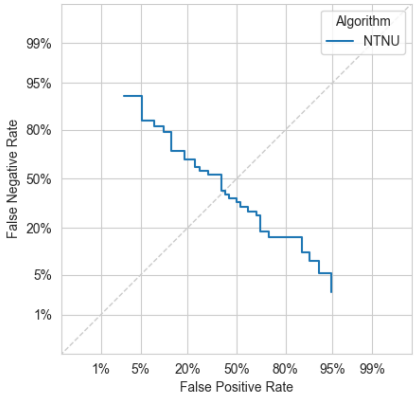

Figure 71: DET curve of algorithm for the quality component No Radial Distortion on the internal test set.

The EDC curves of the NTNU algorithm on the VGGFace2 test set in Figure 72 show a slight decrease of FNMR. A visual inspection of the images assigned to the lowest scalar values indicates that the algorithm frequently misclassifies head coverings as radial distortion.

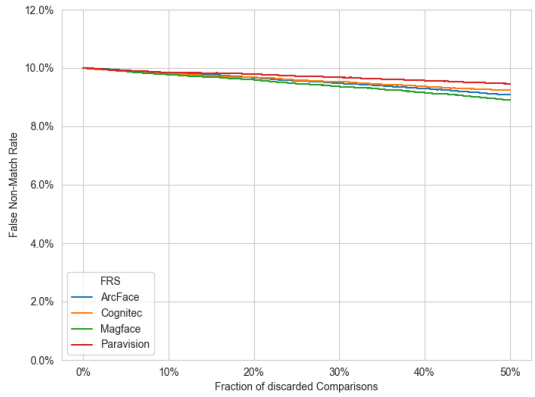

Figure 72: EDC curves for the quality component No Radial Distortion using the NTNU algorithm on the VGGFace2 test set.

6.9.4 Final Algorithm Selection

The quality component Radial Distortion Prevention was discarded in ISO/IEC WD6 29794-5:2023 [5] and in OFIQ, because no sufficiently effective algorithm was available.

7 Subject-related Quality Components

Subject-related quality components mainly depend on how the subject presents his/her face during the capture process.

7.1 Single Face Present

7.1.1 Data Selection

No data was collected for implementation and evaluation.

7.1.2 Selection and Prototyping of Candidate Algorithms

The algorithm from ISO/IEC FDIS 29794-5:2024 [10] was implemented. It outputs a face unicity measure as the fraction of the areas of the second largest and largest bounding boxes output the face detection algorithm. It takes as input the list (a_i, b_i, c_i, d_i) of face bounding boxes output by the SSD face detection algorithm described in Section 4.1.1.3, and performs the following steps:

1. If zero, output "failureToAssess" and terminate.
2. If the number of bounding boxes is, output the face unicity as $f = 0$.
3. Otherwise, output the face unicity as

$$f = \frac{(c_2 - a_2) \cdot (d_2 - b_2)}{(c_1 - a_1) \cdot (d_1 - b_1)}$$

7.1.3 Evaluation

No evaluation of the algorithm for the quality component Single Face Present was performed.

7.1.4 Final Algorithm Selection

For the quality component Single Face Present, the FDIS algorithm specified in Section 7.1.1 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(100(1- q))$$

is used, where ROUND is defined as described in Section 5.4.

7.2 Eyes Open

7.2.1 Data Selection

7.2.1.1 Training Set

A training set was compiled from the following images:

- 2,343 images showing closed eyes from the Closed Eyes in the Wild (CEW) dataset [48].
- 324 images with closed eyes from the Eye_Training_Set of the RPI ISL Eye Database [49].
- 744 images with closed eyes and 776 images with open eyes from the Eyeblink8 dataset [50].
- 2,655 images with closed eyes and 2,767 images with open eyes from the FEAFA dataset [51]. Labels for the eye openness state were deducted from the labels for the facial action units “Left Eye Close“ and „Right Eye Close”.
- 486 images of open eyes of subjects of race “Asian” or “Asian-Southern” and with yaw between -15° and 15° from the FERET database [39]. In addition, 14 images of closed eyes were manually chosen.
- 404 images of open eyes of subjects of race “Asian” or “Asian-Southern” from the FRGCv2 database [34]. In addition, 6 images of closed eyes were manually chosen from FRGCv2.
- 2,966 images of open eyes from the VGGFace2 dataset [15].
- 726 images of closed eyes from the repository Eye-State-Detection[36].

Generally, only images with sufficient resolution and without occlusions were selected. Furthermore, in cases, where a dataset contained many very similar images (e.g. in FEAFA), only a selection of sufficiently different images was used. In summary, the training set contains 6,812 images of closed eyes and 7,399 images of open eyes.

From images showing the complete face (CEW, Eyeblink8, FEAFA, FERET, FRGCv2, VGGFace2), images of the eyes were extracted with the following procedure:

- Landmarks are computed.
- For each eye, the centre *e* of the eye is computed as the centre of mass $e = (e_1 + e_2)$ of both canthi e_1, e_2.
- The image is rotated so that, for both eyes, the y-coordinate e_y of the eye’s centre *e* has the same value.
- For each eye to be extracted, a square bounding box (a, b, c, d) is computed as follows:
    - The left and right border *a* and *c* of the box are set to have a distance of 30% of the eye’s width (distance of both canthi) to the respective canthus, i.e. set to $a = e_x - 0.8 \cdot (e_1 - e_2)$ and $c = e_x + 0.8 \cdot (e_1 - e_2)$, where e_x denotes the x-coordinate of the eye’s centre *e*.
    - The upper and lower border *b* and *d* of the box are chosen so that they have equal distance to the eye’s centre. i.e. $e_y - b = d - c_y$, and that $d - b = c - a$.
- For each eye to be extracted, the image is cropped to the respective bounding box, scaled to dimensions 128x128 and saved as image of the eye.

[36] https://github.com/purnenduvashistha/Eye-State-Detection

For images from FEAFA or Eyeblink8, where the label indicates that only one of the eyes is closed, only this eye was used; in all other cases, both eyes were used. For data augmentation, all images were horizontally flipped and saved as additional images.

The images from the RPI ISL Eye Database and from the repository Eye-State-Detection were already suitably cropped.

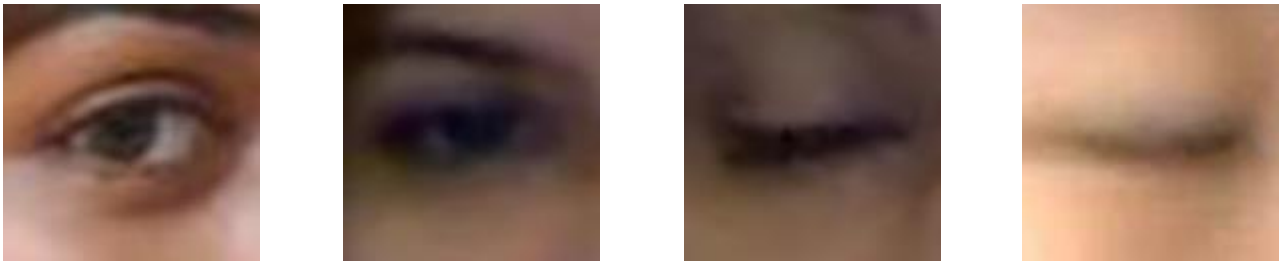

Figure 73: Examples of cropped images in the training set

All images were accordingly labelled with the binary labels “Eye Open” and “Eye Closed”.

7.2.1.2 Test Sets

As a first test set, 1,417 grayscale face images with open and closed eyes have been selected from the CAS-PEAL-R1 database [38]. Example images for both classes are shown in Figure 75.

As a second test set, 510 colour images have been chosen from the ARFace database [35] with open eyes (indexes 1 and 14) and closed eyes (indexes 4 and 17). Example images are shown in Figure 74. Summarises the number of images in each class for both datasets.

Figure 74: Example images of the ARFace test set with open (left) and closed eyes (right).

Figure 75: Example images of the CAS-PEAL-R1 test set with open (left) and closed eyes (right).

Table 10: Number of images in used test sets with closed and open eyes.

Test Set	Eyes open	Eyes closed
CAS-PEAL-R1	1,040	377
ARFace	255	255

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.2.2 Selection and Prototyping of Candidate Algorithms

7.2.2.1 WD5 Algorithm

The first candidate algorithm was taken from ISO/IEC WD5 29794-5:2022 [4]. It takes as input the transformed landmarks L' output by the alignment algorithm in Section 4.4.1, and performs the following steps:

1. Using the landmarks L', compute the inter-eye distance D_{IED} by the algorithm in Section 7.7.2.
2. Measure the largest distance between upper and lower eyelids in the subject's left eye, D_{L}, as the maximum of the distances between the following pairs of landmarks.
   - For dlib, the pairs of landmarks (L'_{43}, L'_{47}) and (L'_{44}, L'_{46}) are used.
   - For MediaPipe, the pairs of landmarks (L'_{385}, L'_{380}), (L'_{386}, L'_{374}), (L'_{387}, L'_{373}) are used.
   - For ADNet, the pairs of landmarks (L'_{69}, L'_{75}), (L'_{70}, L'_{74}), (L'_{71}, L'_{73}) are used.
3. Measure the largest distance between the upper and lower eyelids in the subject's right eye, D_{R}, as the maximum of the distances between the following pairs of landmarks.
   - For dlib, the pairs of landmarks (L'_{41}, L'_{37}) and (L'_{40}, L'_{38}) are used.
   - For MediaPipe, the pairs of landmarks (L'_{144}, L'_{160}), (L'_{145}, L'_{159}), (L'_{153}, L'_{158}) are used.
   - For ADNet, the pairs of landmarks (L'_{61}, L'_{67}), (L'_{62}, L'_{66}), (L'_{63}, L'_{65}) are used.
4. Compute the palpebral aperture as the smaller of the two, $D_{\mathrm{PAL}} = \min(D_{\mathrm{L}}, D_{\mathrm{R}})$
5. Compute and output the eye openness aspect

$$\omega = \frac{D_{\mathrm{PAL}}}{D_{\mathrm{IED}}}$$

7.2.2.2 Eye Aspect Ratio (EAR) Algorithm

The second candidate algorithm was taken from [52] and uses several landmarks of the left and right eye that are depicted in Figure 76 to compute a measure Eye Aspect Ratio (EAR) for the openness of the eyes of a subject. It takes as input the landmarks output by one of the landmark estimation algorithms in Section 4.2.1, and computes, for each eye $e \in$ {left,right}, the EAR_e according to the following formula:

$$\mathrm{EAR}_e = \frac{||p_2 - p_6|| + ||p_3 - p_5||}{2||p_1 - p_4||},$$

where the points p1, …, p6 are assigned to the following landmarks:

- dlib:
  - For the subject's right eye: 36, 37, 38, 39, 40, 41.
  - For the subject's left eye: 42, 43, 44, 45, 46, 47.
- MediaPipe:
  - For the subject's right eye: 33, 160, 158, 133, 153, 144.
  - For the subject's left eye: 362, 385, 387, 263, 373, 380.
- ADNet:
  - For the subject's right eye, 60, 61, 63, 64, 65, 67.
  - For the subject's left eye, 68, 69, 71, 72, 73, 75.

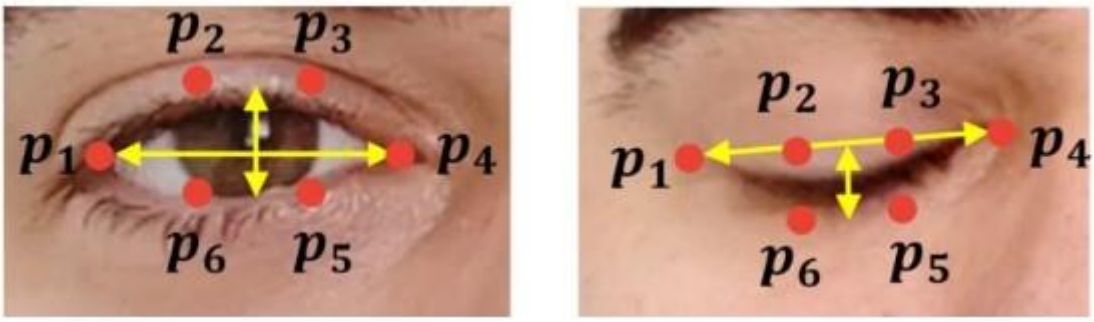

Figure 76: Landmarks used to estimate the Eye Aspect Ratio (EAR).

Finally, the minimum of EAR_{left} and EAR_{right} is output as native score.

7.2.2.3 CD1 Algorithm

In ISO/IEC CD1 29794-5:2023 [6], the definition of the native quality measure was changed in comparison with the algorithm from WD5: The minimum over both eyes of the maximal distance between upper and lower lid is divided by the distance between the chin and the eyes' midpoint. The CD1 algorithm takes as input the transformed landmarks L' output by the alignment algorithm in Section 4.4.1 and performs the following steps:

1. Using the landmarks L', compute the palpebral aperture D_{PAL} using the steps 1-3 of the WD5 algorithm.
2. Using the landmarks L', compute the distance T between the eyes' midpoint and the chin as follows:
   a. Compute the left eye centre $(X_{\text{L}}, Y_{\text{L}})$ and the right eye centre $(X_{\text{R}}, Y_{\text{R}})$ as in the algorithm in Section 7.7.2.
   b. Compute the eyes' midpoint Me as the mean (centre of gravity) of the left and right eye centres, i.e. as $M_{\text{e}} = \big((X_{\text{L}}+X_{\text{R}})/2, (Y_{\text{L}}+Y_{\text{R}})/2\big)$.
   c. Set the position C of the chin to:
      i) Landmark L'_8 for dlib.
      ii) Landmark L'_{152} for MediaPipe.
      iii) Landmark L'_{16} for ADNet.
   d. Compute the distance $T = \|M_{\text{e}} - C\|_2$ between the eyes' midpoint and the chin.
3. Compute and output the eye openness aspect

$$\omega = \frac{D_{\text{PAL}}}{T}$$

For evaluation, SSD was used for face detection and ADNet for landmarks estimation (Section 4.2.1.3).

7.2.2.4 Algorithm using Eyes Open CNN

In addition, a completely new algorithm was implemented based on the implementation published in the repository Eye-State-Detection[36]. In this implementation, the openness state of the eye (eye open or eye closed) is estimated with a small CNN consisting of 6 layers. Since this model was trained on a rather small training set (1,012 images), it was re-trained using the Python code from the repository and the training set described in Section 7.2.1. Furthermore, the dimension of the input image was changed from 256x256 to 56x56, which was found to give the same accuracy on the test sets ARFace and CAS-PEAL-R1.

The resulting algorithm takes as input the aligned image *I* in RGB colour channel order with 8 bits per channel and the aligned landmarks and performs the following steps:

1. For both the left and right eye, execute the following steps:
    a. Using the OpenCV function boundingRect, compute the minimum upright rectangle $R = (a, b, c, d)$, where (a, b) is the upper left and (c, d) the lower right corner, containing the landmarks bounding the eye. For ADNet, the landmarks indices are 60, …, 67 for the subject's right eye and 68, …, 75 for the subject's left eye.
    b. On the left and right side, extend the rectangle by 30% of its width, i.e. set $a = a - 0.3 \cdot (c - a)$ and $c = c + 0.3 \cdot (c - a)$.
    c. Extend the rectangle equally on the top and the bottom to obtain a square, i.e. set $b = \frac{b+d}{2} - \frac{c-a}{2}$ and $b = \frac{b+d}{2} + \frac{c-a}{2}$.
    d. Crop I to the rectangle $R = (a, b, c, d)$.
    e. Resize *I* to size 56x56 using bilinear interpolation.
    f. Normalise *I* using mean $\mu = 0$ and standard deviation $\sigma = 255$ to obtain the array *A*, i.e. set
    $$A_{i,j,k} = I_{i,j,k}/255$$
    g. Encode A as input tensor T of dimensions (1, 3, 56, 56) for the neural network model.
    h. Run a forward pass through the eyes open model using T as input to obtain an output tensor T' of dimensions (1,2).
    i. Set the openness score *S* of the eye to T'_2.
2. Output $X = \min(S_\mathrm{l}, S_\mathrm{r})$, where $S_\mathrm{l}, S_\mathrm{r}$ are the openness scores of the left or right eye, respectively.

7.2.3 Evaluation

The algorithms that only rely on landmark locations are very eligible to evaluate the accuracy of the landmark estimation algorithms. We did so by evaluating the equal error rate (EER) of the Eyes Aspect Ratio (EAR) algorithm on both test sets using all 3 landmark estimation algorithms (Section 4.2.1). As shown in Table 11, by far, the lowest error rates are achieved when using the ADNet landmark estimation (Section 4.2.1.3) and the lowest are achieved when using dlib.

Table 11: Comparison of the EAR algorithm on the test sets using different landmark estimation algorithms

Test Set	SSD + dlib	MediaPipe	SSD + ADNet
ARFace	11.4%	4.3%	1.2%
CAS-PEAL-R1	3.4%	0.8%	0.0%

We then evaluated all 4 algorithms on the two test sets using ADNet landmark estimation (Section 4.2.1.3). The DET curves of the algorithms on ARFace are plotted in Figure 77. The CD1 algorithm performs best (EER = 0.8%) and the EAR algorithm is second (EER = 1.2%), while, overall, the CNN based algorithm has the highest error rates (EER = 2.0%).

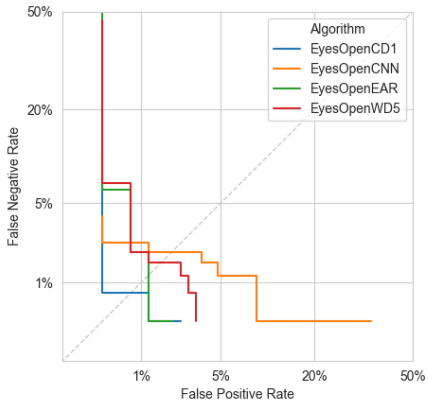

Figure 77: DET curves for the algorithms for the quality component Eyes Open using ADNet landmarks on the ARFace test set.

On CAS-PEAL-R1, all algorithms, when using ADNet landmark estimation (Section 4.2.1.3), achieve a perfect separation of the two classes (Figure 78).

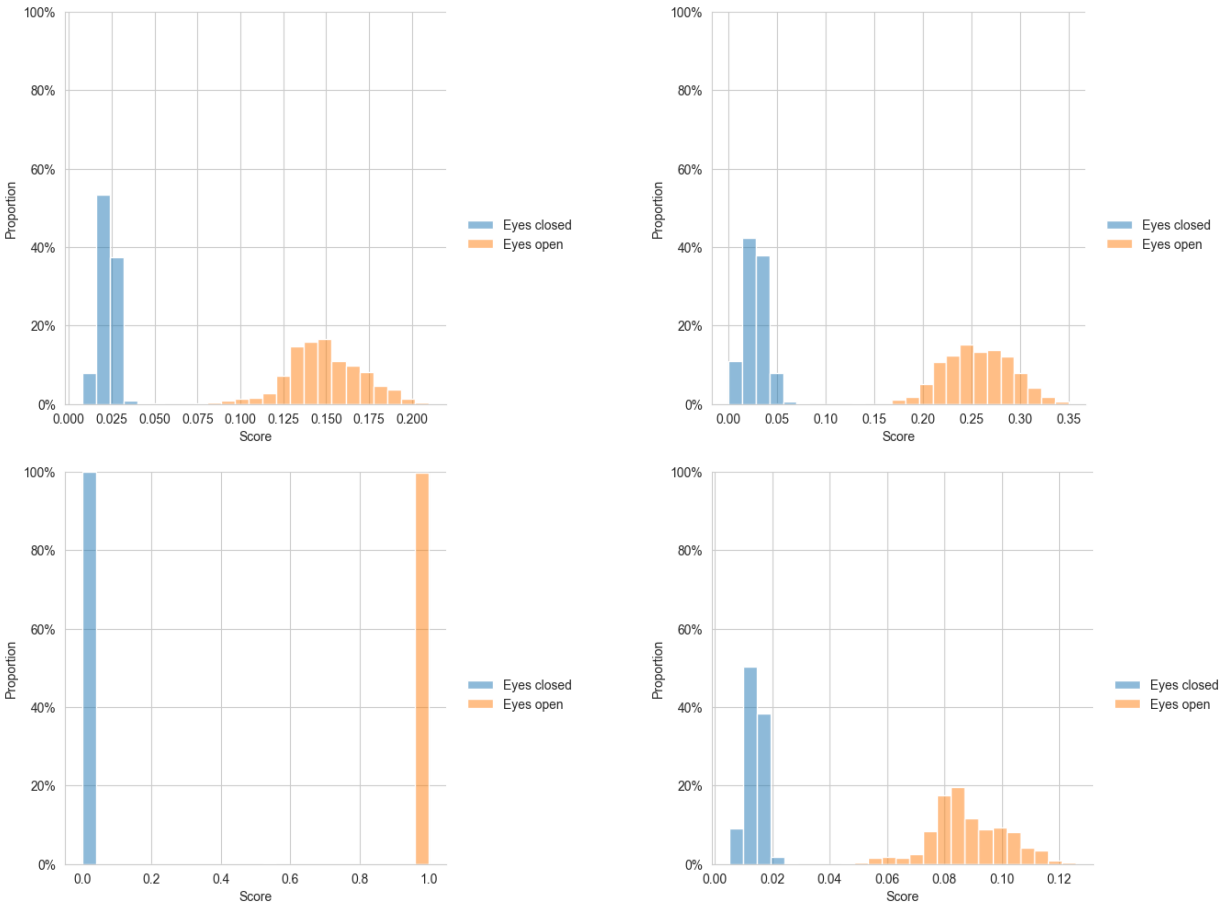

Figure 78: Score histograms of the WD5 algorithm (top left), the EAR algorithm (top right), the CNN algorithm (bottom left) and the CD1 algorithm (bottom right) of the quality component Eyes Open using ADNet landmarks on the CAS-PEAL-R1 test set.

The EDC curves of the algorithms for Eyes Open on the VGGFace2 test set are shown in Figure 79 and Figure 80.

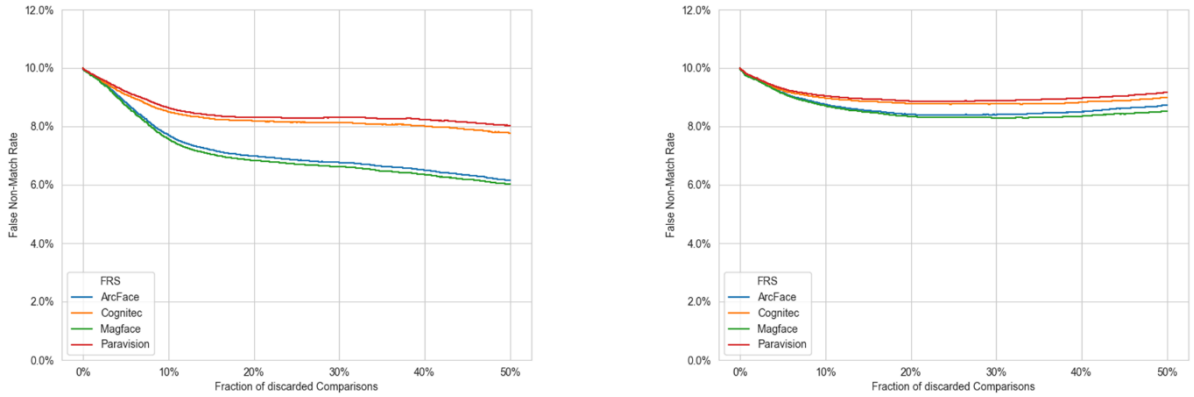

Figure 79: EDC curves for the quality component Eyes Open using the WD5 algorithm (left) and the EAR algorithm (right) using ADNet landmarks on the VGGFace2 test set.

We observe that the EDC curve of Eyes Open EAR features only a slight decline of FNMR. A potential explanation is that, when using ADNet landmarks, the Eyes Open EAR algorithm measures the aperture as the average distance between the outer landmarks of the upper and lower lid, neglecting the middle pair, which results in a smaller estimate for the aperture than in the WD5 and CD1 algorithms, where the maximum of the distances of all 3 pairs of landmarks is used. For images with small resolution, this gives a

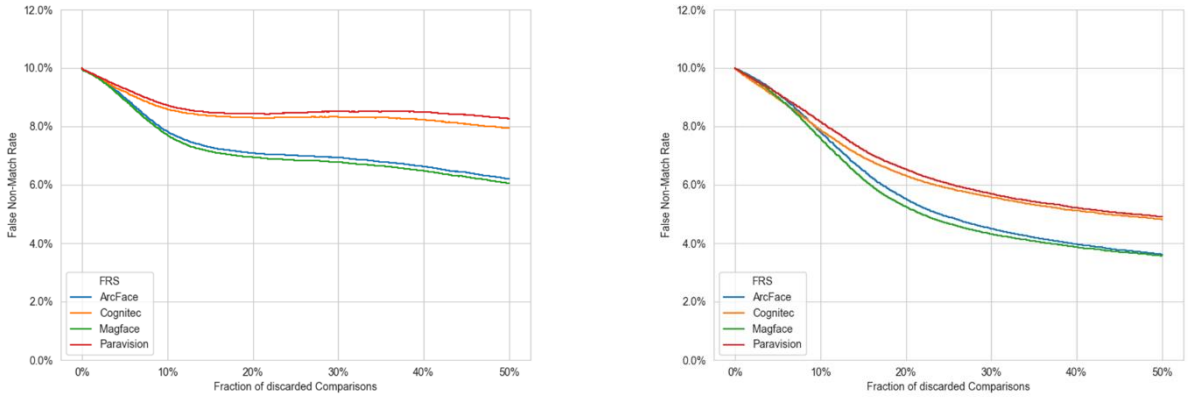

Figure 80: EDC curves for the quality component Eyes Open using the CD1 algorithm (left) and the algorithm based on the occlusion estimation CNN (right) using ADNet landmarks on the VGGFace2 test set.

much coarser estimate, resulting in a less accurate measurement.

On the other hand, the algorithm based on the eyes open CNN shows by far the highest drop of FNMR. However, this finding should be taken with care because a visual inspection of the images to which the CNN based algorithm assigns the lowest scores reveals that in many cases the eyes are hardly visible (due to occlusions or illumination). While discarding such images certainly reduces the FNMR, it is not the aim of the algorithm.

The WD5 algorithm and the CD1 algorithm, both using ADNet landmarks, were inlcuded in the submission secunet_003 to the NIST FATE Quality SIDD track [16].

The WD5 algorithm was implemented for the SIDD component Eyes Open, which measured the ratio between the eyes' aperture and the IED (as in the WD5 algorithm). As visible in Figure 81, it achieved the lowest MAE of all submitted algorithms.

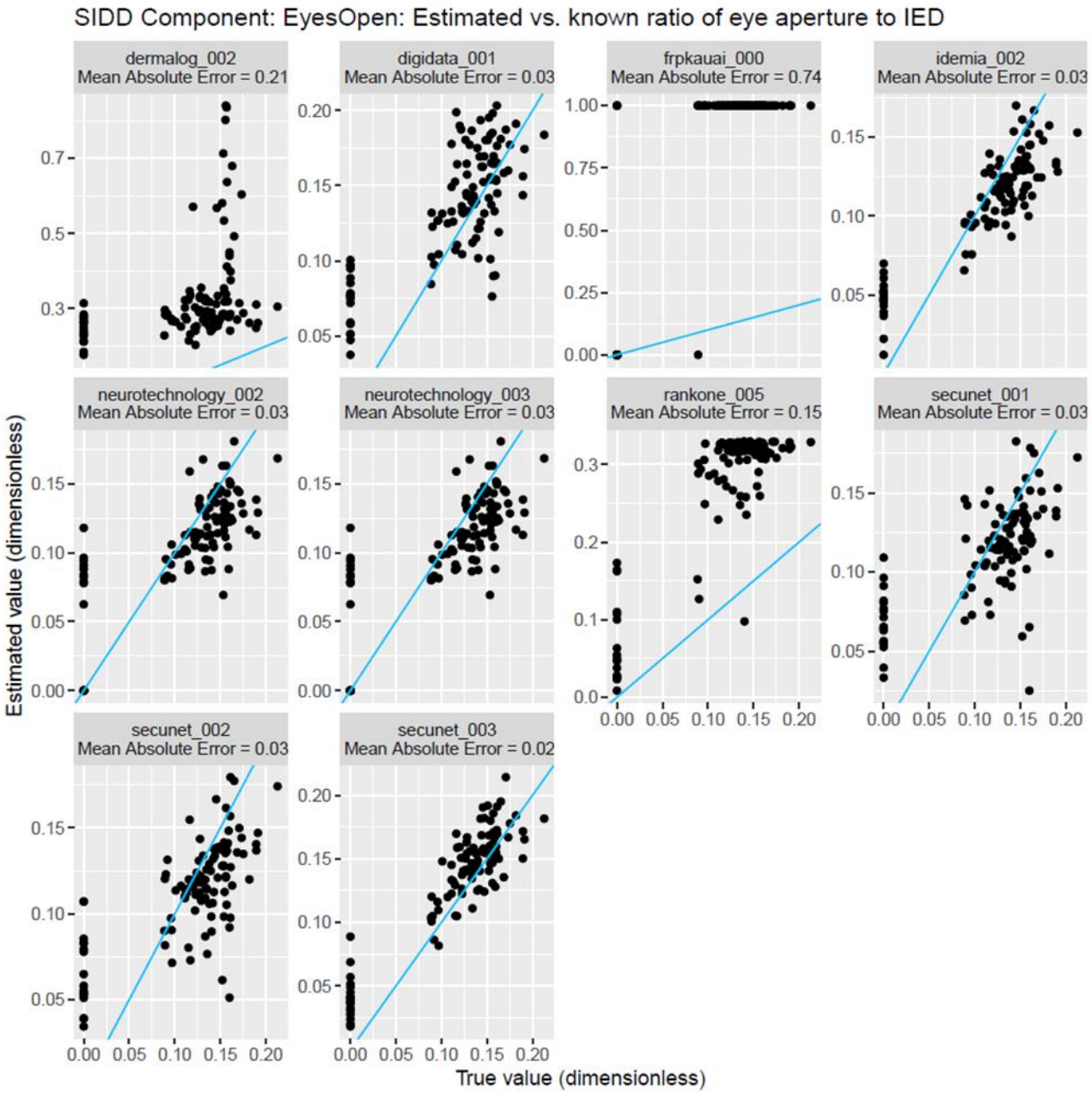

Figure 81: Scatter Plots and MAE of the outputs of the algorithms for the SIDD component Eyes Open vs. ground truth in the NIST FATE Quality SIDD report [16]. The WD5 algorithm using ADNet landmarks is labelled as secunet_003, while secunet_002 denoted the WD5 algorithm using dlib landmarks.

The CD1 algorithm was implemented for the SIDD component Eyes Open 2, which measures the ratio between the eyes' aperture and the T-metric (distance between the eyes' midpoint and the chin, as computed in the CD1 algorithm). The CD1 algorithm achieved the lowerst MAE of all submitted algorithms on par with another algorithm (viante_000).

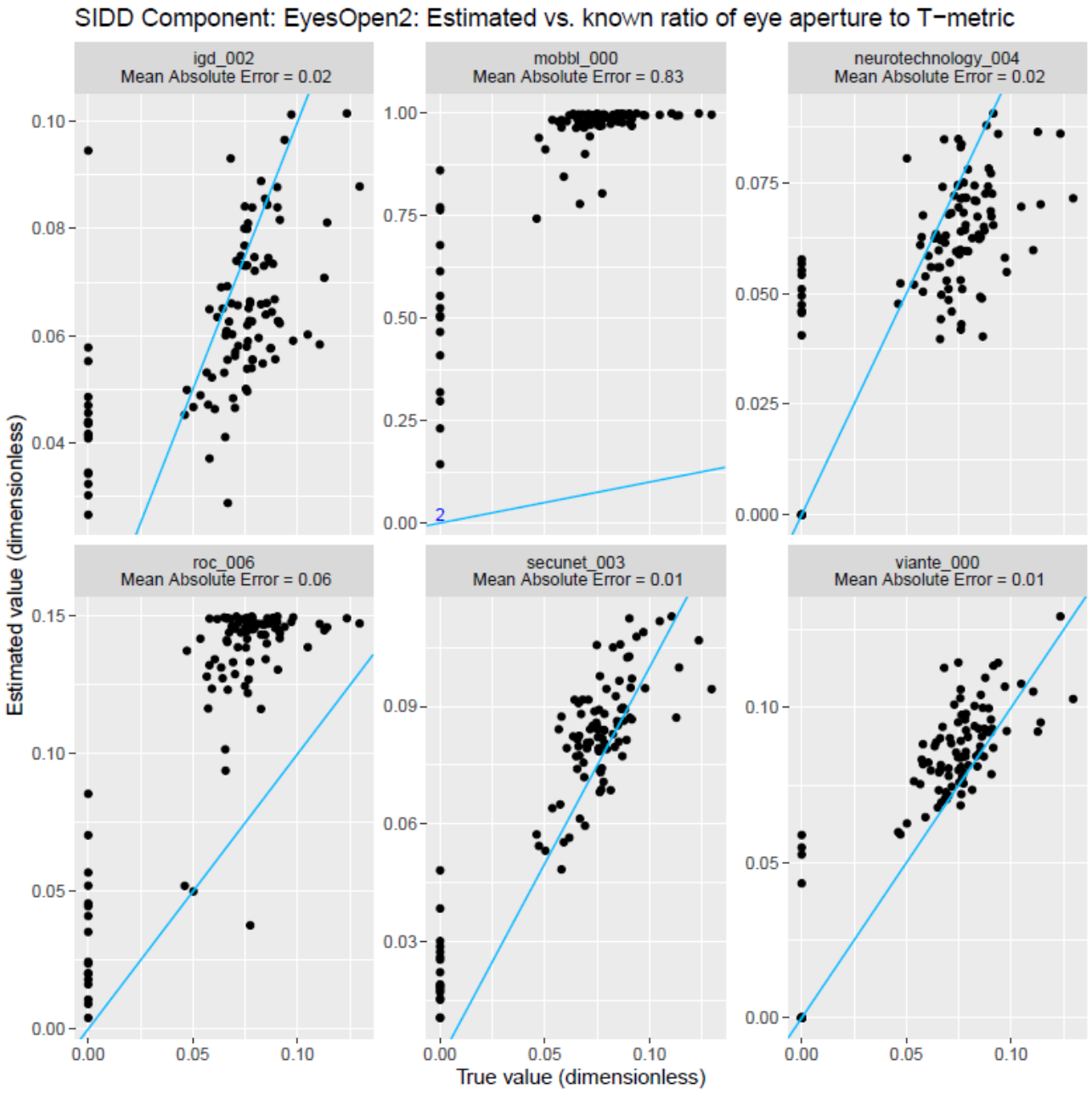

Figure 82: Scatter Plots and MAE of the outputs of the algorithms for the SIDD component Eyes Open 2 vs. ground truth in the NIST FATE Quality SIDD report [16]. The CD1 algorithm using ADNet landmarks is labelled as secunet_003.

7.2.4 Final Algorithm Selection

For the quality component Eyes Open, the CD1 algorithm specified in Section 7.2.2.3 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(100\ \text{SIGMOID}(q, 0.02, 0.01))$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

7.3 Mouth Closed

7.3.1 Data Selection

Test sets were compiled from three public databases, namely the Chicago Face Database [53], the Eurecom Kinect Face [54] and CAS-PEAL-R1 [38], containing facial images with open and closed mouth have been selected. Example images are depicted in the figures below and an overview of the number of images contained in each test set and class (mouth open/closed) are listed in Table 12.

Figure 83: Example images of the Chicago Face test set with open (left) and closed mouth (right).

Figure 85: Example images of the CAS-PEAL-R1 test set with open (left) and closed mouth (right).

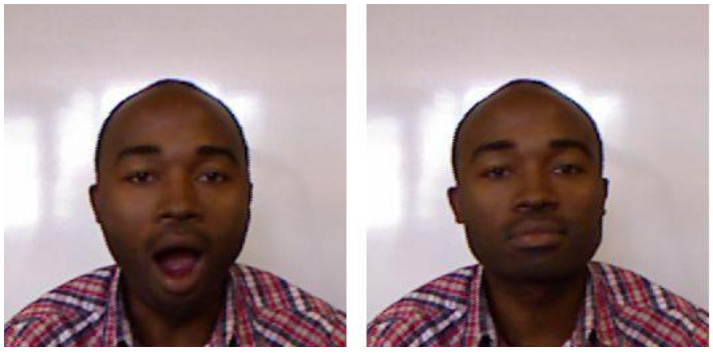

Figure 84: Example images of the Eurecom Kinect Face test set with open (left) and closed mouth (right).

Table 12: Number of images in the used test sets with closed and open mouth.

Test Set	Mouth closed	Mouth open
CAS-PEAL-R1	976	439
Eurecom Kinect Face	101	107
Chicago Face Database	831	154

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.3.2 Selection and Prototyping of Candidate Algorithms

7.3.2.1 WD4 Algorithm

The first candidate algorithm was taken from ISO/IEC WD4 29794-5:2022 [3]. It takes as input the landmarks L output by one of the landmark estimation algorithms in Section 4.2.1, and performs the following steps:

1. Using the landmarks L, compute the inter-eye distance D_{IED} by the algorithm in Section 7.7.2.
2. Using the landmarks L, determine the maximum distance between the inner borders of the upper and lower lip, D_L, as the maximum of the distances between the following pairs of landmarks.
   - For dlib, the pairs of landmarks (L_{61}, L_{67}), (L_{62}, L_{66}), (L_{63}, L_{65}) are used.
   - For MediaPipe, the pairs of landmarks (L_{82}, L_{87}), (L_{13}, L_{14}), (L_{312}, L_{317}) are used.
   - For ADNet, the pairs of landmarks (L_{95}, L_{89}), (L_{94}, L_{90}), (L_{93}, L_{91}) are used.
3. Compute and output the mouth openness aspect

$$\omega = \frac{D_{\mathrm{L}}}{D_{\mathrm{IED}}}$$

7.3.2.2 CD1 Algorithm

The second candidate algorithm uses the distance between eyes midpoint and chin instead of the inter-eye distance. It takes as input the landmarks output by one of the landmark estimation algorithms in Section 4.2.1 and performs the following steps:

1. Using the landmarks L, compute the distance T between the eye's midpoint and the chin as in the algorithm in Section 7.2.2.3.
2. Using the landmarks L, determine the maximum distance between the inner borders of the upper and lower lip, D_{L}, as in the WD4 algorithm (Section 7.3.2.1).
3. Compute and output the mouth openness aspect

$$\omega = \frac{D_{\mathrm{L}}}{T}$$

7.3.3 Evaluation

The algorithms that only rely on landmark locations are very eligible to evaluate the accuracy of the landmark estimation algorithms. We did so by evaluating the equal error rate (EER) of the WD4 algorithm for Mouth Closed on all test sets using the 3 landmark estimation algorithms (Section 4.2.1). As shown in Table 13, by far, the lowest error rates are achieved when using the ADNet landmark estimation (Section 4.2.1.3).

Table 13: Comparison of the WD4 algorithm for mouth closed with different landmark estimation algorithms

Test Set	SSD + dlib	MediaPipe	SSD + ADNet
Chicago Face Database	2.6%	1.0%	0.6%
CAS-PEAL-R1	6.6%	12.5%	2.7%
Eurecom Kinect Face	0.7%	0.5%	0.0%

The DET curves of the implemented algorithms using the ADNet landmark estimation for the test sets CAS-PEAL-R1 and Chicago Face Database are plotted in Figure 87. On Chicago Face Database, the WD4 algorithm seems to be better for smaller FNR, but the extrapolated EER is identical (0.6%). On CAS-PEAL-R1, both algorithms perform equally well (EER of 2.7%). On the test set Eurecom Kinect Face, both algorithms achieve a perfect separation (EER=0%) as shown in Figure 86.

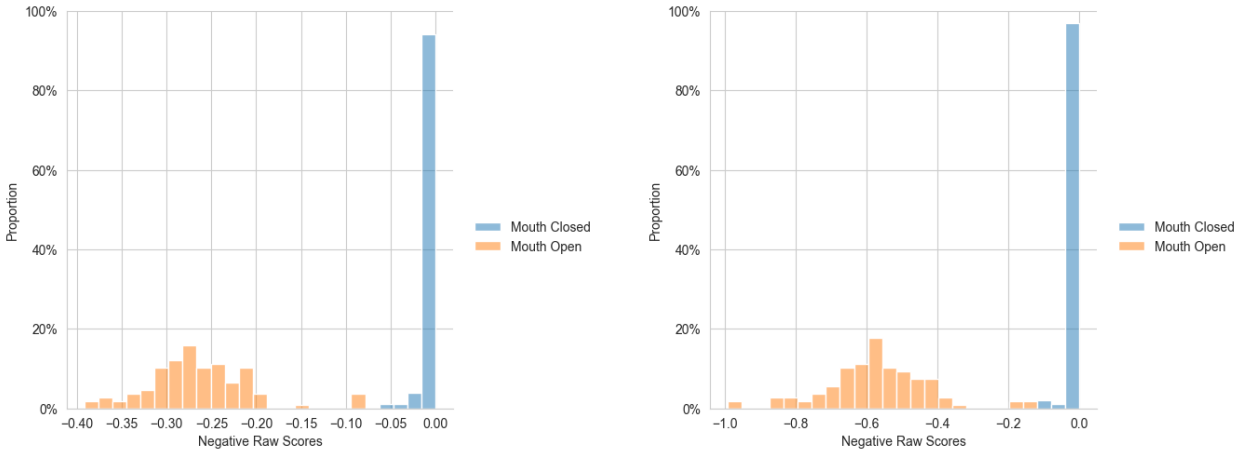

Figure 86: Score distributions of the CD1 algorithm (left) and the WD4 algorithm (right) for the quality component Mouth Closed using SSD and ADNet on the test set Eurecom Kinect Face.

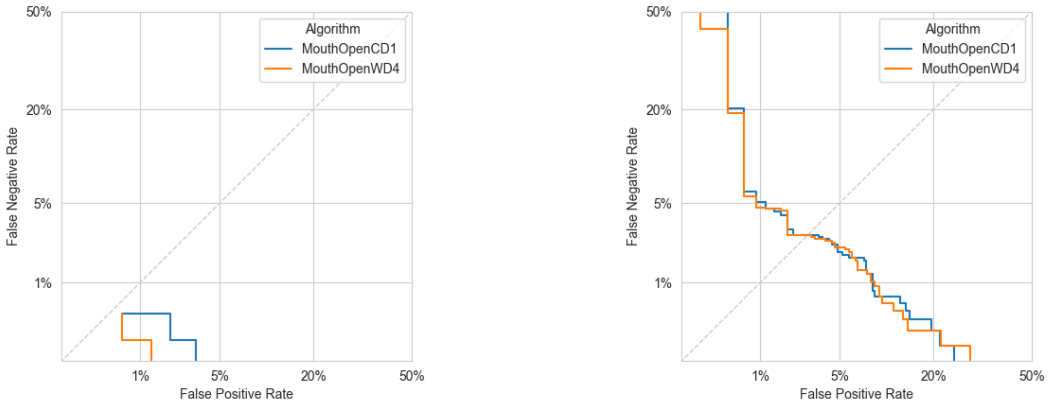

Figure 87: DET curves of the algorithms for the quality component Mouth Closed using SSD and ADNet on the test sets Chicago Face Database (left) and CAS-PEAL-R1 (right).

The EDC curves of the implemented algorithms using the SSD face detector with ADNet landmark estimation in Figure 88 show a slightly stronger decline of FNMR for the CD1 algorithm.

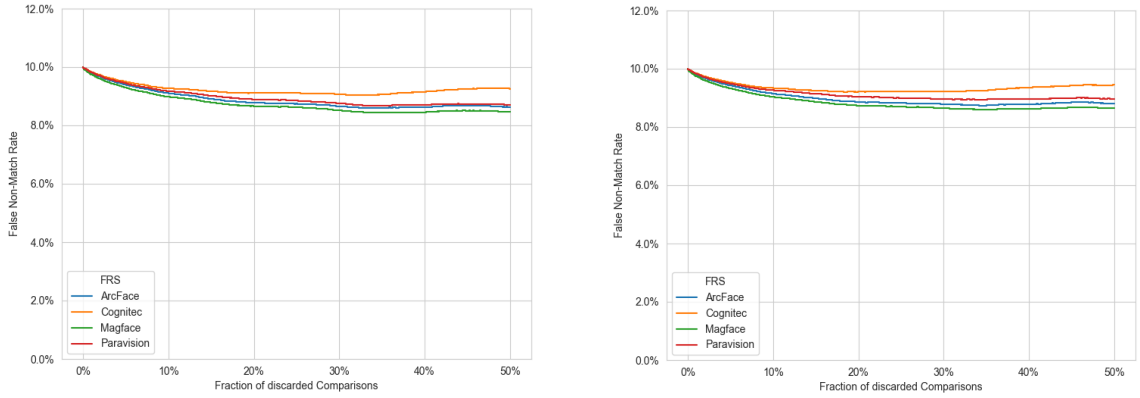

Figure 88: EDC curves for the quality component Mouth Closed using the CD1 algorithm (left) and the WD4 algorithm (right) on the VGGFace2 test set.

The WD4 algorithm using dlib landmarks was included in the submissions secunet_001 and secunet_002 to NIST FATE Quality SIDD track [16], while both the WD4 algorithm and the CD1 algorithm, using ADNet landmarks, were inlcuded in the submission secunet_003.

The WD4 algorithms were implemented for the SIDD component Mouth Open, which measures the ratio between the lips' aperture and the IED (as in the WD4 algorithm). As visible in Figure 89, the WD4 algorithm using ADNet landmarks shows a high accuracy, topped only by the two algorithms of Neurotechnology. As a general trend, the algorithm tends to over-estimate the mouth openness; a potential explanation could be small differences in the horizontal positions of the corresponding landmarks on upper and lower lip, resulting in a larger Euclidean distance than vertical distance. Presumably, an even better result could be achieved by applying a suitable calibration. In contrast, the WD4 algorithm with dlib landmarks shows a much lower accuracy.

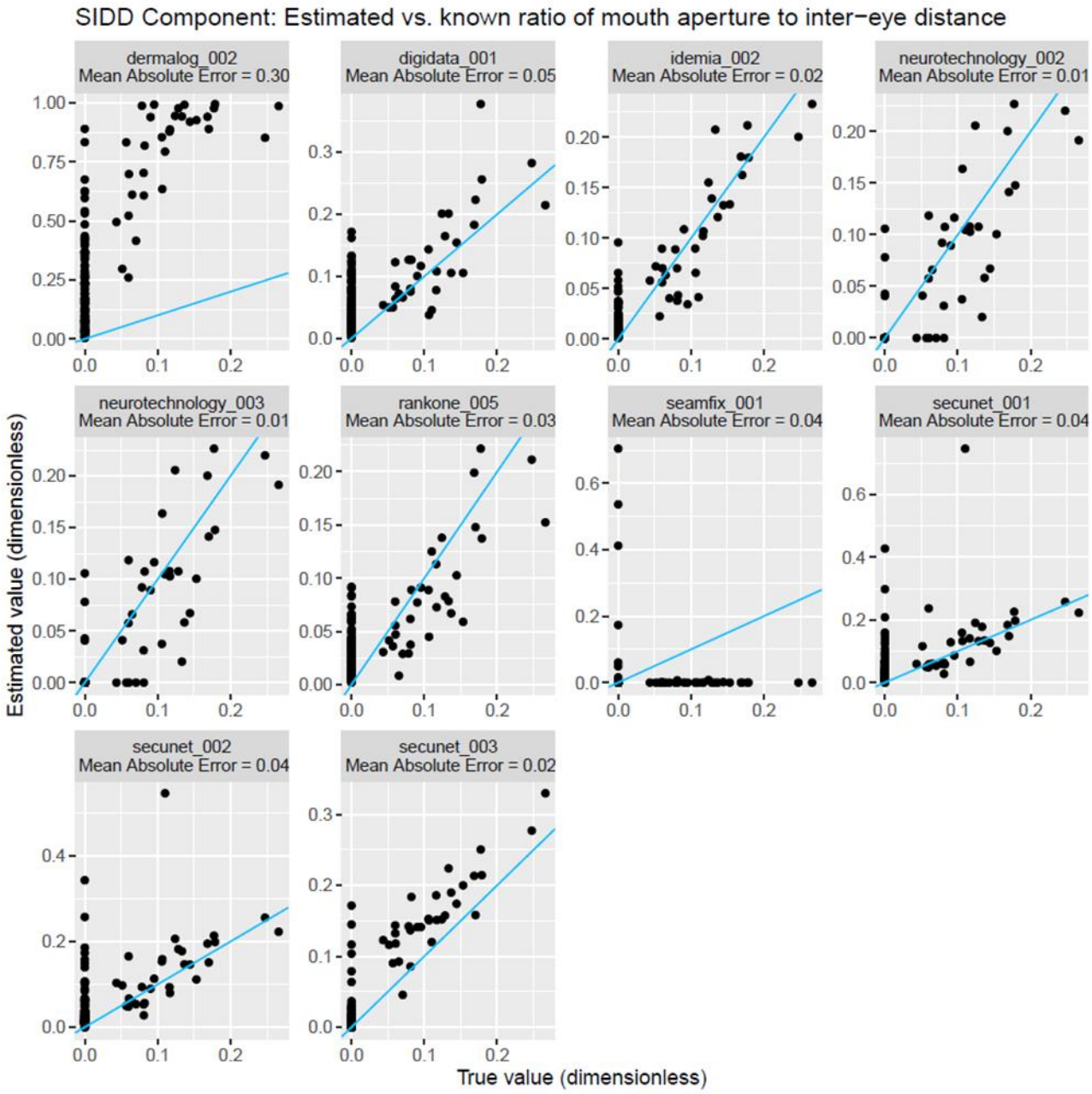

Figure 89: Scatter Plots and MAE of the outputs of the algorithms for the SIDD component Mouth Open vs. ground truth in the NIST FATE Quality SIDD report [16]. The WD4 algorithm using ADNet landmarks is labelled as secunet_003, while secunet_002 denoted the WD4 algorithm using dlib landmarks.

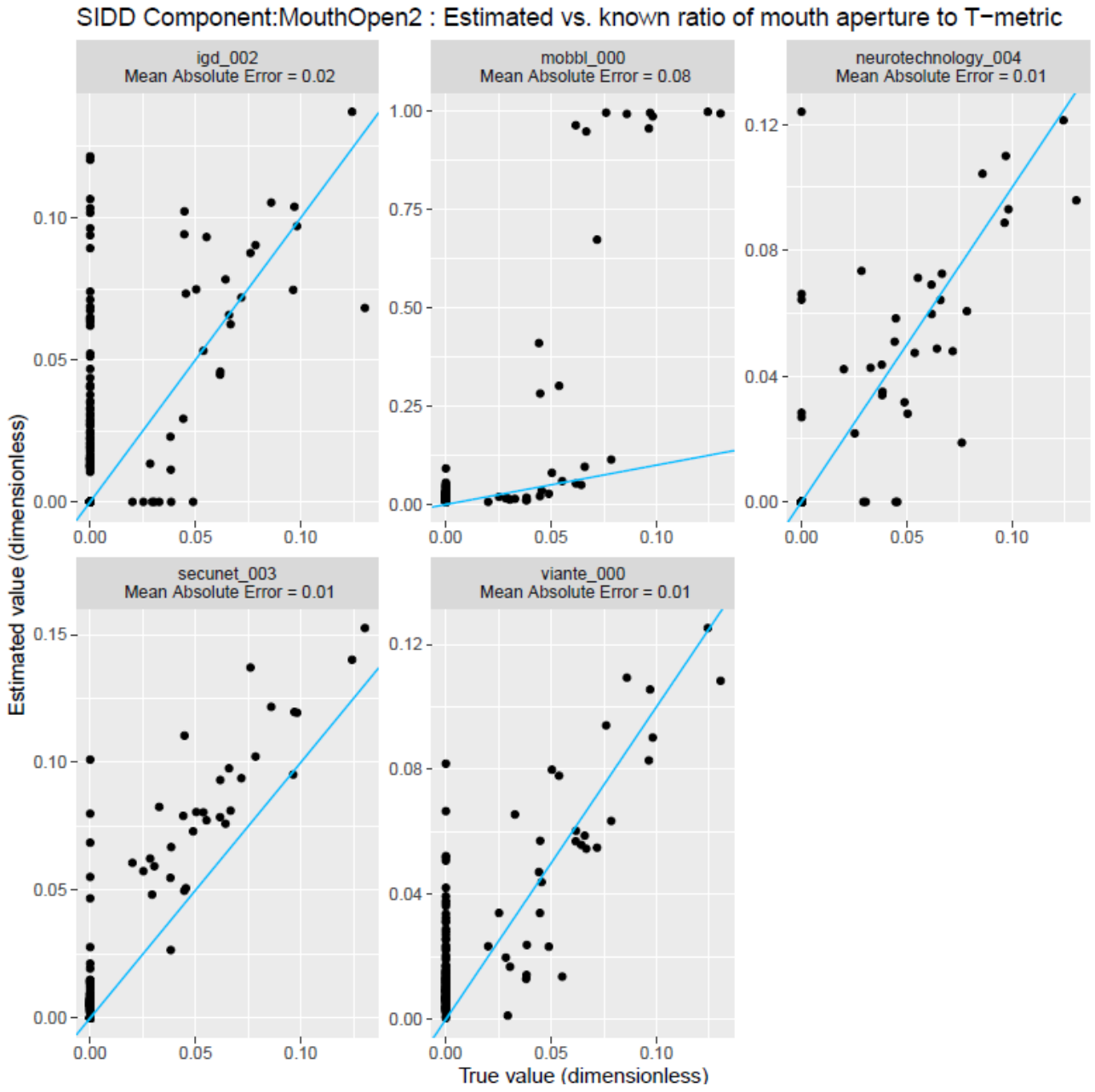

Figure 90: Scatter Plots and MAE of the outputs of the algorithms for the SIDD component Mouth Open 2 vs. ground truth in the NIST FATE Quality SIDD report [16]. The CD1 algorithm using ADNet landmarks is labelled as secunet_003.

The CD1 algorithm was implemented for the SIDD component Mouth Open 2, which measures the ratio between the mouth's aperture and the T-metric (distance between the eyes' midpoint and the chin, as computed in the CD1 algorithm). The CD1 algorithm using ADNet landmarks achieved the lowest MAE, on par with two competitors (neurotechnology_004 and viante_000). Similar to the WD4 algorithm, it tends to over-estimate the mouth openness. Presumably, an even better result could be achieved by applying a suitable calibration and could possibly be improved by a suitable calibration.

7.3.4 Final Algorithm Selection

For the quality component Mouth Closed, the CD1 algorithm specified in Section 7.3.2.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(100(1 - \text{SIGMOID}(q, 0.2, 0.06)))$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

7.4 Eyes Visible

7.4.1 Data Selection

The first test set was compiled from the Sejong Face Database [10], using binary labels indicating whether the eyes are occluded or not. An example image is depicted as part of Figure 91.

Figure 91: Example face image from Sejong Face database test set with visible (left) and occluded eyes (right).

As second test set, the test set partition of The Caltech Occluded Face in the Wild (COFW) database [55] (500 images) is used as test set. For these images, numeric labels were computed specifying the percentage of the Eyes Visibility Zone (EVZ) according to ISO/IEC 39794-5:2019 [1] that is un-occluded; the labels were computed by combining the occlusion maps for the COFW test set contained in the FaceOcc dataset [25] with the EVZ region estimated from the landmarks output by ADNet (Section 4.2.1.3) with the the method of the algorithm in Section 7.4.2.2. Example images of COFW along with their occlusion segmentation maps are shown in Figure 102.

The size and label distributions of the test sets for Eyes Visible are given in Table 14

Table 14: Number of images per label in test sets used for Eyes Visible.

Test Set	Eyes visible	Eyes occluded	Numerical labels
Sejong Face Database	2,816	1,542	-
COFW	-	-	500

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.4.2 Selection and Prototyping of Candidate Algorithms

7.4.2.1 CD1 Algorithm

The first candidate algorithm combines the face occlusion segmentation with a segmentation of the *Eyes Visibility Zone (EVZ)* as specified in ISO/IEC 39794-5:2019 [1] by landmarks (Figure 92). It takes as input the transformed landmarks L' output by the alignment algorithm in Section 4.4.1 when taking as input the landmarks computed with the ADNet landmark estimation algorithm (Section 4.2.1.3), the map S output by the face occlusion segmentation algorithm in Section 4.3.3.1, and the yaw angle α output by one of the algorithms in Section 7.10.2, and performs the following steps:

1. Using the landmarks L' and the yaw angle α, compute the Eyes Visibility Zone as follows:
    a. Compute the inter-eye distance D_{IED} as in the algorithm in Section 7.7.2.

b. Compute the offset distance V of the Eyes Visibility Zone as $V = D_{\mathrm{IED}}/20$.

c. Using the OpenCV function minAreaRect, compute the minimum (not necessarily upright) rectangle R1 containing the landmarks $L'60$, …, $L'67$ marking the contour of the subject's right eye.

d. Expand both the width and height of R1 by 2V.

e. Using the OpenCV function minAreaRect, compute the minimum (not necessarily upright) rectangle R2 containing the landmarks $L'68$, …, $L'75$ marking the contour of the subject's left eye.

f. Expand both the width and height of R2 by 2V.

g. Compute the joined Eyes Visibility Zone E as the union of R1 and R2, i.e. E = R1∩ R2.

2. Compute the occluded region U of the Eyes Visibility Zone as the intersection $U = O \cap E$ of the region $O = \{(x, y) | S(x, y) = 0\}$ marked as occluded in S and the Eyes Visibility Zone E.

3. Compute the proportion P of occlusion of the Eyes Visibility Zone as the fraction of the area of U and the complete area of E, i.e. $P = |U|/|E|$, where $|R|$ denotes the number of pixels of an image region R.

4. Output P.

The resulting algorithm was specified in ISO/IEC CD1 29794-5 [6].

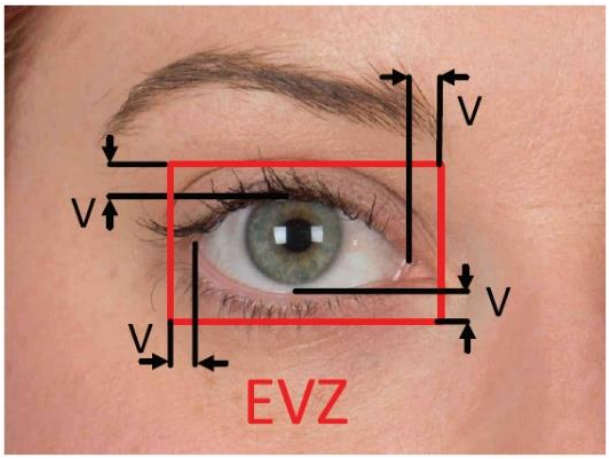

Figure 92: Definition of the Eyes Visibility Zone (EVZ) in ISO/IEC 39794-5:2019

7.4.2.2 CD3 Algorithm

The CD1 algorithm was improved as follows: In step 2 and 4, the minimum upright rectangle was computed (using the OpenCV function boundingRect) instead of the minimum rectangle with arbitrary orientation. The motivations of this change are:

- The visualisation of the Eyes Visibility Zone in ISO/IEC 39794-5:2019 [1] (see Figure 92) seems to imply an upright rectangle.
- Due to the alignment, the eyes are, typically, horizontally aligned so that, typically, an upright rectangle fits well to its shape.
- Experiments have shown that the minimum upright rectangle is more stable with respect to small changes to the landmark positions as compared to the minimum (potentially rotated) rectangle.

The improved algorithm takes as input the transformed landmarks L' output by the alignment algorithm in Section 4.4.1 when taking as input the landmarks computed with the ADNet landmark estimation algorithm (Section 4.2.1.3), the map S output by the face occlusion segmentation algorithm in Section 4.3.3.1, and the yaw angle α output by one of the algorithms in Section 7.10.2, and performs the following steps:

1. Using the landmarks L' and the yaw angle α, compute the Eyes Visibility Zone as follows:
   a. Compute the inter-eye distance D_{IED} as in the algorithm in Section 7.7.2.
   b. Compute the offset distance V of the Eyes Visibility Zone as $V = D_{\mathrm{IED}}/20$.
   c. Using the OpenCV function boundingRect, compute the minimum upright rectangle R1 containing the landmarks $L'60$, …, $L'67$ marking the contour of the subject's right eye.
   d. Expand both the width and height of R1 by 2V, i.e., extend R1 on all sides by V pixels.
   e. Using the OpenCV function boundingRect, compute the minimum upright rectangle R2 containing the landmarks $L'68$, …, $L'75$ marking the contour of the subject's left eye.
   f. Expand both the width and height of R2 by 2V, i.e., extend R2 on all sides by V pixels.
   g. Compute the Eyes Visibility Zone E as the union of R1 and R2, i.e. $E = R_1 \cup R_2$.
2. Compute the occluded region U of the Eyes Visibility Zone as the intersection $U = O \cap E$ of the region $O = \{(x, y) | S(x, y) = 0\}$ marked as occluded in S and the Eyes Visibility Zone E.
3. Compute the proportion P of occlusion of the Eyes Visibility Zone as the fraction of the area of U and the area of E, i.e. $P = |U|/|E|$, where $|R|$ denotes the number of pixels of an image region R.
4. Output P.

This algorithm is specified in ISO/IEC CD3 29794-5 [8].

7.4.2.3 Algorithm based on Occlusion Estimation CNN

An alternative algorithm was implemented based on a CNN trained to jointly infer the individual degrees of occlusion of

- the Eyes Visibility Zone according to ISO/IEC 39794-5:2019,
- the mouth region and
- the face region (up to the eye brows)

directly from the face image. The design, the training and an evaluation of the CNN (using numeric labels for the occlusion degree) is documented in [56] (in German). The algorithm for Eyes Visible based on this CNN takes as input the aligned face image I output by the algorithm in Section 4.4.1 in RGB colour channel order and performs the following steps:

1. Resize I to size 449x449 using bilinear interpolation.
2. Crop I by 61 pixels from the top, by 73 pixels from the bottom, by 76 pixels from the left and by 73 pixels from the right, to obtain an image I of size 300x300.
3. Divide I by 255 and normalise the result using mean $\mu = (0.485, 0.456, 0.406)$ and standard deviation $\sigma = 0.229, 0.224, 0.225$ to obtain array A, i.e. set
   $$A_{i,j,k} = (I_{i,j,k}/255 - \mu_k)/\sigma_k$$
   for all i, j, k.
4. Encode A as input tensor T of dimensions (1, 3, 300, 300) for the neural network model.
5. Run a forward pass through the CNN model using T as input to obtain an output vector $(x_{\mathrm{e}}, x_{\mathrm{m}}, x_{\mathrm{f}})$, representing the predicted occlusions (in percent) of eyes, mouth and face, respectively.
6. Output $\min(0, \max(\lfloor x_{\mathrm{e}} \rceil, 100))$, where the rounding function $\lfloor x \rceil$ is defined as $\lfloor x + 0.5 \rfloor$ for positive x and as $\lceil x - 0.5 \rceil$ for negative x.

7.4.3 Evaluation

The DET curves on the Sejong Database test set and the ECDF curves on the COFW test set of the algorithms for the Eyes Visible quality component are shown in Figure 93. On the Sejong Database test set, the error rates of the CD1 algorithm and the CD3 algorithm are comparable with an EER of 1.7% each, and are better than the error rates of the algorithm based on the occlusion estimation CNN (EER of 2.5%).

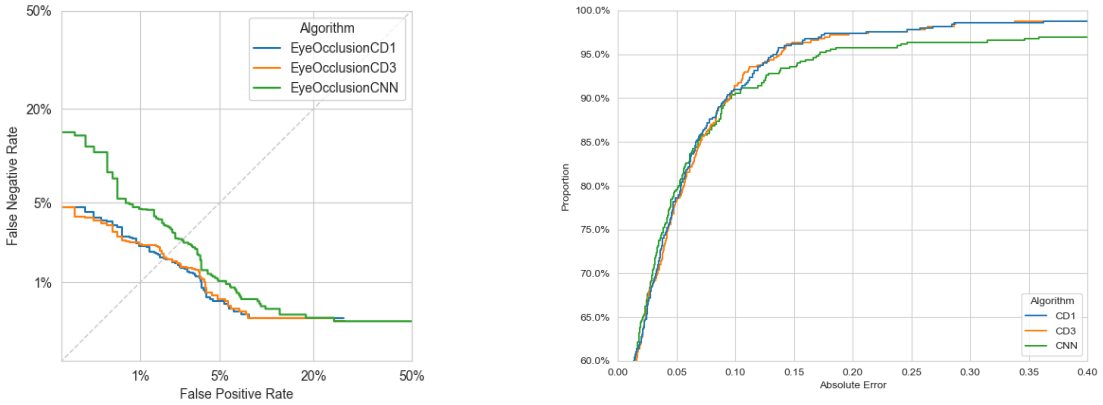

Figure 93: DET curves of the algorithms for the quality component Eyes Visible on the Sejong Face Database test set (left) and the corresponding ECDF curves on the COFW test set (right).

On the COFW test set, up to an absolute error of 5%, the algorithm based on the Occlusion Estimation CNN shows a slightly better accuracy, but for more difficult images, the algorithms from CD1 and CD3 perform much better. The ECDF curves of the CD1 algorithm and the CD3 algorithm are very similar. The MAE of both the CD1 algorithm and the CD3 algorithm (0.034 each) are not only lower than that of the algorithm based on the Occlusion Estimation CNN (0.046).

However, it must be noted that the computation of the labels of the COFW test set (percentage of occlusion of the EVZ) was based on the method to estimate the EVZ in the CD3 algorithm. Thus, the occlusion estimation of the CD3 algorithm deviates from the computation of the labels of the COFW test set only in how the face occlusion segmentation is obtained.

The EDC curves of the algorithms for the Eyes Visible quality component are shown in Figure 94 and Figure 95. For all algorithms, a clear decline of FNMR is observed. For the CD1 and CD3 algorithms, the curves are very similar while, for the algorithm based on the occlusion estimation CNN, the curve is less steep.

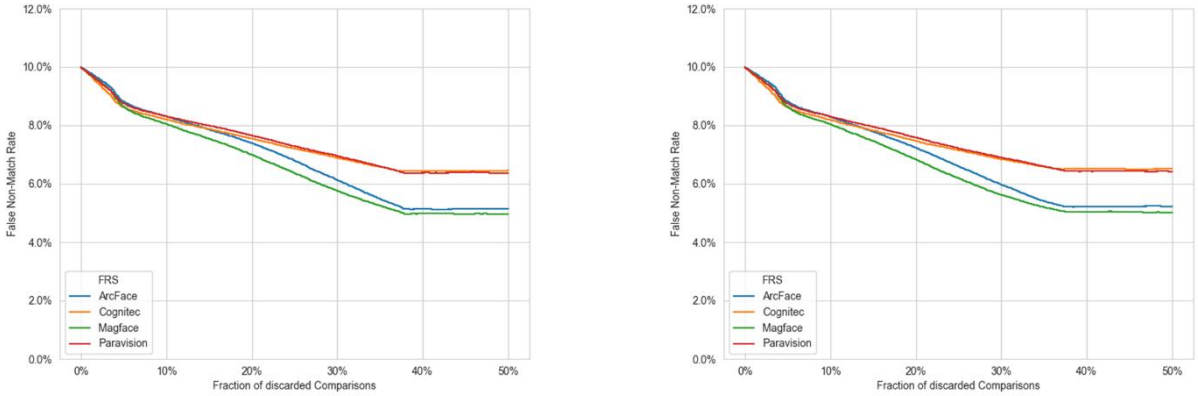

Figure 94: EDC curves for the quality component Eyes Visible using the CD1 algorithm (left) and the CD3 algorithm (right) on the VGGFace2 test set.

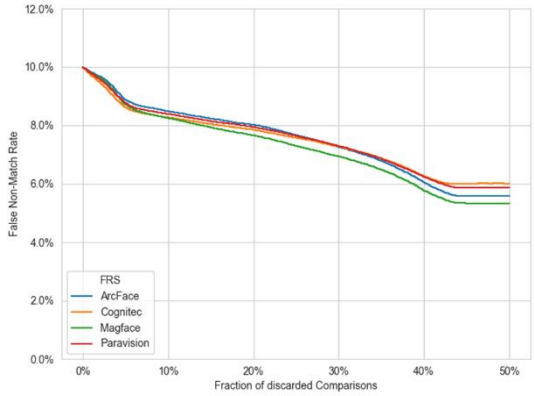

Figure 95: EDC curves for the quality component Eyes Visible using the algorithm based on the Occlusion Estimation CNN on the VGGFace2 test set.

7.4.4 Final Algorithm Selection

For the quality component Eyes Visible, the CD3 algorithm specified in Section 7.4.2.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(100\ (1 - q))$$

is used, where ROUND is defined as described in Section 5.4.

7.5 Mouth Occlusion Prevention

7.5.1 Data Selection

The first test set was compiled from the Sejong Face Database [10], using binary labels indicating whether the mouth is occluded or not. Example images are shown in Figure 96

Figure 96: Example face image from Sejong Face Database test set with visible (left) and occluded mouth (right).

As second test set, the test set partition of The Caltech Occluded Face in the Wild (COFW) database [55] (500 images) is used as test set. For theses images, numeric labels were computed specifying the percentage of the mouth area that is un-occluded; the labels were computed by combining the occlusion maps for the COFW test set contained in the FaceOcc dataset [25] with the mouth region, where the mouth region was computed as the convex hull of the landmarks marking the lips computed with ADNet (Section 4.2.1.3). Example images of COFW along with their occlusion segmentation maps are shown in Figure 102.

The size and label distributions of the test sets for Eyes Visible are given in Table 15.

Table 15: Number of images per label in test sets used for Mouth Occlusion Prevention.

Test set	Mouth visible	Mouth occluded	Numerical Labels
Sejong Face Database	2,884	1,474	-
COFW	-	-	500

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.5.2 Selection and Prototyping of Candidate Algorithms

7.5.2.1 CD1 Algorithm

The first candidate algorithm combines the face occlusion segmentation with a segmentation of the mouth by landmarks. It takes as input the transformed landmarks L' output by the alignment algorithm in Section 4.4.1 when taking as input the landmarks computed with the ADNet landmark estimation algorithm (Section 4.2.1.3) and the face occlusion segmentation map S of the image, and performs the following steps:

1. Determine the mouth region M as the region enclosed by the landmarks L'_{76}, …, L'_{87} marking the contour of the mouth.

2. Compute the occluded mouth region U as the intersection $U = O \cap M$ of the region $O = \{(x, y)|S(x, y) = 0\}$ marked as occluded in S and the mouth region M.
3. Compute the proportion P of occlusion of the mouth region as the fraction of the area of the occluded mouth region and the complete area of the mouth region, i.e. $P = |U|/|M|$, where $|R|$ denotes the number of pixels of an image region R.
4. Output P.

This algorithm is also specified in ISO/IEC CD1 29794-5 [6].

The CD1 algorithm was evaluated using the improved face occlusion segmentation algorithm from Section 4.3.1.3 and by using ADNet for landmark estimation (Section 4.2.1.3).

7.5.2.2 Algorithm based on Occlusion Estimation CNN

Furthermore, an algorithm based on the occlusion estimation CNN was implemented. The algorithm is identical to that described in Section 7.4.2.3 except that it outputs $\min(0, \max(\lfloor x_m \rceil, 100))$ in the last step, where the rounding function $\lfloor x \rceil$ is defined as $\lfloor x + 0.5 \rfloor$ for positive x and as $\lceil x - 0.5 \rceil$ for negative x.

7.5.3 Evaluation

The DET curves on the Sejong Database test set and the ECDF curves on the COFW test set of the algorithms are plotted in Figure 97. On the Sejong Database test set, the algorithm based on the occlusion estimation CNN achieves much better error rates with an EER of 3.7%.

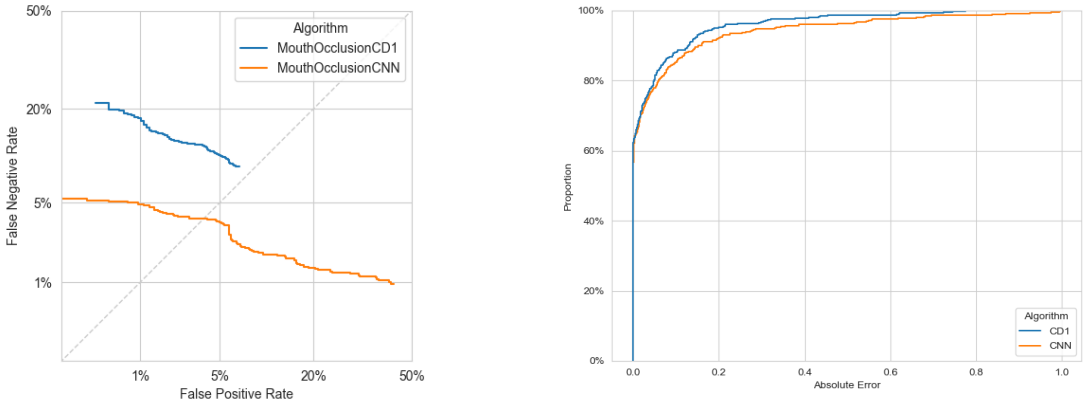

Figure 97: DET curves on the Sejong Database test set (left) and ECDF curves on the COFW test set (right) of the algorithms for the quality component Mouth Occlusion Prevention.

The error rates of the algorithms are still relatively high. The reason for this is that the occlusion-aware segmentation still sometimes incorrectly labels the mouth region within dark artificial beards as completely occluded, as shown in Figure 98.[37].

[37] However, in some of these images, the mouth is actually partially occluded by the artificial beard, which is not reflected by the labels.

The ECDF curves on the COFW test set show that the MAE of the CD1 algorithm (0.039) is considerably lower than that of the algorithm based on the Occlusion Estimation CNN (0.057).

Figure 98: Incorrect labelling of the mouth region within dark artificial beards by the occlusion-aware segmentation.

The EDC curves of the algorithms for No Occlusion of the Mouth on the VGGFace2 test set are shown in Figure 99. The EDC curves of the CD1 algorithm decline only up to a discard rate of 11%, which can be explained by the fact that images with occlusions of the mouth only account for approximately 11% of the comparisons.

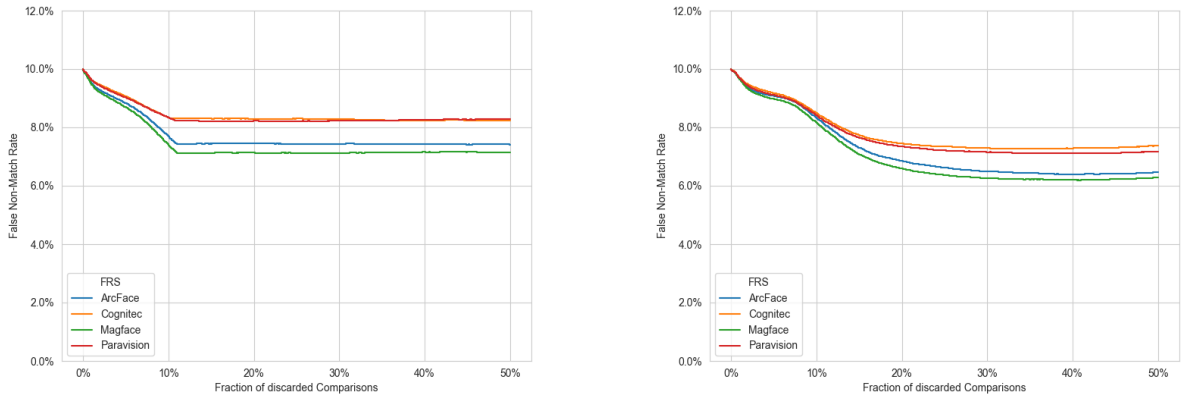

Figure 99: EDC curves for the quality component Mouth Occlusion Prevention using the CD1 algorithm (left) and the algorithm based on the Occlusion Estimation CNN (right) on the VGGFace2 test set.

Surprisingly, the algorithm based on the Occlusion Estimation CNN shows a decline even for higher discard rates. However, a visual inspection of the images to which the Occlusion Estimation CNN based algorithm assigns the lowest scores while the CD1 algorithm doesn't detect any occlusion reveals that these images mainly belong to one of the following classes:

- Images of painted or sketched faces
- Images, where watermarks are visible in the mouth area
- Images of very low resolution

While discarding such images certainly reduces the FNMR, it is not the aim of the algorithm.

7.5.4 Final Algorithm Selection

For the quality component Mouth Occlusion Prevention, the CD1 algorithm specified in Section 7.5.2.1 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(100\ (1 - q))$$

is used, where ROUND is defined as described in Section 5.4.

7.6 Face Occlusion Prevention

7.6.1 Data Selection

7.6.1.1 Training Set

As training set, CelebAMask-HQ [24] was used, which contains 30,000 high-definition images (upscaled using a super-resolution CNN) and with semantic segmentation maps for all facial components and accessories such as face skin, nose, eyes, eyebrows, ears, mouth, lips, hair, head covering, eyeglasses, earring, necklace, neck, and clothes. However, the segmentation maps do not mark occlusions, i.e. occluded parts of the face are not excluded from the skin segmentation map. An example is shown in Figure 100.

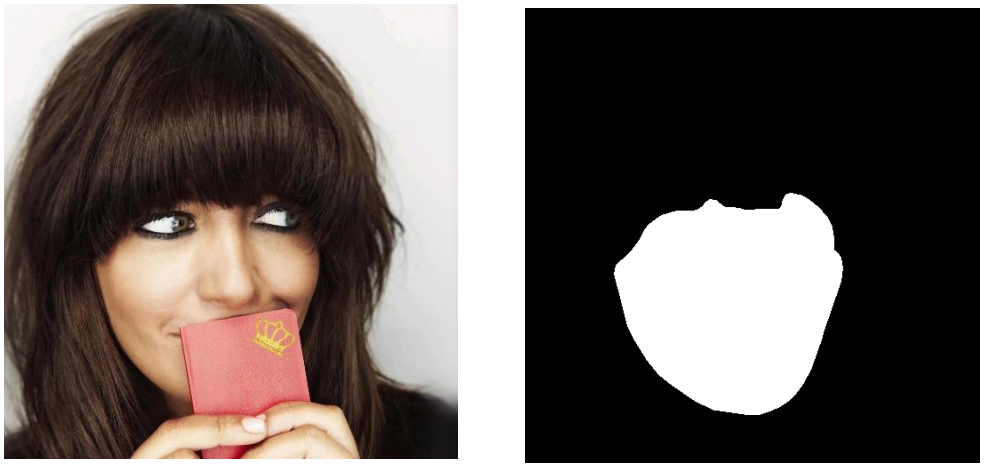

Figure 100: Image with occlusion of the face and corresponding segmentation map for "skin". The occluded parts are not excluded and can, thus, not be determined from the segmentation map.

Fortunately, the dataset FaceOcc [25] contains face occlusion maps for CelebAMask-HQ (30,000 images).

The FaceOcc occlusion masks are based on the following definition of what is considered as occlusion, partly documented in [25]:

- Lenses of glasses are only considered as occlusion if they are either opaque or if they are coloured and their colour differs considerably from the skin colour
- Frames of glasses as well as the shadows cast by them onto the face are considered as occlusion.
- Saturated reflections on lenses of glasses are considered as occlusions.
- Facial hair (beards, moustaches, eye brows) are not considered as occlusion
- A tongue poked out of the mouth is considered as occlusion
- Makeup is only considered as occlusion if there is a clearly visible border to face skin

Figure 101: Aligned image from CelebAMask-HQ, its aligned face parsing map for „hat“ from CelebAMask-HQ, and its occlusion mask from FaceOcc.

However, in the FaceOcc segmentation maps for CelebAMask-HQ (encoded by the image's alpha channel) the labelled occlusions don't contain hairs or head coverings (e.g. see Figure 101). Therefore, in order to compute the labels for the face occlusion, i.e. the percentage of the face area that is occluded, for CelebAMask-HQ, we combined the FaceOcc occlusion maps with the regions labelled as "hair" or "hat" (head coverings) in the face parsing segmentation maps of CelebAMask-HQ to obtain complete face occlusion maps. Since the FaceOcc masks for the CelebAMask-HQ images have been determined after alignment of the images (using landmarks computed with 3DDFAv2), we aligned the face parsing segmentation maps of CelebAMask-HQ in the same way.

Using these combined occlusion segmentation masks for CelebAMask-HQ, the occlusion labels were set to the percentage of the landmarked region (Section 4.3.1.1) with parameter $\alpha=0$ computed by ADNet landmarks (Section 4.2.1.3) that is marked as un-occluded.

7.6.1.2 Test Set

As test set, the test set partition of The Caltech Occluded Face in the Wild (COFW) database [55] (500 images) is used as test set. For these images, numeric labels were computed specifying the percentage of the face region that is un-occluded; the labels were computed by combining the occlusion maps for the COFW test set contained in the FaceOcc dataset [25] (which already cover occlusions by hair and hats) with the landmarked region (Section 4.3.1.1) with parameter $\alpha=0$ computed with ADNet landmarks (Section 4.2.1.3). Example images of COFW along with their occlusion segmentation maps are shown in Figure 102.

Figure 102: Example face images from COFW with corresponding occlusion segmentation map.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.6.2 Selection and Prototyping of Candidate Algorithms

7.6.2.1 CD1 Algorithm

The first candidate algorithm combines the face occlusion segmentation with the landmarked region segmentation.

The algorithm takes as input the aligned image I and the transformed landmarks L' output by the alignment algorithm in Section 4.4.1 when taking as input the landmarks computed with the ADNet landmark estimation algorithm (Section 4.2.1.3), and the map S computed by the face occlusion segmentation algorithm specified in Section 4.3.3.1, and performs the following steps:

1. From the transformed landmarks, compute the landmarked region segmentation map R using the algorithm specified in Section 4.3.1.1.
2. Compute the occluded face region U as the intersection $U = O \cap R$ of the region $O = \{(x, y) | S(x, y) = 0\}$ marked as occluded in S and the landmarked region R.
3. Compute the proportion P of occlusion of the face as the fraction of the area of the occluded face region and the area of the landmarked region, *i.e.* $P = |U|/|R|$, where $|A|$ denotes the number of pixels of an image region A.
4. Output P.

The output depends on the selection of the parameter α used by the landmarked region segmentation which defines to which extent the forehead is included in the considered face region. When using $\alpha = 0$, this algorithm is equivalent to that specified in ISO/IEC CD2 29794-5:2023 [7]. In contrast for the NIST FATE Quality SIDD track, $\alpha = 0.85$ is used to comply with the definition of the ground truth data.

7.6.2.2 Algorithm based on Occlusion Estimation CNN

Furthermore, an algorithm based on the occlusion estimation CNN was implemented. The algorithm is identical to that described in Section 7.4.2.3 except that it outputs $\min(0, \max(\lfloor x_f \rceil, 100))$ in the last step, where the rounding function $\lfloor x \rceil$ is defined as $\lfloor x + 0.5 \rfloor$ for positive x and as $\lceil x - 0.5 \rceil$ for negative x.

7.6.3 Evaluation

The ECDF curves for both algorithms on the COFW test set are plotted in Figure 103. It can be observed that the CD1 algorithm achieves a significantly better detection performance (MAE of 0. 028) as compared to the algorithm based on the Occlusion Estimation CNN (MAE of 0.038).

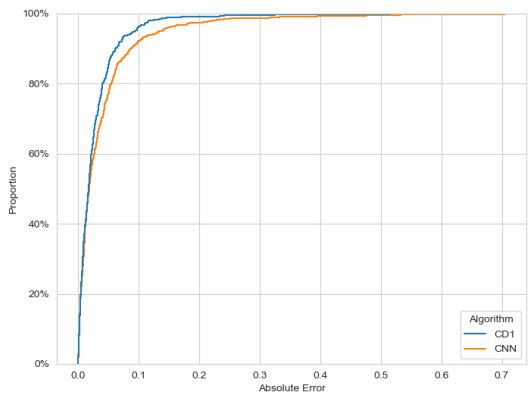

Figure 103: ECDF curves of the algorithms for the quality component Face Occlusion Prevention on the COFW test set.

The EDC curves of the algorithms for face occlusion on the VGGFace2 test set using SSD and ADNet for face detection and landmark estimation are shown in Figure 104. For both algorithms, a clear drop of FNMR is observed, with a stronger decline for the CD1 algorithm.

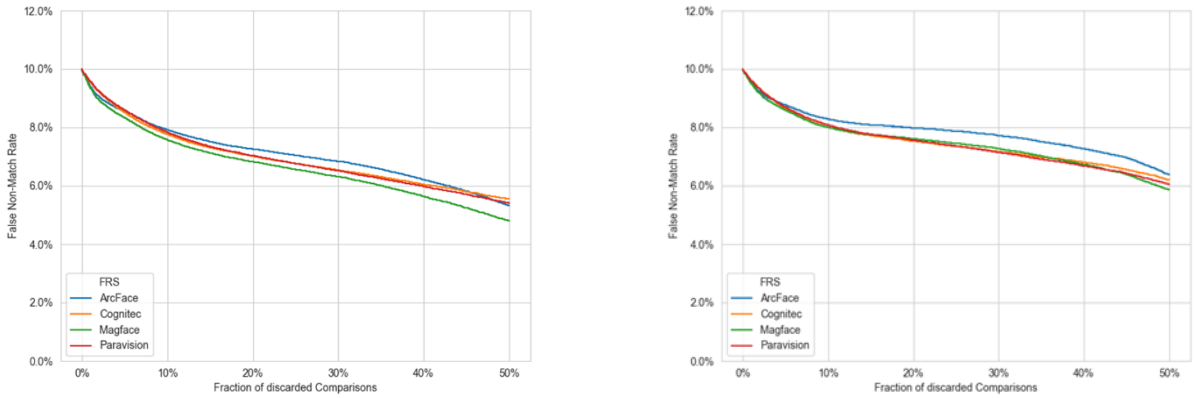

Figure 104: EDC curves for the quality component Face Occlusion Prevention using the CD1 algorithm (left) and the algorithm based on the occlusion estimation CNN (right) on the VGGFace2 test set.

The CD1 algorithm and the algorithm based on the occlusion estimation CNN were included in the submissions secunet_003 and secunet_004 to NIST FATE Quality SIDD track [16]. There, the CD1 algorithm was evaluated for the quality measures Face Occlusion and Face Occlusion 2, while the algorithm based on the occlusion estimation CNN was only evaluated for Face Occlusion 2; the difference of these two measures is the underlying definition of the face region, which, in Face Occlusion, includes the forehead, in contrast to the definition for Face Occlusion 2, which excludes the forehead like in CD3 and in the training process of the occlusion estimation CNN. In order to allow evaluation of the CD1 algorithm for both measures, it was adapted to use a landmarked region estimation with parameter $\alpha = 0.85$ (which implies inclusion of the forehead, see Figure 10) for the Face Occlusion measure and a landmarked region estimation with parameter $\alpha = 0$ (which doesn't include the forehead) for Face Occlusion 2.

In the evaluation of the measure Face Occlusion, the CD1 algorithm achieved the highest accuracy of all algorithms with a mean absolute error (MAE) of 0.04.

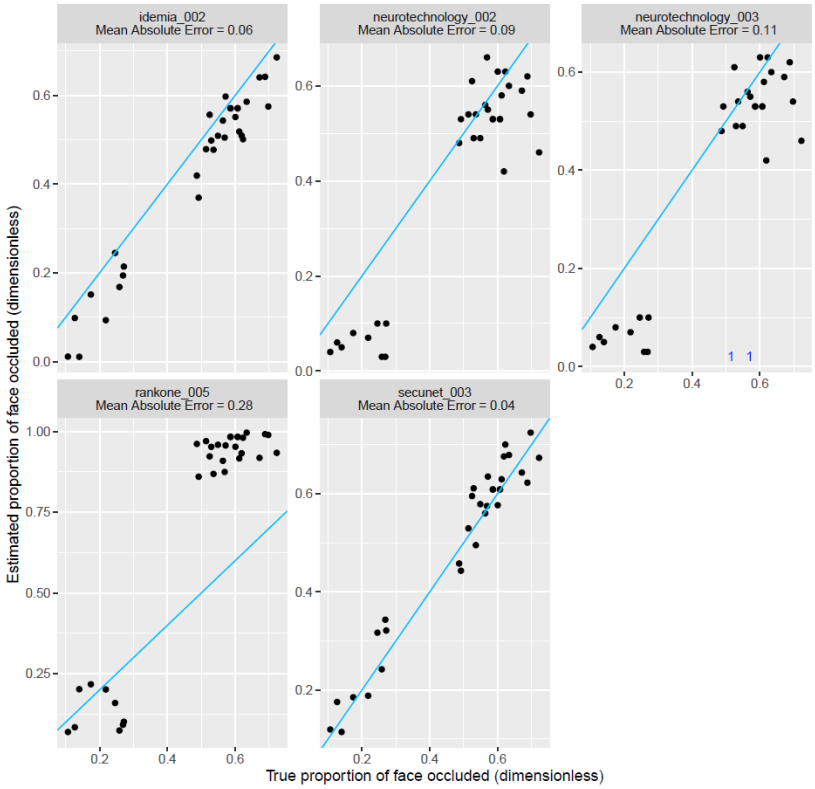

Figure 105: Scatter plots of prediction vs ground truth and MAE of the algorithms for the measure Face Occlusion in the NIST FATE Quality SIDD report [16]. The CD1 algorithm is labelled as secunet_003 (bottom right).

In the evaluation of Face Occlusion 2, both algorithms achieved the same MAE of 0.06 which is the highest accuracy of all submitted algorithms. The higher MAE as compared to the measure Face Occlusion can be explained by the smaller face area used in the denominator in the proportion of face occluded.

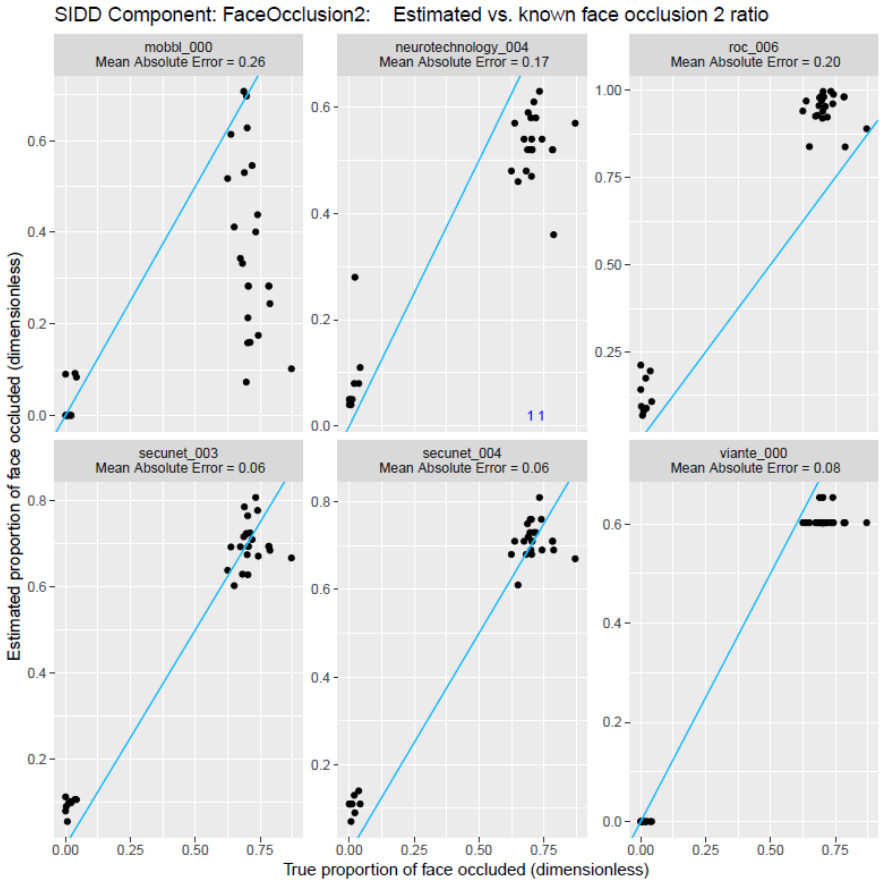

Figure 106: Scatter plots of prediction vs ground truth and MAE of the algorithms for the measure Face Occlusion 2 in the NIST FATE Quality SIDD report [16]. The CD1 algorithm is labelled as secunet_003 (middle), the algorithm based on the occlusion estimation CNN is labelled as secunet_004 (right).

7.6.4 Final Algorithm Selection

For the quality component Mouth Occlusion Prevention, the CD1 algorithm specified in Section 7.6.2.1 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(100\ (1 - q))$$

is used, where ROUND is defined as described in Section 5.4.

7.7 Inter-Eye Distance

7.7.1 Data Selection

Since the accuracy of the algorithm only depends on the accuracy of the computed landmarks, no evaluation with respect to ground truth data is conducted.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.7.2 Selection and Prototyping of Candidate Algorithms

The algorithm used to estimate the inter-eye distance D_{IED} is taken from ISO/IEC CD1 29794-5 [6] and accounts for perspective projections due to the head pose. It takes as input the landmarks output by one of the landmark estimation algorithms from Section 4.2.1 and the yaw angle α computed using one of the head algorithms in Section 7.10.2, and performs the following steps:

1. Compute the left and right eye centres as $(X_L, Y_L) = (L_a + L_b)/2$ and $(X_R, Y_R) = (L_c + L_d)/2$
   - For dlib, use a=36, b=39, c=42, d=45
   - For MediaPipe, use a=33, b=133, c=263, d=362
   - For ADNet, use a=60, b=64, c=68, d=72
2. Compute the inter-eye distance as $D_{\mathrm{IED}} = \sec(\alpha) \cdot \sqrt{(X_L - X_R)^2 + (Y_L - Y_R)^2}$, where $\sec(x) = 1/\cos(x)$.

7.7.3 Evaluation

The EDC curves of the algorithm using ADNet is depicted in Figure 107. Similar results were achieved for the same algorithm using dlib or MediaPipe for landmark estimation.

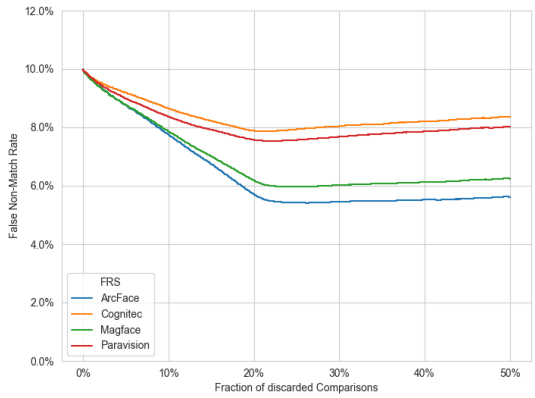

Figure 107: EDC curves for the quality component Inter-Eye Distance using the CD1 algorithm with ADNet landmarks on the VGGFace2 test set.

The implementation of the CD1 algorithm with ADNet landmarks and SynergyNet or 3DDFA_V2, respectively, were inlcuded in the submissions secunet_003 and secunet_004, respectively, to NIST FATE Quality SIDD track [16]. The algorithms were tested on three test sets, the first two showing generally frontal poses, and the third containing pairs of images, one being frontal and one with considerable yaw.

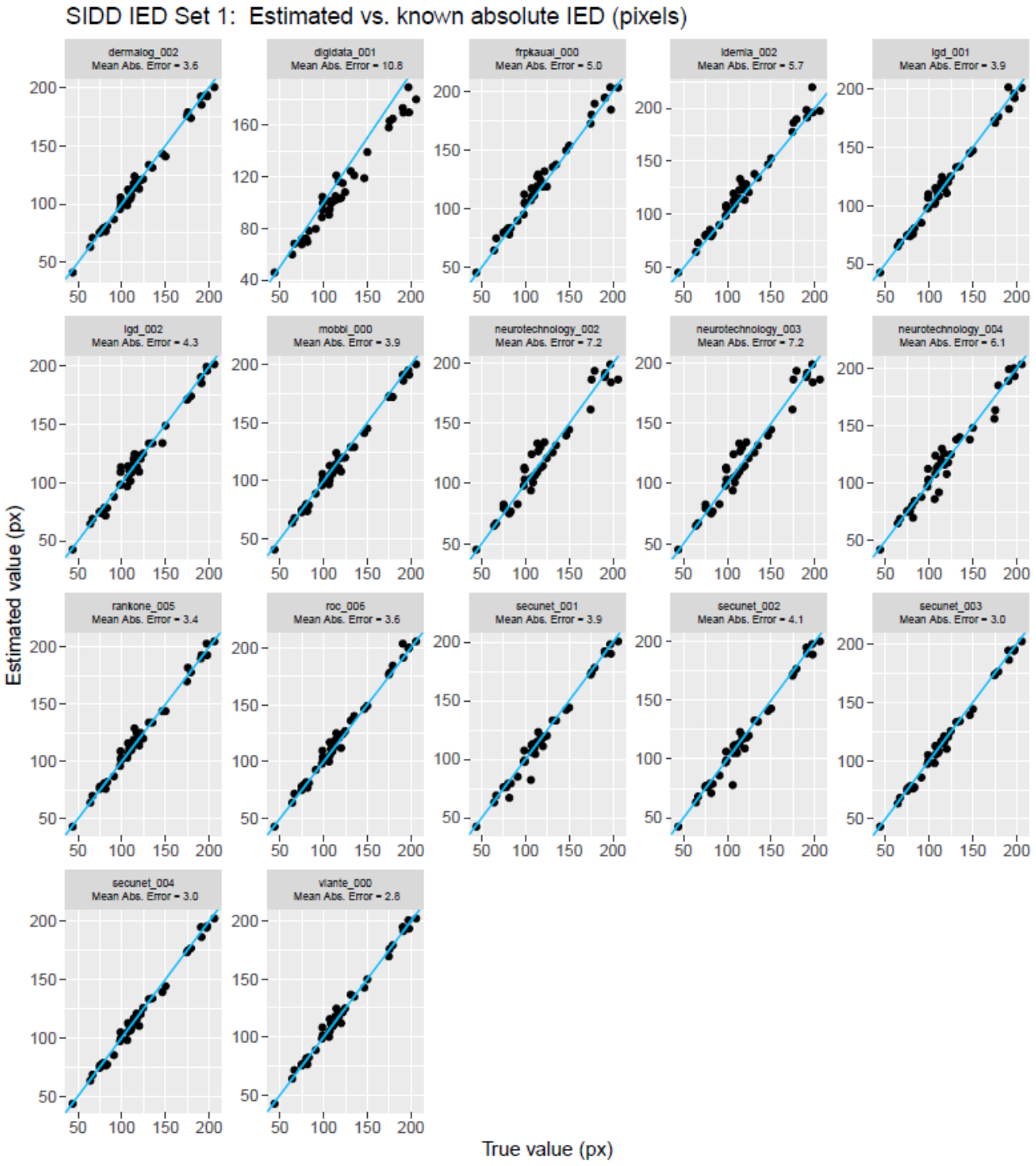

Figure 108: Scatter Plots and MAE of the outputs of the algorithms for the SIDD component IED vs. ground truth for test set 1 in the NIST FATE Quality SIDD report [16]. The CD1 algorithm using ADNet and SynergyNet is labelled as secunet_003, the CD1 algorithm using ADNet and 3DDFA_v2 is labelled as secunet_004, and the WD5 algorithm using dlib is labelled as secunet_001 and secunet_002 (same implementation).

On all three test sets, both implementations of the CD1 algorithm (with SynergyNet and 3DDFA_V2, respectively) show very high accuracy. The implementation with 3DDFA_V2 is slightly better than that with SynergyNet.

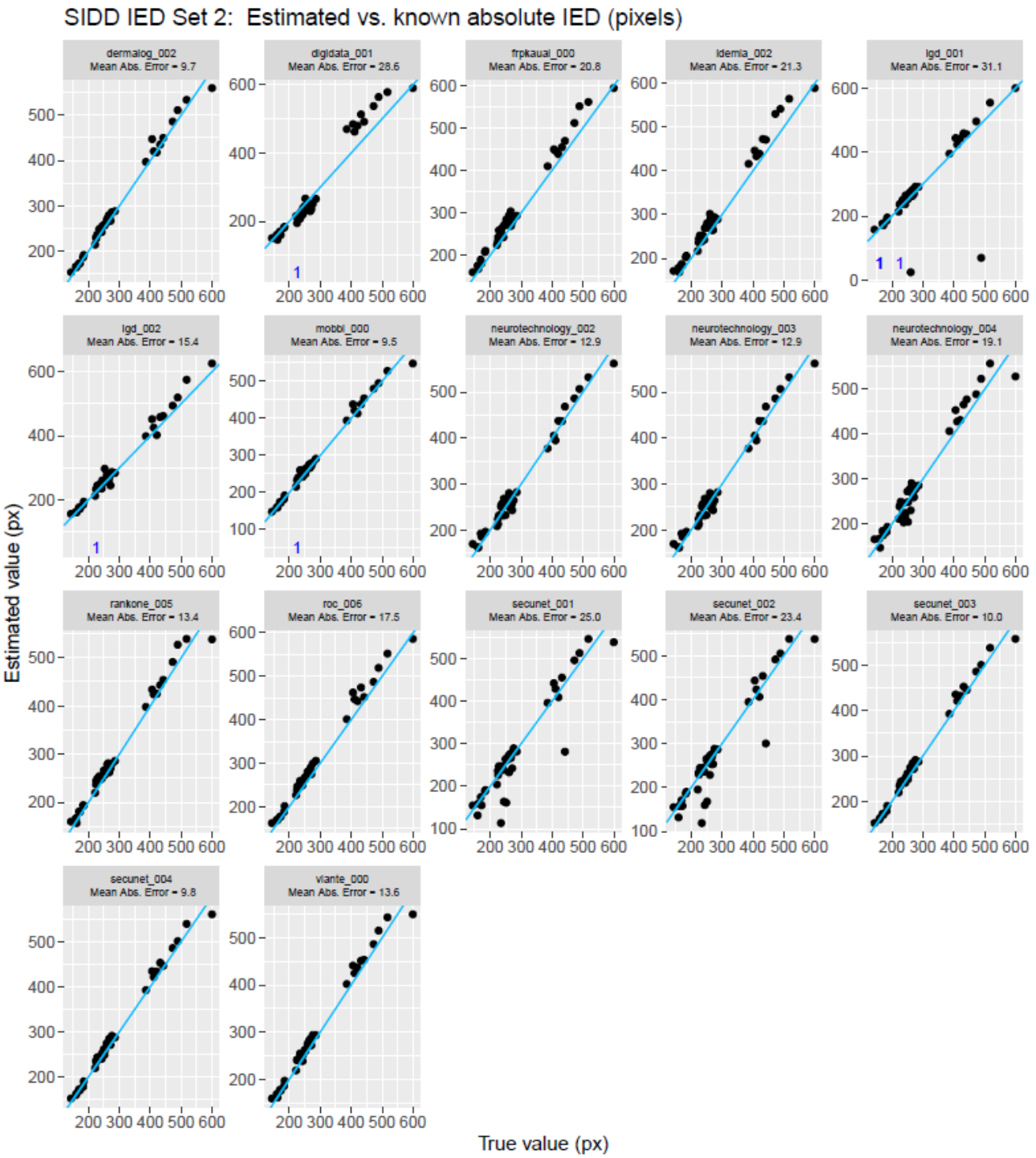

Figure 109: Scatter Plots and MAE of the outputs of the algorithms for the SIDD component IED vs. ground truth for test set 2 in the NIST FATE Quality SIDD report [16]. The CD1 algorithm using ADNet and SynergyNet is labelled as secunet_003, the CD1 algorithm using ADNet and 3DDFA_v2 is labelled as secunet_004, and the WD5 algorithm using dlib is labelled as secunet_001 and secunet_002 (same implementation).

For the third test set, NIST plotted the dependency of the "error" (defined as deviation of IED estimates between mated images) vs. the yaw angle. Again, the CD1 algorithms were among the most accurate algorithms submitted (rank 1 and 2), with 3DDFA_V2 giving slightly better results than SynergyNet. Algorithms not considering the yaw angle show a dependency of the deviation mean from the yaw.

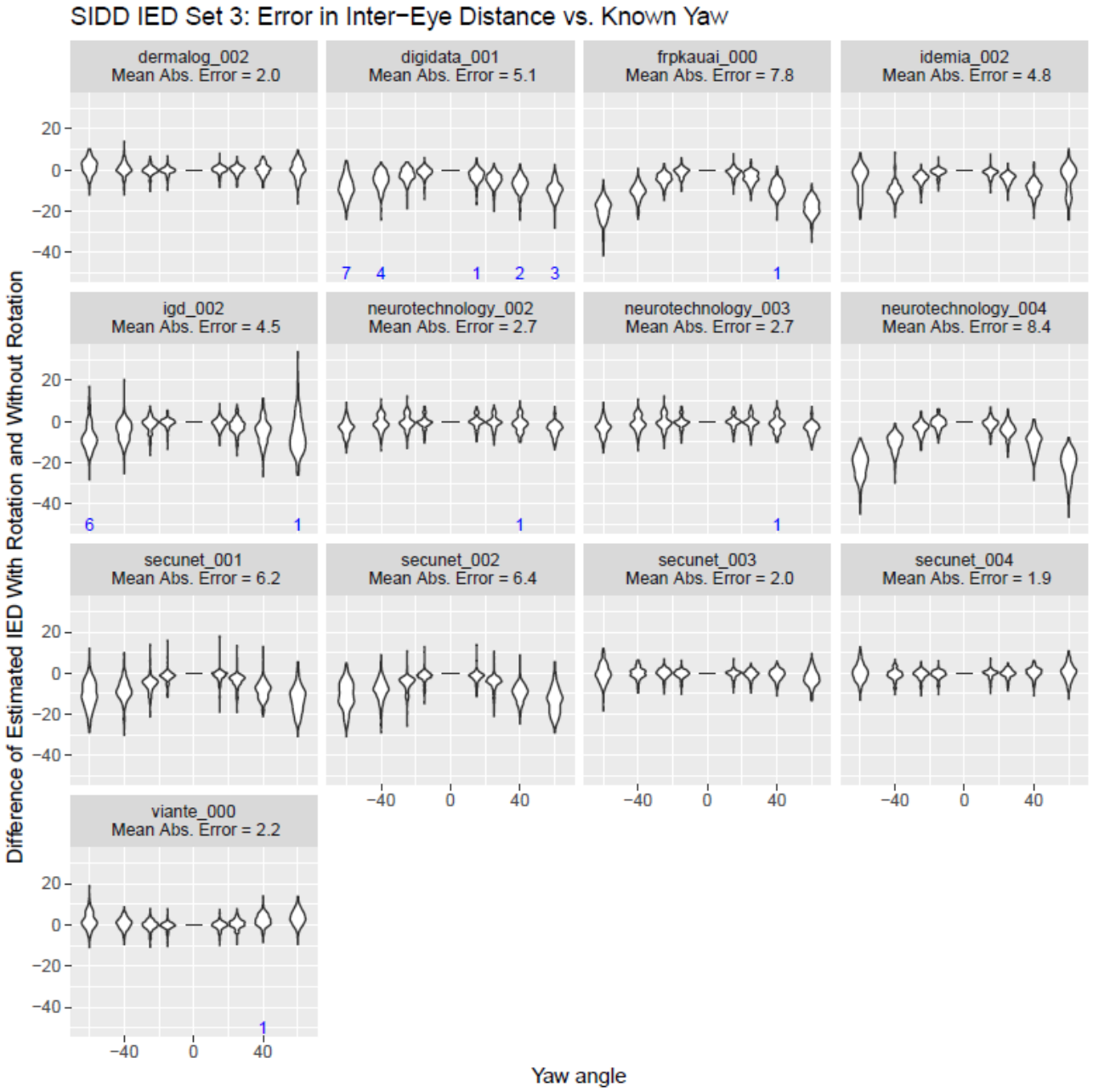

Figure 110: Deviation of IED between mating images vs. yaw angle and MAE for test set 3 of the SIDD quality component IED in the NIST FATE Quality SIDD report [16]. The CD1 algorithm using ADNet and SynergyNet is labelled as secunet_003, the CD1 algorithm using ADNet and 3DDFA_v2 is labelled as secunet_004, and the WD5 algorithm using dlib is labelled as secunet_001 and secunet_002 (same implementation).

7.7.4 Final Algorithm Selection

For the quality component Inter-Eye Distance, the CD1 algorithm specified in Section 7.7.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure D_{IED} to the quality component value Q in the target range [0,100], the function

$$Q = \mathrm{ROUND}(100\ \mathrm{SIGMOID}(D_{\mathrm{IED}}, 70, 20))$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

7.8 Head Size

7.8.1 Data Selection

No data was collected for implementation and evaluation.

7.8.2 Selection and Prototyping of Candidate Algorithms

The algorithm from ISO/IEC FDIS 29794-5:2024 [10] was implemented. It takes as input the landmarks L computed with the ADNet landmark estimation algorithm (Section 4.2.1.3) and the height B of the image, and performs the following steps:

1. Using the landmarks L, compute the distance T between the eye's midpoint and the chin as in the algorithm in Section 7.2.2.3.
2. Compute and output D = T/B, where B is the height of the image.

7.8.3 Evaluation

No evaluation of the algorithm for the quality component Single Face Present was performed.

7.8.4 Final Algorithm Selection

For the quality component Head Size, the FDIS algorithm specified in Section 7.9.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \text{ROUND}(200\ (1 - \text{SIGMOID}(|q - 0.45|, 0, 0.05))$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

7.9 Crop of the Face

7.9.1 Data Selection

No data was collected for implementation and evaluation.

7.9.2 Selection and Prototyping of Candidate Algorithms

The algorithms from ISO/IEC FDIS 29794-5:2024 [10] were implemented. Here, they are described as a common algorithm, that takes as input the landmarks L computed with the ADNet landmark estimation algorithm (Section 4.2.1.3) as well as the width A and the height B of the image, and performs the following steps:

1. Using the landmarks L, compute the left eye centre $(X_\mathrm{L}, Y_\mathrm{L})$, the right eye centre $(X_\mathrm{R}, Y_\mathrm{R})$ and the inter-eye distance D_IED as in the algorithm in Section 7.7.2 with yaw angle $\alpha = 0$.[38]
2. Compute the eyes' midpoint $(X_\mathrm{C}, Y_\mathrm{C})$ as the mean (centre of gravity) of the left and right eye centres, i.e. as $(X_\mathrm{C}, Y_\mathrm{C}) = \big((X_\mathrm{L}+X_\mathrm{R})/2, (Y_\mathrm{L}+Y_\mathrm{R})/2\big)$.
3. Compute the distance T between the eye's midpoint and the chin as $T = \|(X_\mathrm{C}, Y_\mathrm{C}) - L_{16}\|_2$.
4. Compute and output the following values:
    a. The measure $q_\mathrm{l} = X_\mathrm{R}/D_\mathrm{IED}$ for the leftward crop.
    b. The measure $q_\mathrm{r} = A - X_\mathrm{L}/D_\mathrm{IED}$ for the rightward crop, where A is the width of the image.
    c. The measure $q_\mathrm{a} = Y_\mathrm{C}/T$ for the margin above the face.
    d. The measure $q_\mathrm{b} = (B - Y_\mathrm{C})/T$ for the margin below the face.

7.9.3 Evaluation

No evaluation of the algorithm for the quality component Single Face Present was performed.

7.9.4 Final Algorithm Selection

For the quality component Head Size, the FDIS algorithms (see Section 7.9.2) were selected and implemented in OFIQ.

For the mapping of the native quality measures q_l, q_r, q_a, q_b for the leftward crop, rightward crop, margin above the face and margin below the face, respectively, to the corresponding quality component values Q_l, Q_r, Q_a, Q_b in the target range [0,100], the functions are used, where ROUND and SIGMOID are defined as described in Section 5.4:

$$Q_\mathrm{l} = \mathrm{ROUND}(100\ \mathrm{SIGMOID}\ (q_\mathrm{l}, 0.9, 0.1))$$

$$Q_\mathrm{r} = \mathrm{ROUND}(100\ \mathrm{SIGMOID}\ (q_\mathrm{r}, 0.9, 0.1))$$

$$Q_\mathrm{a} = \mathrm{ROUND}(100\ \mathrm{SIGMOID}\ (q_\mathrm{a}, 1.4, 0.1))$$

$$Q_\mathrm{b} = \mathrm{ROUND}(100\ \mathrm{SIGMOID}\ (q_\mathrm{b}, 1.8, 0.1))$$

[38] The planar IED is used, because the width of the measurement zones should decrease with the yaw angle.

7.10 Head Pose

7.10.1 Data Selection

Three test sets for Head Pose were compiled from publicly available databases.

The first test set, AFLW2000v2, uses the images from the AFLW2000 database [19], which contains 2,000 facial images with labels for all three head-angles: pitch, yaw and roll (calculation of angles from Euler matrices). 74 images with incorrect or ambiguous ground truth angles have been excluded:

- 9 images with (incorrect) ground truth angles greater than 180 degrees or smaller than -180 degrees
- 36 images with ground truth angles greater than 89 or smaller than -89; note that these images have also been excluded for the evaluations in [57] and [58].
- Further 24 images, for which the ground truth angles were found to be incorrect
- 5 images that depict two comparably large faces which can result in ambiguous results.

The second test set consist of 2,790 facial images from the Pointing'04 database [59] with the head pose angles pitch and yaw.

The third test set is composed of 14,316 face images from the CMU Multi-PIE [36] database with labels for the yaw angle.

Figure 111: Example images from the test sets AFLW2000, Pointing'04 and CMU Multi-PIE.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.10.2 Selection and Prototyping of Candidate Algorithms

Three candidate algorithms based on publicly available CNNs for head pose estimation were implemented and evaluated.

In contrast to the original implementations, the candidate algorithms are based on the SSD face detector (Section 7). Since the CNN models were trained on bounding boxes output by different face detectors, the boundaries of the bounding boxes output by the SSD were cropped and/or extended using parameters that had been optimised with respect to the resulting MAE on the AFLW2000v2 test set.

For all candidate algorithms, the computation of the pose angles from the CNN's output has been unified so that first a rotation matrix R is computed and then the Euler angles are computed from R using the following function based on the formulas used to compute the angles from those specified in [60]:

Function *matrix2angle* takes as input a rotation matrix R and performs the following steps:

a. If $R_{31} > 0.9975$, set
$\phi_{\text{pitch}} = -\frac{\pi}{2}$, $\phi_{\text{yaw}} = \text{atan2}(-R_{12}, -R_{13})$, $\phi_{\text{roll}} = 0$.
b. If $R_{31} < -0.9975$, set
$\phi_{\text{pitch}} = \frac{\pi}{2}$, $\phi_{\text{yaw}} = \text{atan2}(R_{12}, R_{13})$, $\phi_{\text{roll}} = 0$.
c. If $R_{31} > -0.9975$ and $R_{31} < 0.9975$, set
$\phi_{\text{pitch}} = -\arcsin(R_{31})$, $\phi_{\text{yaw}} = \text{atan2}\left(\frac{R_{32}}{\cos(\phi_{\text{pitch}})}, \frac{R_{33}}{\cos(\phi_{\text{pitch}})}\right)$, $\phi_{\text{roll}} = \text{atan2}\left(\frac{R_{21}}{\cos(\phi_{\text{pitch}})}, \frac{R_{11}}{\cos(\phi_{\text{pitch}})}\right)$.[39]

7.10.2.1 Dense Head Pose Estimation

The first candidate algorithm uses a model ("sparse model") from the repository Dense-Head-Pose-Estimation[40].

The resulting algorithm takes as input a face image I in RGB colour channel order with 8 bits per channel and the face bounding box (a, b, c, d) output by the SSD face detector, and performs the following steps:

1. Compute the height $l = d - b$ and the centre (x, y) of the face bounding box.
2. Set $b = \lfloor y - 0.59 \cdot l \rfloor$ and $d = \lfloor y + 0.55 \cdot l \rfloor$.
3. Extend the box to square size by setting $a = \lfloor x + 0.5 \cdot (d - b) \rfloor$ and $c = a + d - b$.
4. If the new face bounding box (a, b, c, d) is not within the image region $[0, w-1] \times [0, h-1]$, where w, h are the width and height of I, respectively, pad the image:
    a. If $a < 0$, pad the image on the left by $p_x = -a$ pixels, set $c = c + a$ and $a = 0$.
    b. If $b < 0$, pad the image on the top by $p_y = -b$ pixels, set $d = d + b$ and $b = 0$.
    c. If $c \geq w$, pad the image on the right by $c - w + 1$ pixels.
    d. If $d \geq h$, pad the image on the right by $d - h + 1$ pixels.
5. Crop I to the rectangle defined by the corners (a, b) and (c, d).
6. Scale I to size 120x120.
7. Perform a min-max normalisation of the values of I to the range [-1,1].
8. Reshape I to an input tensor $T^{[0]}$of dimensions (1,3,120,120).
9. Run a forward pass through the CNN using $T^{[0]}$ as input to obtain an output tensor $T^{[1]}$ of dimensions (3,4).
10. Delete the last column of $T^{[1]}$ to obtain the rotation matrix R.
11. Compute ϕ_{pitch}, ϕ_{yaw}, ϕ_{roll} from R using the function *matrix2angle*.
12. Output ϕ_{pitch}, ϕ_{yaw}, ϕ_{roll}.
13. For each angle α, compute the quality scalar as $q_i = \max(0, \lfloor 100 \cdot \cos \alpha^2 \rceil)$, where the rounding function $\lfloor x \rceil$ is defined as $\lfloor x + 0.5 \rfloor$ for positive x and as $\lceil x - 0.5 \rceil$ for negative x.
14. Output the quality scalars q_1, q_2, q_3.

[39] Note that the division by $\cos(\phi_{\text{pitch}})$ is necessary only to adjust the sign of R_{21} and R_{11}.
[40] https://github.com/1996scarlet/Dense-Head-Pose-Estimation

7.10.2.2 3DDFA-V2

The second candidate algorithm is based on a head pose estimation model from the repository 3DDFA_V2[41] converted to ONNX.

The resulting algorithm takes as input the image I in BGR colour channel order with 8 bits per channel and the face bounding box (a, b, c, d) output by the SSD face detector, and performs the following steps:

1. Compute the height $l = d - b$ and the centre (x, y) of the face bounding box.
2. Set $b = \lfloor y - 0.44 \cdot l \rfloor$ and $d = \lfloor y + 0.51 \cdot l \rfloor$.
3. Extend the box to square size by setting $a = \lfloor x + 0.5 \cdot (d - b) \rfloor$ and $c = a + d - b$.
4. If the new face bounding box (a, b, c, d) is not within the image region, pad the image as in the algorithm of Dense Head Pose Estimation (Section 7.10.2.1).
5. Crop I to the bounding box.
6. Scale I to size 120x120.
7. Normalise I with mean $\mu = 127.5$ and standard deviation $\sigma = 128$ to obtain array A by setting
$$A_{i,j,k} = (I_{i,j,k} - \mu)/\sigma$$
for all i, j, k.
8. Reshape A to an input tensor T of dimensions (1,3,120,120).
9. Run a forward pass through the CNN using T as input to obtain an output tensor T' of dimensions (62,1)
10. Set $R_1 = \left(T_1^{[1]}, T_2^{[1]}, T_3^{[1]}\right)$ and $R_2 = \left(T_5^{[1]}, T_6^{[1]}, T_7^{[1]}\right)$
11. De-normalise R_1 by mean μ = (3.4926363e-04, 2.5279013e-07, -6.8751979e-07) and standard deviation σ = (1.7632153e-04, 6.7379435e-05, 4.4708489e-04), i.e. set
$$R_{1,i} = \sigma_i \cdot R_{1,i} + \mu_i$$
12. De-normalise R_2 by mean μ' = (-6.2955132e-07, 5.7572004e-04, -5.0853912e-05) and standard deviation σ' = (1.2313770e-04, 4.4930217e-05, 7.9236706e-05), i.e. set
$$R_{2,i} = \sigma'_i \cdot R_{2,i} + \mu'_i$$
13. Normalise R_1 and R_2 to unit length.
14. Set R_3 to the cross product of R_1 and R_2, i.e. $R_3 = R_1 \times R_2$.
15. Compose a matrix R from the column vectors R_1, R_2, R_3.
16. Compute $\phi_{\text{pitch}}, \phi_{\text{yaw}}, \phi_{\text{roll}}$ from R using the function *matrix2angle*.
17. Output $\phi_{\text{pitch}}, \phi_{\text{yaw}}, \phi_{\text{roll}}$.

7.10.2.3 SynergyNet

The third candidate algorithm uses a head pose estimation model from the repository SynergyNet[42], converted to ONNX.

Table 16: Means absolute errors on AFLW2000v2 of the two models published in the SynergyNet repository.

SynergyNet Model	MAE Pitch	MAE Yaw	MYA Roll
best.pth.tar	3.93	3.38	2.57
best_pose.pth.tar	3.91	3.30	2.56

[41] https://github.com/cleardusk/3DDFA_V2
[42] https://github.com/choyingw/SynergyNet

The resulting algorithm based on the SynergyNet CNN model takes as input an image I in BGR colour format and the coordinates (a, b, c, d) of the largest face bounding box, and performs the following steps.

1. Compute the height $l = d - b$ and the centre (x, y) of the face bounding box.
2. Set $b = \lfloor y - 0.46 \cdot l \rfloor$ and $d = \lfloor y + 0.64 \cdot l \rfloor$.
3. Extend the box to square size by setting $a = \lfloor x + 0.5 \cdot (d - b) \rfloor$ and $c = a + d - b$.
4. If the new face bounding box (a, b, c, d) is not within the image region, pad the image as in the algorithm of Dense Head Pose Estimation (Section 7.10.2.1).
5. Crop I to the rectangle defined by the corners (a, b) and (c, d).
6. Resize the image I to size 120×120 using OpenCV with bilinear interpolation.
7. Normalise I using mean $\mu = 127.5$ and standard deviation $\sigma = 128$ to obtain array A, i.e. set
$$A_{i,j,k} = (I_{i,j,k} - \mu)/\sigma$$
for all i, j, k.
8. Encode A as input tensor T of dimensions (1, 3, 120, 120) for the neural network model.
9. Run a forward pass through the CNN model using T as input to obtain an output tensor T' of dimension (1, 62).
10. Set $V = (T'_1, T'_2, T'_3, T'_5, T'_6, T'_7)$.[43]
1. De-normalise V to obtain W using mean
$\mu = (0.000316018, -0.000001804, -0.000001798, 0.000002551, 0.000434692, -0.000033830)$
and standard deviation
$\sigma = (0.000185973, 0.000303770, 0.000322442, 0.000253192, 0.000187914, 0.000202994)$
by setting
$$W_i = V_i \cdot \sigma_i + \mu_i \text{ for all } i.$$
11. Set $R_1 = (W_1, W_2, W_3)$ and $R_2 = (W_4, W_5, W_6)$.
12. Normalise R_1 and R_2 to unit length.
13. Set $R_3 = R_1 \times R_2$, where $\times$ denotes the cross product.
14. Compose a matrix R from the row vectors R_1, R_2 and R_3.
15. Compute $\phi_{\text{pitch}}, \phi_{\text{yaw}}, \phi_{\text{roll}}$ from R using the function *matrix2angle*.
16. Output $\phi_{\text{pitch}}, \phi_{\text{yaw}}, \phi_{\text{roll}}$.

7.10.2.4 Landmark-based Algorithms for Roll Angle

For computation of the roll angle, two further algorithms based on landmarks were implemented:

1. The first algorithm takes as input the affine transformation matrix M output by the alignment algorithm in Section 4.4.1 and computes the roll angle as $\theta = \text{atan2}(M_{12}, M_{11})$.
2. The second algorithm computes the eyes centres (X_L, Y_L) and (X_R, Y_R) as in the algorithm for the inter-eye distance, and computes the roll angle as $\theta = \text{atan2}(Y_L - Y_R, X_L - X_R)$.

For both algorithms the landmarks are computed using ADNet.

[43] The 4th element and the last 55 elements are not used for pose estimation

7.10.3 Evaluation

The model sizes and computational performances (using ONNXRuntime for 3DDFA-V2 and SynergyNet, and Tensorflow Lite for Dense Head Pose Estimation) are shown in the following table.

Table 17: Computational resources required by the implemented algorithms

Algorithm	Model Size	Computation time per image
3DDFA-V2	12.4 MB	1.5 ms
Dense Head Pose Estimation	12.5 MB	1.5 ms
SynergyNet	8.8 MB	2.5 ms

7.10.3.1 Evaluation with ground truth data

All CNNs were evaluated on the test sets w.r.t the Mean Absolute Error (MAE) of the native outputs (pose angles in degree), prior to mapping them to [0,100]. For the AFLW2000v2 and Pointing'04 test sets, yaw angles close to 90° (i.e. full profile faces) can result in an inaccurate result of the MAE metric due to ambiguities induced by the gimbal lock, as explained in [61]; therefore, we also evaluate the Geodetic Error (GE) according to [61], which measures the distance on the unit sphere (in degrees) between the rotation matrices from ground truth and prediction. For the CMU Multi-PIE test set, the gimbal lock is not an issue as the pitch and roll angles are not evaluated.

The results are summarised in Table 18:

Table 18: Evaluation results for the tested pose estimation algorithms on different test sets.

Algorithm	AFLW2000v2				CMU Multi-PIE	Pointing04		
	MAE pitch	MAE yaw	MAE roll	GE	MAE yaw	MAE pitch	MAE yaw	GE
Dense-Head-Pose-Estimation	5.3	3.1	3.2	6.0	3.3	14.0	12.4	26.5
3DDFA-V2	4.8	3.0	3.1	5.6	3.3	13.9	12.5	27.1
SynergyNet	4.2	3.0	2.6	6.0	4.2	14.4	14.1	28.3

ECDF curves for pitch and yaw angles on the AFLW2000v2 test set shown in Figure 112: For pitch, SynergyNet outperforms 3DDFA-V2 and Dense Head Pose Estimation, while for yaw, it is slightly worse.

As visible in Figure 113, for the roll angle on AFLW2000v2, SynergyNet performs best, and the two landmark-based algorithms have much higher error rates than all three CNN-based algorithms.

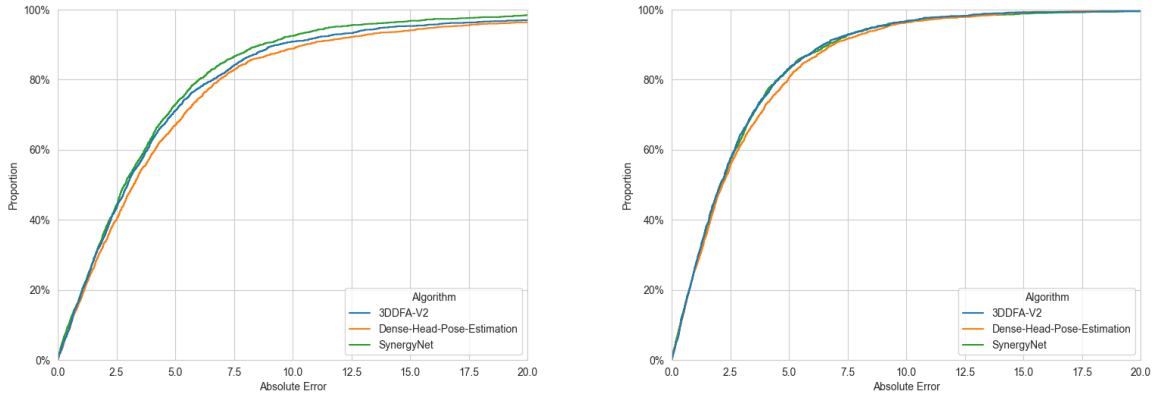

Figure 112: ECDF curves of the algorithms for the pitch (left) and yaw (right) angles of the quality component Head Pose on the AFLW2000v2 test set.

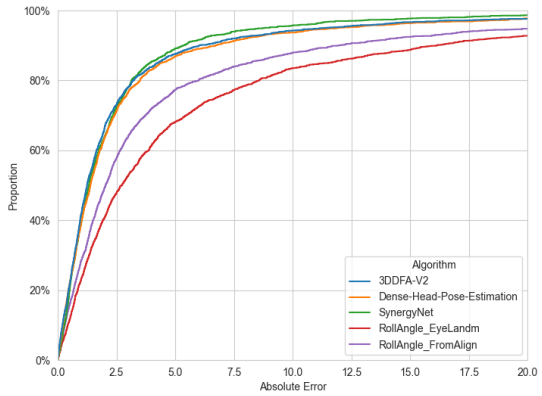

Figure 113: ECDF curves of the algorithms for the roll angles of the Head Pose quality component on the AFLW2000v2 test set.

On the Pointing'04 test set, SynergyNet performs also worse than 3DDFA-V2 and Dense Head Pose Estimation, in particular for yaw angles.

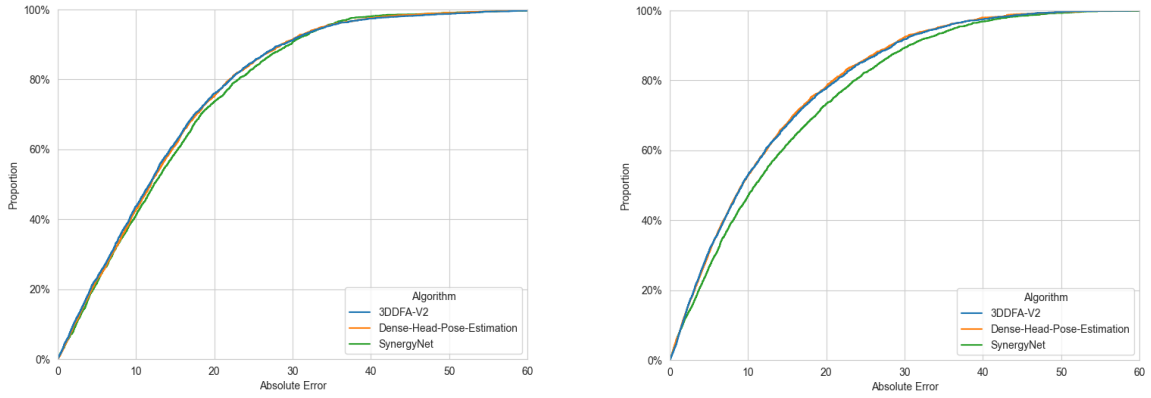

Figure 114: ECDF curves of the algorithms for the pitch (left) and yaw (right) angles of the quality component Head Pose on the Pointing'04 test set.

On the CMU Multi-PIE test set, where only labels for the yaw angle are available, 3DDFA-V2 and Dense Head Pose Estimation perform much better than Synergy as shown in Figure 115.

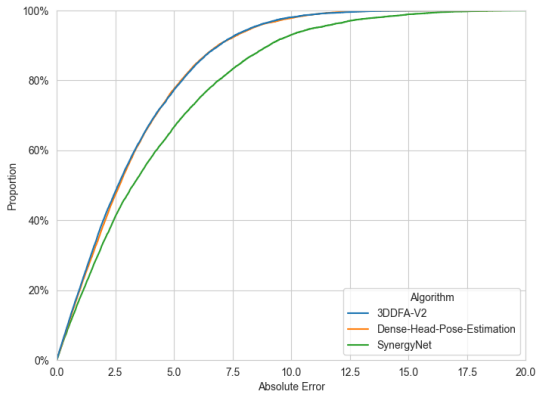

Figure 115: ECDF curves angles of the algorithms for the yaw angles of the quality component Head Pose on the CMU Multi PIE test set.

7.10.3.2 Evaluation by EDC curves

For the computation of the ECD curves, the quality components values (in the range [0,100]) were used, which are computed from the angles $(\phi_{\text{pitch}}, \phi_{\text{yaw}}, \phi_{\text{roll}}) = (\alpha_1, \alpha_2, \alpha_3)$ as $Q_i = \lfloor 100 \cdot \max(0, \cos(\alpha_i))^2 \rceil$, where the rounding function $\lfloor x \rceil$ is defined as $\lfloor x + 0.5 \rfloor$ for positive x and as $\lceil x - 0.5 \rceil$ for negative x.

The EDC curves for the head pose angles pitch, yaw and roll are shown in Figure 116, Figure 117, Figure 118, Figure 120, Figure 119.

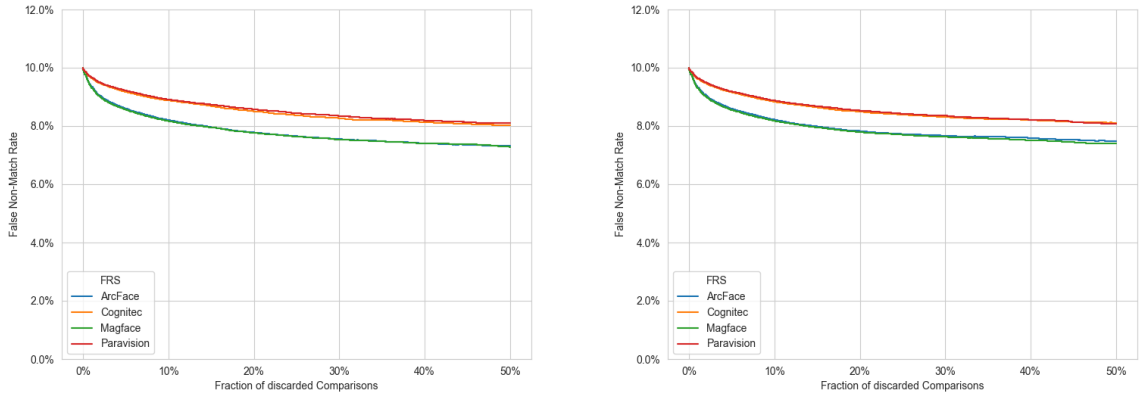

Figure 116: EDC curves for the pitch angle of the quality component Head Pose using the algorithms based on 3DDFAV2 (left) and Dense Head Pose Estimation (right) on the VGGFace2 test set.

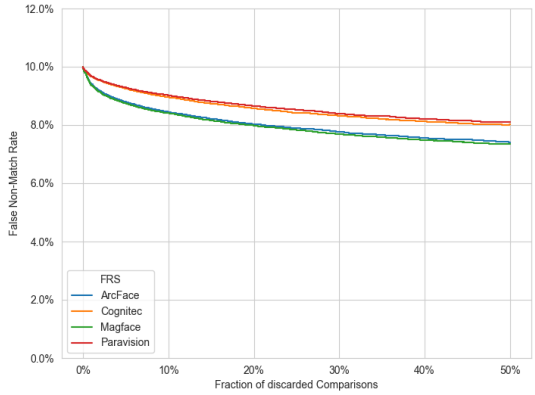

Figure 117: EDC curves for the pitch angle of the quality component Head Pose using the algorithm based on SynergyNet on the VGGFace2 test set.

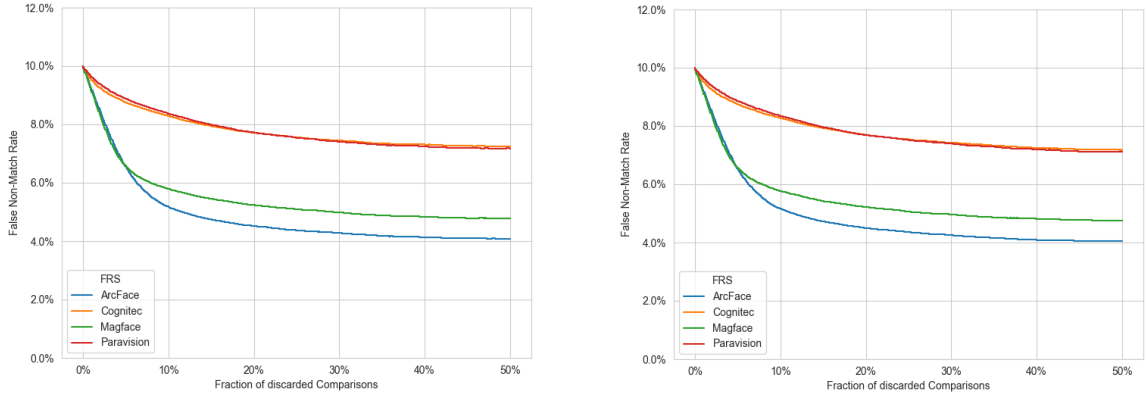

Figure 118: EDC curves for the yaw angle of the quality component Head Pose using the algorithm based on 3DDFAV2 (left) and Dense Head Pose Estimation (right) on the VGGFace2 test set.

For all angles (pitch, yaw and roll), 3DDFAV2 gives the largest decline of FNMR, very closely followed by Dense Head Pose Estimation, while, for SynergyNet, the effect on the FNMR is slightly lower.

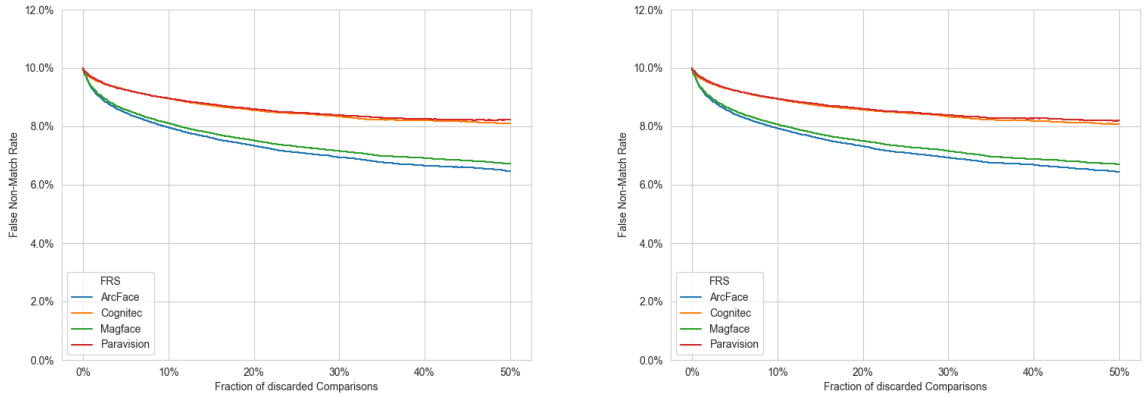

Figure 119: EDC curves for the roll angle of the quality component head pose using the algorithms based on 3DDFAV2 (left) and Dense Head Pose Estimation (right).

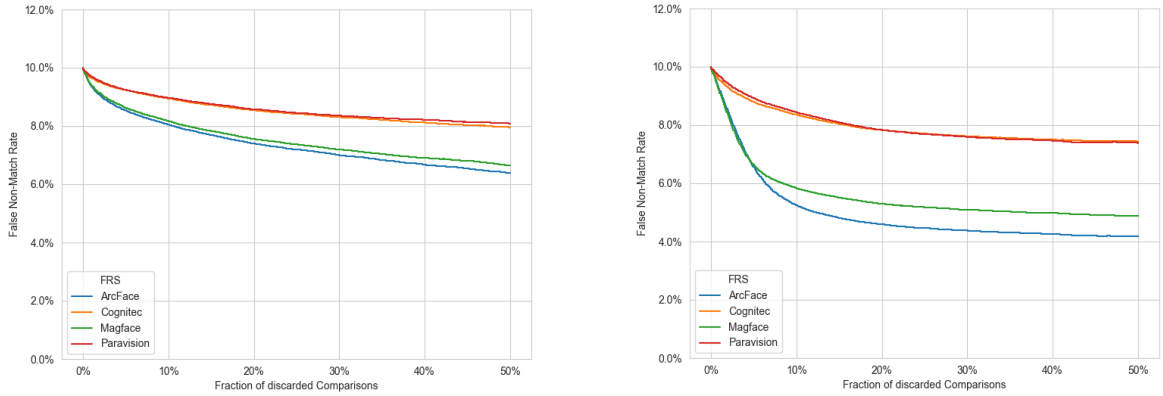

Figure 120: EDC curves for the roll angle (left) and the yaw angle (right) of the quality component Head Pose using the algorithm based on SynergyNet on the VGGFace2 test set.

7.10.3.3 Evaluation Results of NIST FATE

The algorithms based on SynergyNet and 3DDFA_V2 were implemented in the submissions secunet_003 and secunet_004, respectively, to the NIST FATE Quality SIDD track [16]. The algorithms were used for the SIDD quality components Yaw, Pitch and Roll.

It is important to note that, according to the FATE evaluation protocol, on the following plots *"The small numbers [...] represent the count of images for which the software did not detect a face".* In case of yaw and roll angles, for such images, the error was set to 45 degrees, in case of the pitch angle, it was set to 30 degrees. This protocol results in a significant increase of the reported mean absolute error (MAE) values by failures of the face detectors.

For the SIDD component Pose Yaw, three test sets are used. On average, the algorithm based on 3DDFA_V2 shows a higher accuracy than the algorithms based on SynergyNet and Dense Head Pose Estimation, which is in accordance with the results of the internal evaluation.

On test set 1, the algorithms based on 3DDFA_V2 and Dense Head Pose Estimation perform similarly and are on rank 9 and 10, respectively, whereas SynergyNet is on rank 12 and with a slightly higher MAE. However, face detection of the submitted algorithms failed for 734 (Dense Head Pose Estimation) and 786 (3DDFA_v2 and SynergyNet) out of the 6267 images from this test set, most of which having a yaw angle of 90 degrees, which results in an increase of the MAE by 5.2 and 5.6, respectively.

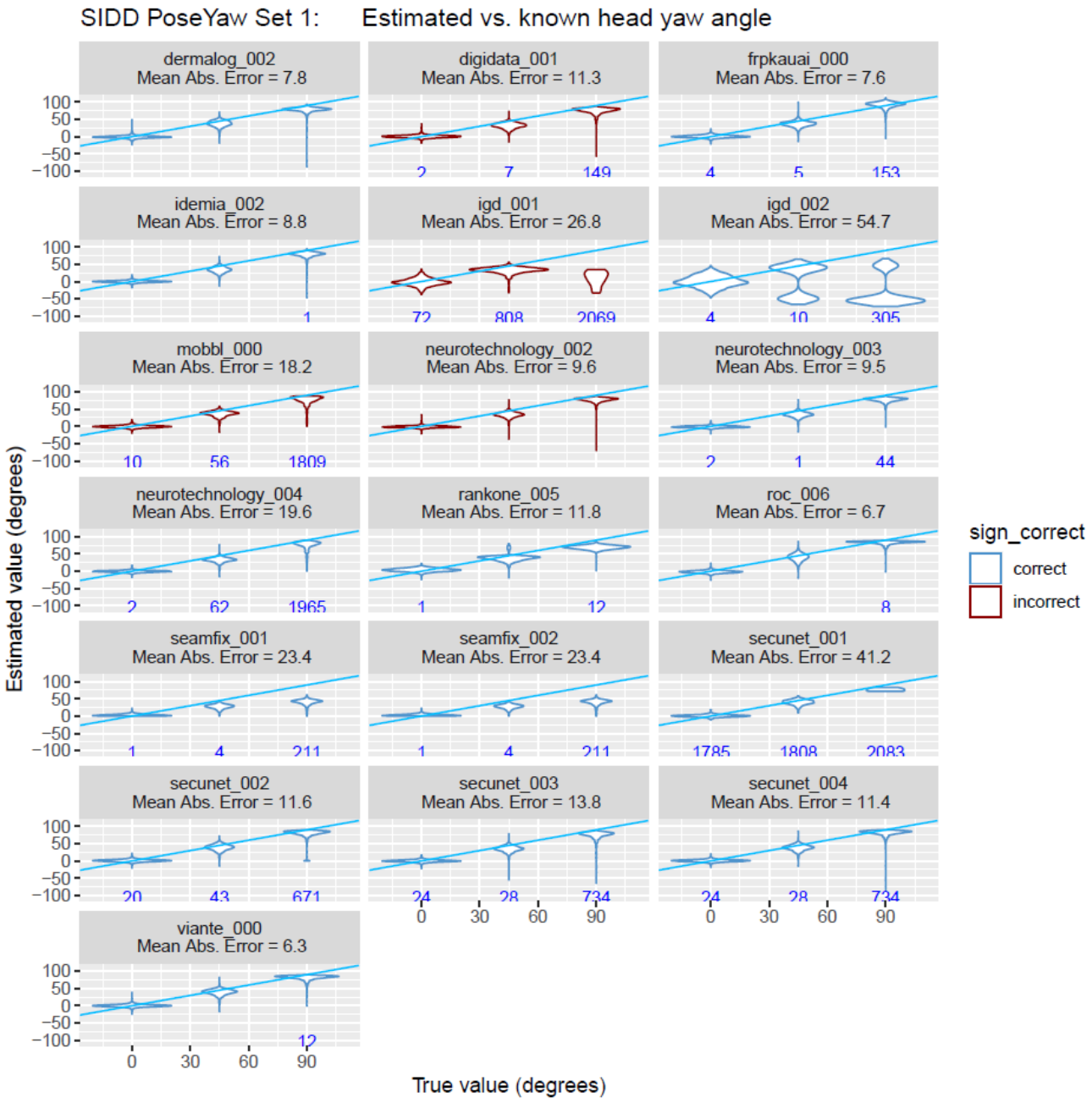

Figure 121: Outputs of the algorithms for the SIDD quality component Pose Yaw Angle vs. ground truth on test set 1 in the NIST FATE Quality SIDD report [16]. The algorithm based on SynergyNet is labelled as secunet_003, the algorithm based on 3DDFA_V2 is labelled as secunet_004 and the algorithm based on Dense Head Pose Estimation is labelled as secunet_002. The algorithms in secunet_001 and secunet_002 differ only by the minimum face width used in the face detection.

On test set 2, the algorithm based on 3DDFA_V2 is on rank 2, SynergyNet on rank 7 and Dense Head Pose Estimation on rank 13 out of 19. Face detection failed for 26 out of the 11.338 images for Dense Head Pose Estimation and for 18 images for SynergyNet and 3DDFA_V2, resulting in only a slight increase of the MAE by approximately 0.1.

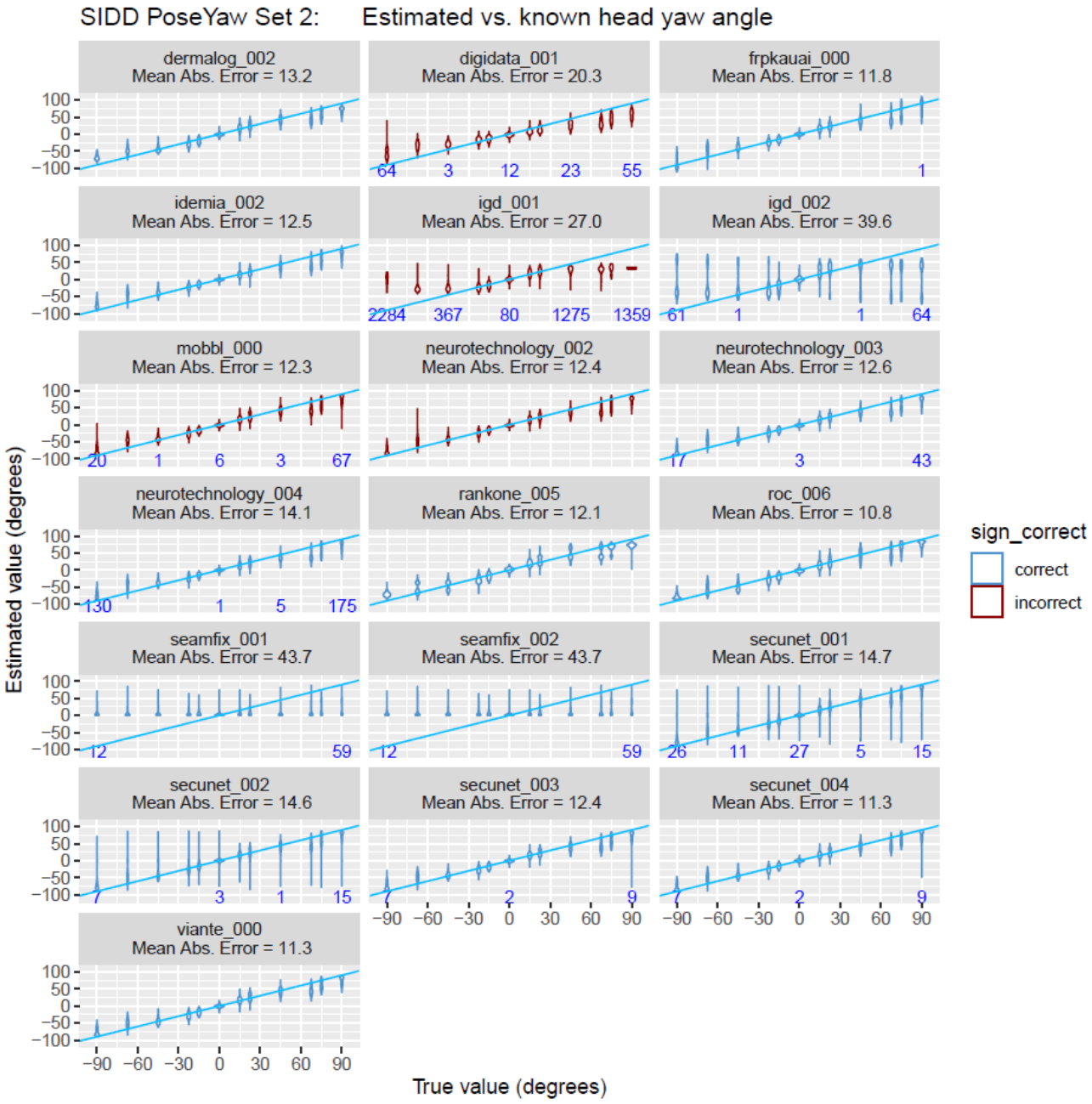

Figure 122: Outputs of the algorithms for the SIDD quality component Pose Yaw Angle vs. ground truth on test set 2 in the NIST FATE Quality SIDD report [16]. The algorithm based on SynergyNet is labelled as secunet_003, the algorithm based on 3DDFA_V2 is labelled as secunet_004 and the algorithm based on Dense Head Pose Estimation is labelled as secunet_002. The algorithms in secunet_001 and secunet_002 differ only by the minimum face width used in the face detection

On test set 3, all three algorithms are among the better ones but slightly less accurate than the leading competitors. SynergyNet is on rank 3, 3DDFA_V2 on rank 7 and Dense Head Pose Estimation on rank 8. On this test set, no failures of the face detection of the submitted algorithms occurred.

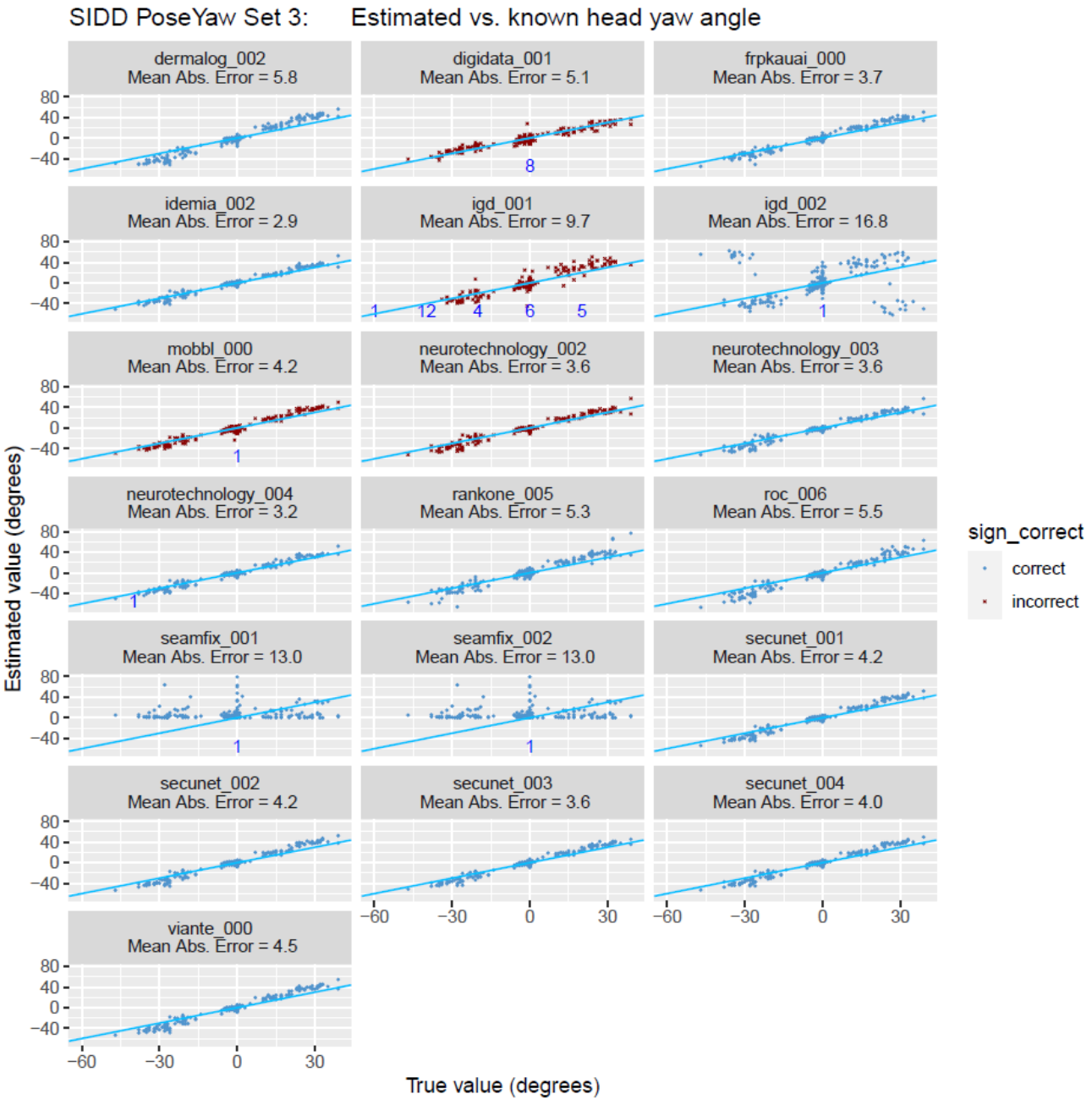

Figure 123: Outputs of the algorithms for the SIDD quality component Pose Yaw Angle vs. ground truth on test set 3 in the NIST FATE Quality SIDD report [16]. The algorithm based on SynergyNet is labelled as secunet_003, the algorithm based on 3DDFA_V2 is labelled as secunet_004 and the algorithm based on Dense Head Pose Estimation is labelled as secunet_002. The algorithms in secunet_001 and secunet_002 differ only by the minimum face width used in the face detection.

For the SIDD component Pose Pitch Angle, three test sets are used. On average, the algorithm based on SynergyNet shows a higher accuracy than the algorithms based on 3DDFA_V2 and Dense Head Pose Estimation, which, once more, is in accordance with the results of the internal evaluation.

On test set 1, however, 3DDFA_V2 and Dense Head Pose Estimation achieve the highest accuracy of all algorithms, i.e. rank 1, while SynergyNet is only in on rank 9 out of 19. On this test set, face detection failed for Dense Head Pose Estimation (secunet_002) for 132 out of the 6291 images, and for SynergyNet and 3DDFA_V2, for 262 images (mostly for images with ground truth pitch value of 30 degrees), resulting in an increase of the MAE by 0.6 and 1.3, respectively.

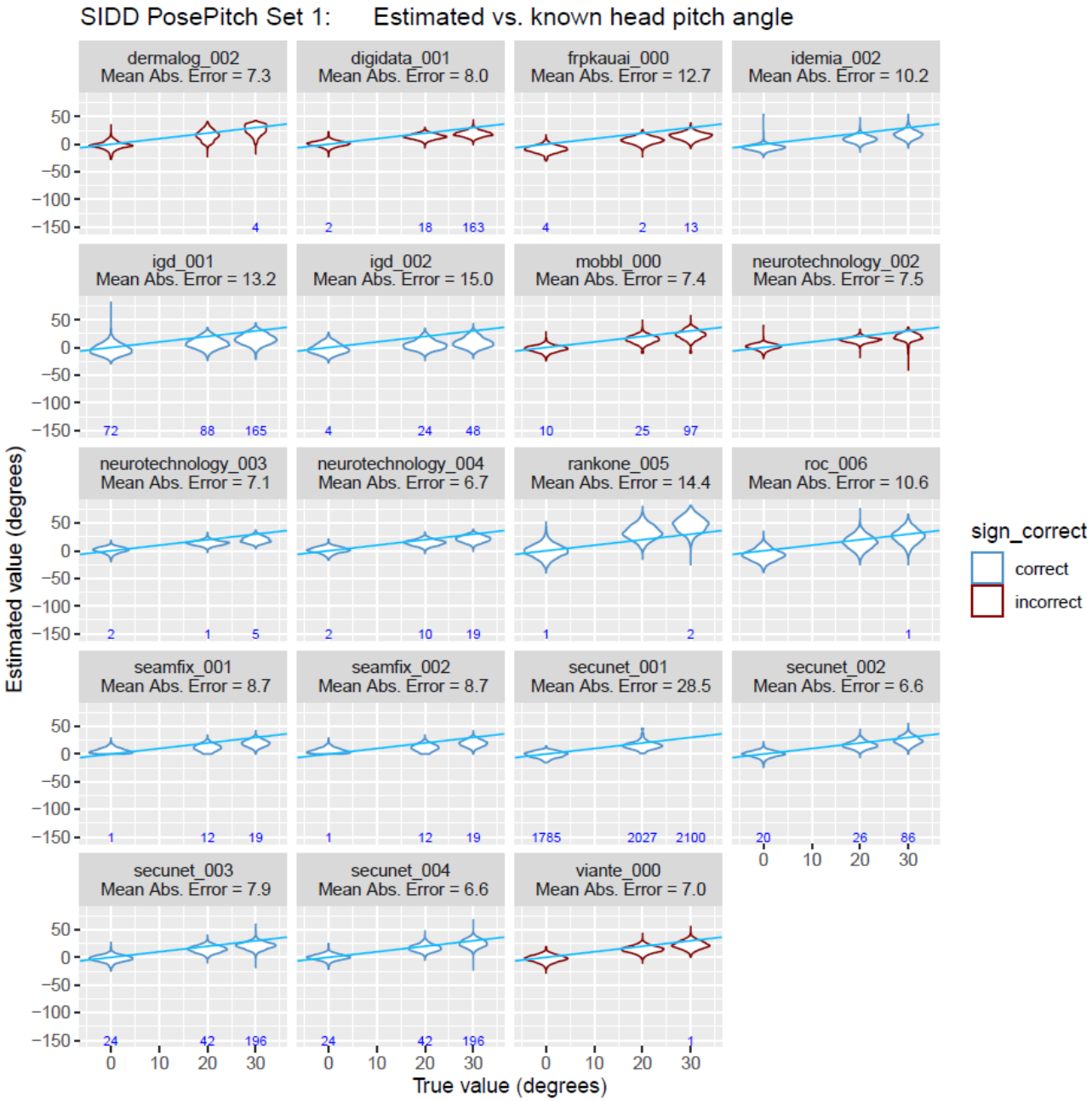

Figure 124: Outputs of the algorithms for the SIDD quality component Pose Pitch Angle vs. ground truth on test set 1 in FATE Face Image Quality Vector Assessment [16]. The algorithm based on SynergyNet is labelled as secunet_003, the algorithm based on 3DDFA_V2 is labelled as secunet_004 and the algorithm based on Dense Head Pose Estimation is labelled as secunet_002. The algorithms in secunet_001 and secunet_002 differ only by the minimum face width used in the face detection.

On test set 2, SynergyNet (secunet_003) is on rang 12, Dense Head Pose Estimation (secunet_002) on rank 15 and 3DDFA_V2 (secunet_004) on rank 18 out of 19. For all three algorithms, face detection failed for around 100 out of the 7145 images, resulting in an increase of the MAE by 0.4. Clearly, this does not explain the high MAE values. The violin plots show that, for some images, the estimation error is very large (over 60 degrees).

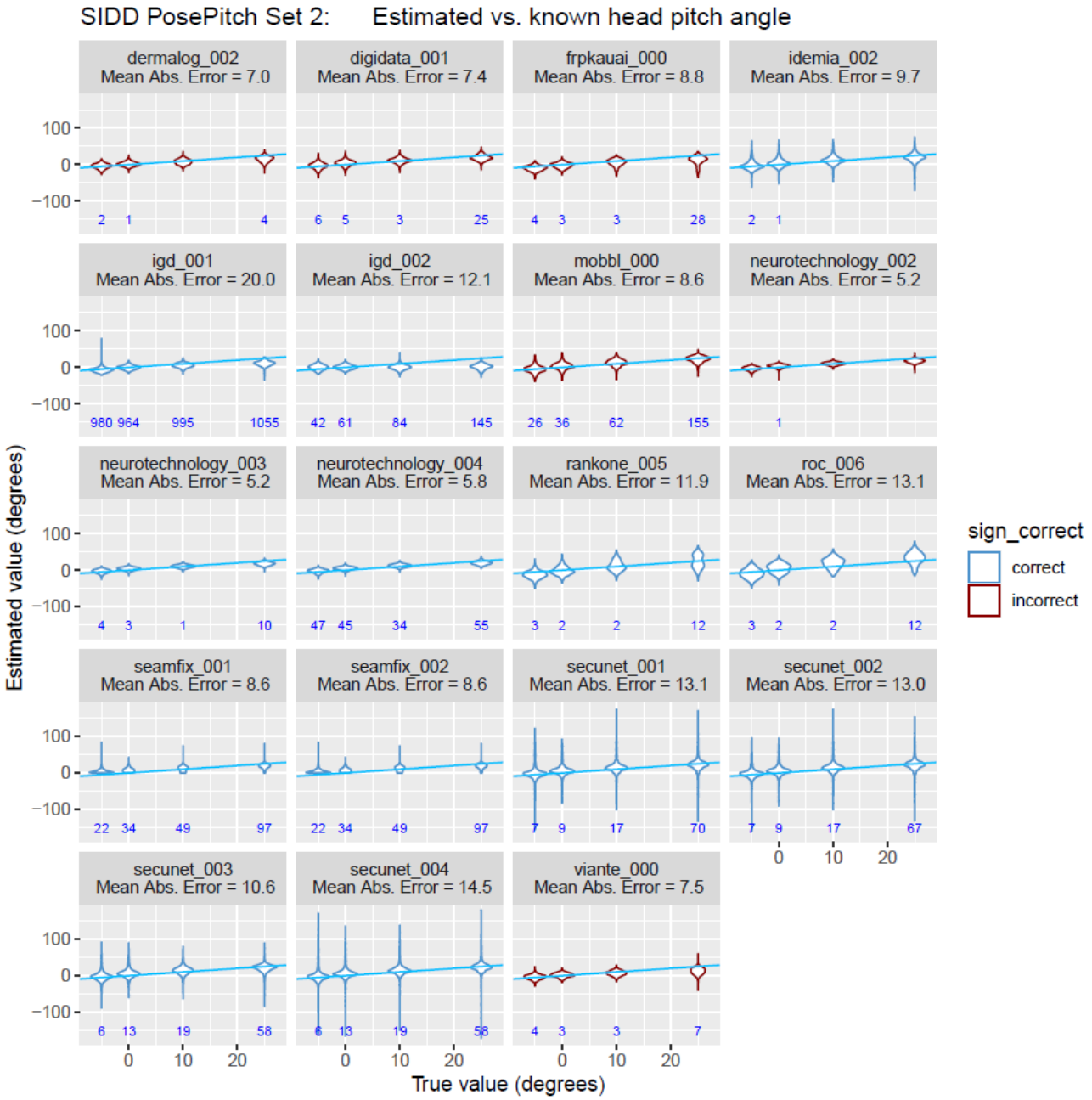

Figure 125: Outputs of the algorithms for the SIDD quality component Pose Pitch Angle vs. ground truth on test set 2 in the NIST FATE Quality SIDD report [16]. The algorithm based on SynergyNet is labelled as secunet_003, the algorithm based on 3DDFA_V2 is labelled as secunet_004 and the algorithm based on Dense Head Pose Estimation is labelled as secunet_002. The algorithms in secunet_001 and secunet_002 differ only by the minimum face width used in the face detection.

On test set 3, SynergyNet is on rank 2, Dense Head Pose Estimation on rank 5 and 3DDFA_V2 on rank 10 out of 19. However, the MAE values of the top 9 algorithms are very close, and even that of 3DDFA_V2 is not much higher. No failures of face detection occurred for the submitted algorithms.

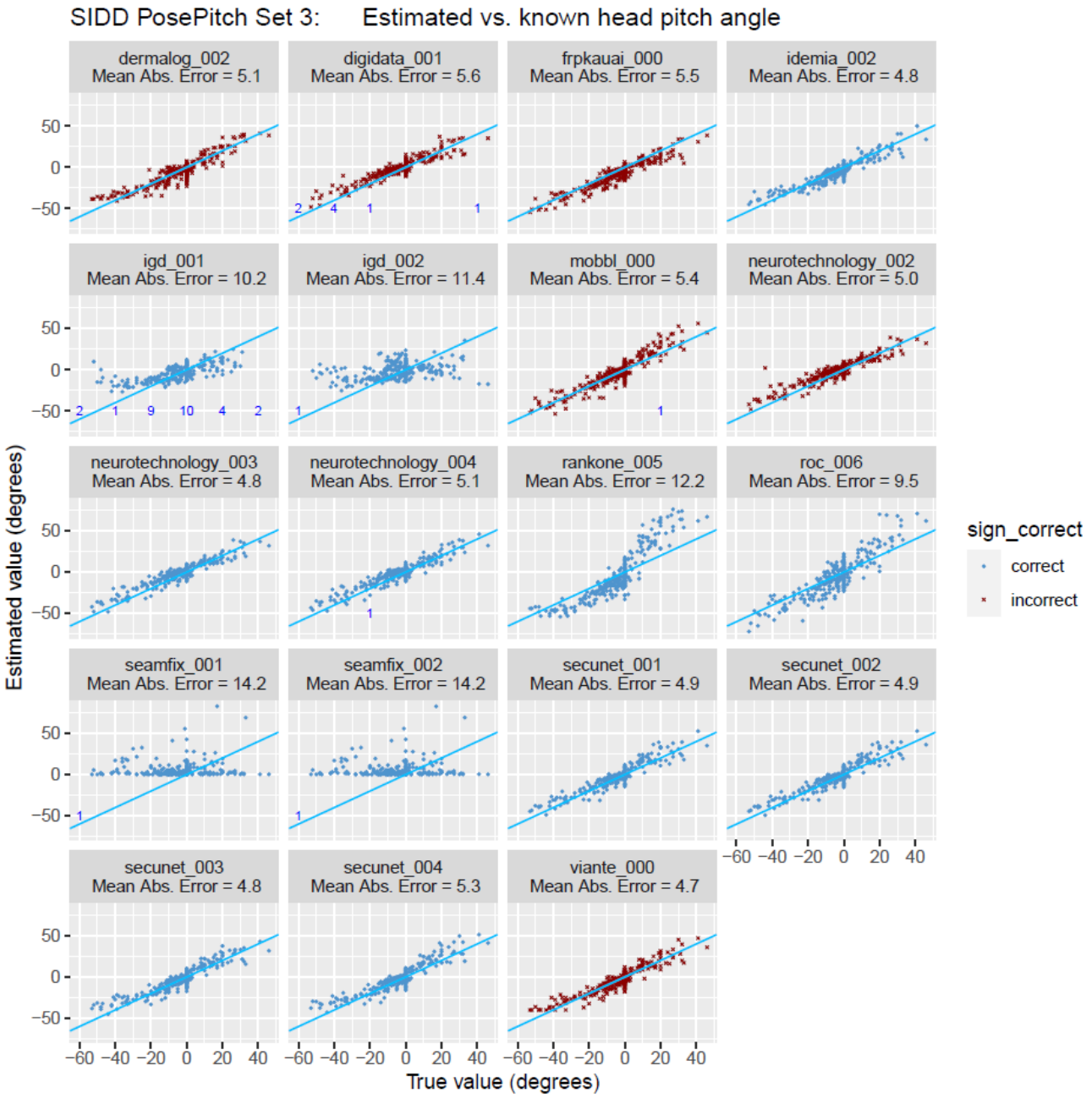

Figure 126: Outputs of the algorithms for the SIDD quality component Pose Pitch Angle vs. ground truth on test set 3 in the NIST FATE Quality SIDD report [16]. The algorithm based on SynergyNet is labelled as secunet_003, the algorithm based on 3DDFA_V2 is labelled as secunet_004 and the algorithm based on Dense Head Pose Estimation is labelled as secunet_002. The algorithms in secunet_001 and secunet_002 differ only by the minimum face width used in the face detection.

For the SIDD component Pose Roll Angle, only one test set is used. Here, the algorithms based on SynergyNet and 3DDFA_V2 are on ranks 3 and 5 out of 19, both having a very low MAE and only one failure to detect a face. Dense Head Pose Estimation is on rank 10 and failed to detect a face for 28 out of the 12.000 images which results in an increase of the MAE by 0.1.

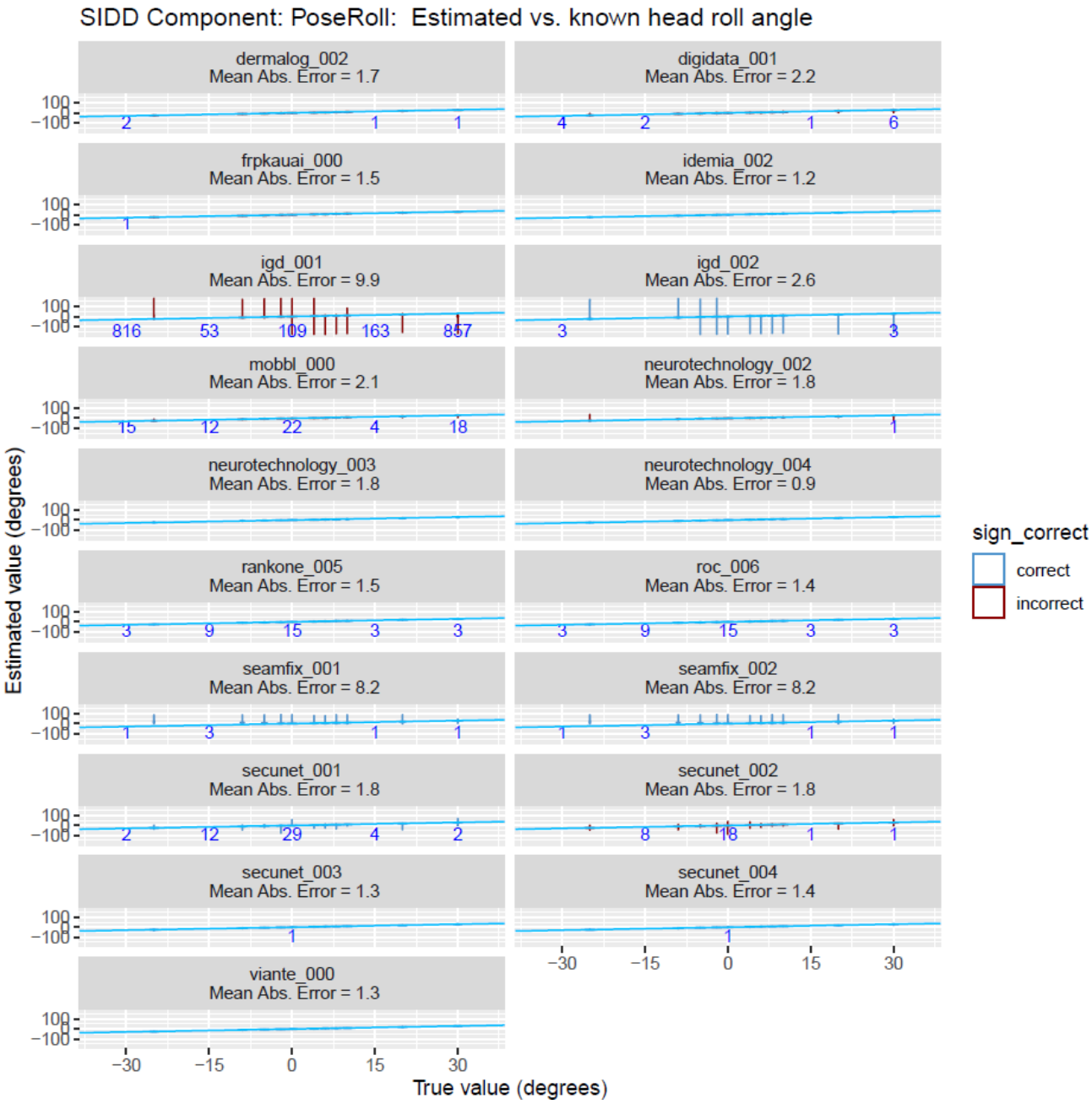

Figure 127: Outputs of the algorithms for the SIDD quality component Pose Roll Angle vs. ground truth in the NIST FATE Quality SIDD report [16]. The algorithm based on SynergyNet is labelled as secunet_003, the algorithm based on 3DDFA_V2 is labelled as secunet_004 and the algorithm based on Dense Head Pose Estimation is labelled as secunet_002. The algorithms in secunet_001 and secunet_002 differ only by the minimum face width used in the face detection and the sign of the output.

7.10.4 Final Algorithm Selection

For the quality component Head Pose, the algorithm 3DDFA-V2 specified in Section 7.10.2.2 was selected and implemented in OFIQ.

For the mapping of the native quality measures $(\phi_{\text{pitch}}, \phi_{\text{yaw}}, \phi_{\text{roll}}) = (\alpha_1, \alpha_2, \alpha_3)$ to the corresponding quality component values Q_1, Q_2, Q_3 in the target range [0,100] the function

$$Q_i = \text{ROUND}(100 \cdot \max(0, \cos(\alpha_i))^2)$$

is used, where ROUND is defined as described in Section 5.4.

7.11 Expression Neutrality

7.11.1 Data Selection

7.11.1.1 Training Set

A training set was compiled as follows:

- 920 images from CK+ [62], of which 593 have emotion label Neutral and the others have emotion labels Anger, Contempt, Disgust, Fear, Happiness, Sadness, or Surprise.
- 3,086 images from MultiPIE [36], of which 1,140 have emotion label Neutral and the others have emotion labels Disgust, Smile (Happiness), Surprise, or Squint.
- 1,441 images from the Chicago Face Database (CFD) [53], of which 831 have emotion label Neutral and the others have emotion labels Anger, Fear, Happy Open Mouth, or Happy Closed Mouth.
- 2,692 images from FEAFA [51], which contains images with labels (between 0 and 1) for 24 action units (AU) and action descriptors (AD) from the Facial Action Coding System (FACS) [63]. 796 images were chosen for which all labels are 0 (Neutral Expression) and the remaining 1,896 images were chosen with labels fulfilling one of the following conditions:
  - Unilaterally distorted mouth: Large value for Left Lip Corner Pull and small value for Right Lip Corner Pull or vice versa (both AU12).
  - Pressed lips: Large values for Upper Lip Suck and Lower Lip Suck (both AU28).
  - Pursed lips: Large values for Lip Pucker (AU18).
  - Inflated cheeks: Large values for Cheeks Puff (AD34)
- 967 images from FRGCv2 [34] (captured under constraint conditions) with neutral expression.
- 385 images from Flickr Faces HQ (FFHQ) [64] and CelebA-HQ [19], of which 186 have emotion label Neutral and the others have emotion labels Contempt or Sadness (labels taken from Kaggle [65]).

Example images from CK+, Multi-PIE, CFD and FEAFA are shown in Figure 128 and Figure 129.

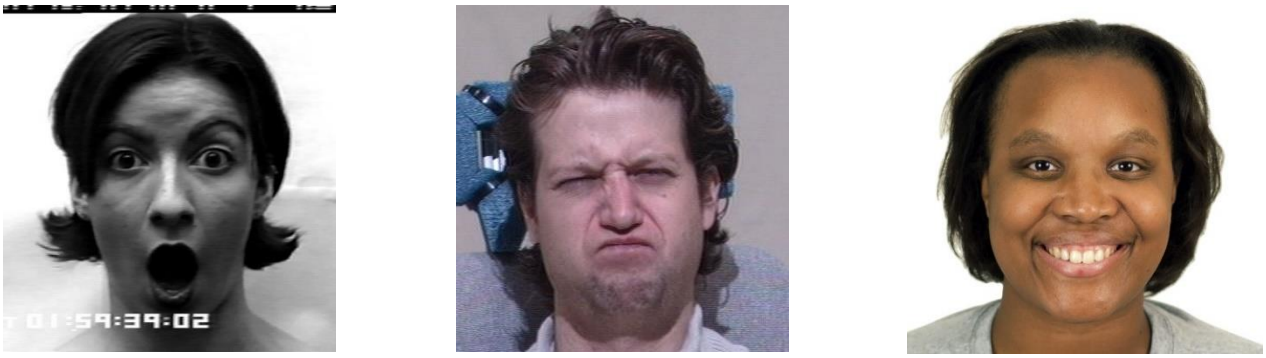

Figure 128: Example images with different emotion labels (surprise, disgust and happiness) from CK+ (left), CMU Multi-PIE (middle) and the Chicago Face database (right).

Figure 129: Examples of expressions from FEAFA: Unilaterally distorted mouth, pressd lips, prused lips and inflated cheeks.

The resulting distribution of source datasets and original labels in the training set is shown in the following table.

Table 19: Distribution of expressions and source data sets in the training set.

	CK+	Multi-PIE	CFD	FEAFA	FRGC	FFHQ & CelebA-HQ	Total
Anger	45	0	154	0	0	0	199
Contempt	18	0	0	0	0	168	186
Disgust	59	444	0	0	0	0	503
Fear	25	0	149	0	0	0	174
Happiness	69	942	307	0	0	0	1,318
Sadness	28	0	0	0	0	131	159
Surprise	83	406	0	0	0	0	489
Squint	0	154	0	0	0	0	154
AU+AD	0	0	0	1,896	0	0	1,896
Neutral	593	1,140	831	796	967	186	4,513
Total	920	3,086	1,441	2,692	967	385	9,591

All images in the training set were re-labelled with binary labels “Neutral” and “Non-Neutral” based on the original emotion labels

7.11.1.2 Test Sets

Two test sets were compiled for the evaluation.

The first test set, *MUG+FERETv2*,[44] contains 355 images from the MUG Facial Expression Database [66], of which 51 show a neutral expression, and 245 images with neutral expression from the FERET database [39]. Example images are shown in Figure 130.

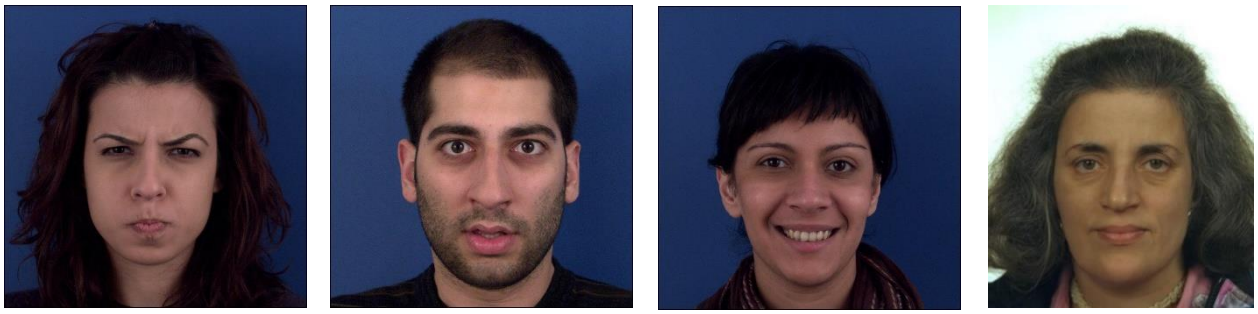

Figure 130: Example images of the test set MUG+FERETv2 from MUG (3 images on the left) and from FERET (right).

The second test set, *DISFA+*, was created using the following images:

- 45 images with neutral expressions and 364 images with non-neutral expressions from the DISFA dataset [67], which contains images with labels for Action Units (AU) and action descriptors (AD) from the Facial Action Coding System (FACS) [63]. The neutral images were chosen to have value 0 for all provided Action Units labels (AU1, AU2, AU4, AU5, AU6, AU9, AU12, AU15, AU17, AU20, AU25, AU26). The non-neutral images were chosen to have, for at least one AU label, a value greater or equal than 3.
- 71 images with neutral and 209 with non-neutral expression from the first 9 subjects of FEAFA. In order to ensure a clean separation between training and test set, no images of these subjects were used in the training set.
- 255 images with neutral and 32 images with non-neutral expression (with original label Anger) from the ARFace database [35].
- 70 images with neutral expression from the H-BRS multispectral SWIR/RGB Database [17].
- 124 images with neutral expression from the Sejong face database [10].

In summary, the DISFA+ contains 565 images with neutral and 605 images with non-neutral expression.

Figure 131: Example images of the test set DISFA+ taken from (left to right) ARFace, H-BRS, DISFA, Sejong Database and FEAFA. The second and fourth image show a neutral expression.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

[44] The „v2“ refers to a slight improvement of the test set in the course of evaluation.

7.11.2 Selection and Prototyping of Candidate Algorithms

Four candidate algorithms were implemented and evaluated. The first two candidate algorithms are based on the Distract your Attention Network (DAN) [68] for emotion estimation. The third candidate algorithm is based on DMUE [69] for emotion estimation. The fourth candidate algorithm is based on the expression neutrality approach proposed in [70].

7.11.2.1 Expression Estimation using DMUE

The algorithms based on DMUE [69] uses a CNN model trained on AffectNet with 7 classes published in the official DMUE repository[45],

The first algorithm takes the output of the CNN model for the emotion class Neutral as estimation for expression neutrality; since no additional classifier is used on top of the CNN, we henceforth refer to this algorithm as *DMUE with trivial classifier*. It takes as input an aligned image I output by the algorithm in Section 4.4.1 in RGB colour channel order with 8 bits per channel, and the face bounding box (a, b, c, d) of the largest face in the aligned image computed with the SSD face detector specified in Section 7. To initialise the algorithm, the CNN model file is loaded in PyTorch.

1. Crop I to the bounding box.
2. Resize I to 224x224 with bilinear interpolation.
3. Divide I by 255 and normalise the result with the values m= (0.485, 0.456, 0.406) and s = (0.229, 0.224, 0.225) for mean and standard deviation.
4. Reshape I to a 4-dimensional input tensor T of dimensions (1,3,224,224).
5. Run a forward pass through the model M with T as input to obtain an output tensor T' of dimension (1,7)
6. Cast the output tensor T' to a feature vector V.
7. Apply the Softmax function to V to obtain a feature vector F, i.e. $F = \text{softmax}(V)$.
8. If the trivial classifier is used, output F_i, where i is the index of the output node representing emotion class “neutral”.
9. Otherwise, run the Random Forest classifier model on the feature vector F to obtain an output x.
10. Output x.

The second algorithm uses an additional Random Forest model trained to classify the expression neutrality using the 7 outputs of the same DMUE CNN model. In the course of development, also SVM and AdaBoost classifiers were tested, but the Random Forest was found slightly superior.

The algorithm is identical to the previous one, except an additional final step, where the Random Forest classifier is used (*DMUE with Random Forest classifier*). It takes as input an aligned image I output by the algorithm in Section 4.4.1 in RGB colour channel order with 8 bits per channel, and the face bounding box (a, b, c, d) of the largest face in the aligned image computed with the SSD face detector specified in Section 7. To initialise the algorithm, the CNN model file is loaded in PyTorch and the corresponding Random Forest model file is loaded with OpenCV.

1. Crop I to the bounding box.
2. Resize I to 224x224 with bilinear interpolation.
3. Divide I by 255 and normalise the result with the values m= (0.485, 0.456, 0.406) and s = (0.229, 0.224, 0.225) for mean and standard deviation.

[45] https://github.com/JDAI-CV/FaceX-Zoo/tree/main/addition_module/DMUE

4. Reshape *I* to a 4-dimensional input tensor T of dimensions (1,3,224,224).
5. Run a forward pass through the model *M* with T as input to obtain an output tensor T' of dimension (1,7)
6. Cast the output tensor T' to a feature vector *V*.
7. Apply the Softmax function to *V* to obtain a feature vector *F*, i.e. $F = \text{softmax}(V)$.
8. Run the Random Forest classifier model on the feature vector *F* to obtain an output *x*.
9. Output *x*.

7.11.2.2 Expression Estimation using DAN and Random Forest Classifier

The algorithm based on DAN use a CNN model trained on AffectNet [71] with 7 classes published in the official DAN repository[46] and a Random Forest model trained to classify the expression neutrality using the 7 outputs of the CNN model. In the course of development, also SVM and AdaBoost classifiers were tested, but the Random Forest was found slightly superior.

The algorithm takes as input an aligned image *I* output by the algorithm in Section 4.4.1 in RGB colour channel order with 8 bits per channel, and the face bounding box (a, b, c, d) of the largest face in the aligned image computed with the SSD face detector specified in Section 7. To initialise the algorithm, the CNN model file is loaded in PyTorch and the corresponding Random Forest model file is loaded with OpenCV.

10. Adjust the bounding box by setting

    $a = a - (c - a)/8$,

    $c = c + (c - a)/8$, and

    $b = b - (d - b)/20$.
11. Crop *I* to the bounding box.
12. Resize *I* to 224x224 with bilinear interpolation.
13. Divide *I* by 255 and normalise the result with the following values m and s for mean and standard deviation:
    a. If the DAN AffectNet7 model is used, m= (0.485, 0.456, 0.406) and s = (0.229, 0.224, 0.225)
    b. If the DMUE model is used, m = (0.5, 0.5, 0.5) and s = (0.5, 0.5, 0.5).
14. Reshape *I* to a 4-dimensional input tensor T of dimensions (1,3,224,224).
15. Run a forward pass through the model object *M* with T as input to obtain an output tensor T'.
    a. If the DAN AffectNet7 model is used, T' has dimensions (1,7)
    b. If the DMUE model is used, T' has dimensions (1,8).
16. Cast the output tensor T' to a feature vector *V*.
17. Apply the Softmax function to *V* to obtain a feature vector *F*, i.e. $F = \text{softmax}(V)$.
18. Run the Random Forest classifier model on the feature vector *F* to obtain an output *x*.
19. Output *x*.

The amount of cropping of the face bounding box was chosen to resemble the cropping to the face region in the original implementation.

[46] https://github.com/yaoing/DAN

7.11.2.3 Algorithm HSEmotionWithTwoCNNs

This algorithm is based the expression neutrality classification method proposed in [70]. It uses two CNN models EfficientNet-b0 model enet_b0_8_best_vgaf and the EfficientNet-b2 model enet_b2_8 from the HSEmotion library[47] (henceforth called "HSE"), which is the official implementation of [72], for feature extraction, and the AdaBoost classifier "HSE-1-2" from GitHub repository Efficient-Expression-Neutrality-Estimation[48], which is the official implementation of [70]. The CNN models were modified to output the embeddings of the second last layers, which are used as features for the AdaBoost classifier, and, in order to prevent misuse of the implementation for emotion estimation (which may be restricted by legislation in certain regions), the weights of the final layer were all set to zero (the output of this layer is not used anyway). For performance reasons, the CNN models were converted to the ONNX format and executed with ONNXRuntime.

The pre-processing of the input images and the concatenation of the embeddings of the CNNs was aligned (by personal communication) with that used by the authors of [70]. The resulting algorithm, HSEmotionWithTwoCNNs, takes as input an aligned input image I in BGR colour channel order output by the algorithm in Section 4.4.1 and performs the following steps:

1. Convert I to RGB colour channel order.
2. Crop I by 144 pixels from the left and right, respectively, by 148 from the top and by 128 from the bottom.
3. Resize I to dimension (224,224) using OpenCV with bilinear interpolation.
4. Normalise I with mean $\mu = (123.7, 116.3, 103.5)$ and standard deviation $\sigma = (58.4, 57.1, 57.4)$ to obtain array A by setting

   $$A_{i,j,k} = (I_{i,j,k} - \mu_k)/\sigma_k$$

   for all i, j, k.
5. Reshape A to a 4-dimensional input tensor T_1 of dimensions (1,3,224,224).
6. Feed the tensor T_1 to the modified CNN model HSE enet_b0_8_best_vgaf to obtain an output tensor T'_1 of dimensions (1,1280).
7. Cast the output tensor T'_1 to a feature vector F_1 of length 1280.
8. Resize I to dimension (260,260) using bilinear interpolation to obtain I_2.
9. Reshape I_2 to a 4-dimensional input tensor T_2 of dimensions (1,3,260,260).
10. Feed the tensor T_2 to the modified CNN model HSE enet_b2_8 to obtain an output tensor T'_2 of dimensions (1,1408).
11. Cast the output tensor T'_2 to a feature vector F_2 of length 1408.
12. Concatenate F_1 and F_2 to obtain a final feature vector F of length 2.688.
13. Run the AdaBoost classifier model HSE-1-2 on the feature vector F to obtain the output x.
14. Output x.

7.11.3 Evaluation

The Detection Error Trade-Off (DET) curves of all implemented algorithms are plotted in Figure 132 for the test sets DISFA+ and MUG+FERETv2, respectively. The SSD face detector and ADNet landmark estimation

[47] https://github.com/HSE-asavchenko/face-emotion-recognition
[48] https://github.com/dasec/Efficient-Expression-Neutrality-Estimation

were used for face detection and alignment. On DISFA+, the algorithm based on two HSEmotion models and an AdaBoost shows by far the lowest error rates. On MUG+FERETv2, however, the algorithm based on DMUE and the trivial classifier (see Section 7.11.2.1) also shows competitive performance but is outperformed by the HSEmotionWithTwoCNNs algorithm for lower false positive rates and for low false negative rates. A potential explanation is that, since the Adaboost classifier has been trained on internal embeddings of the HSE model, it is better in detecting non-neutral expressions not associated with emotions, but the trivial classifier, which uses the emotion classification of the DMUE model, is better in detecting emotion-based expressions.

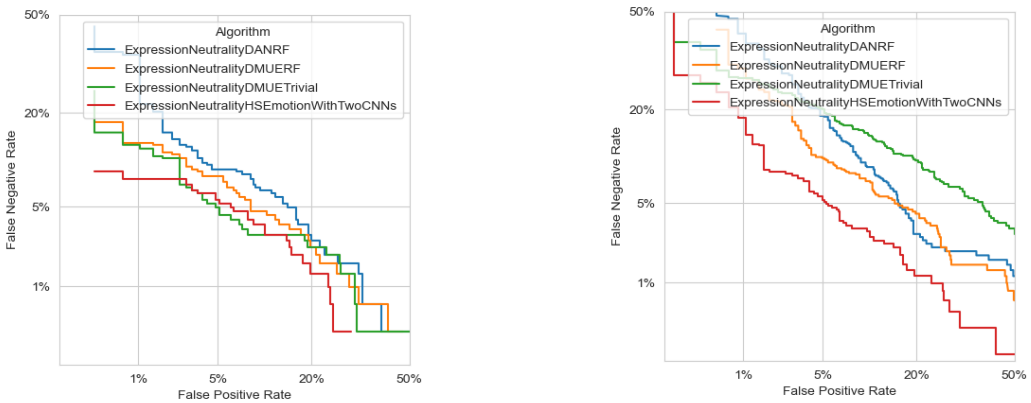

Figure 132: DET curves for the algorithms for quality component Expression Neutrality on the test sets MUG+FERETv2 (left) and DISFA+ (right)

The corresponding EDC curves are shown in Figure 134 and Figure 133. Among the tested algorithms, the one based on DMUE and the trivial classifier shows the best reduction in terms of FNMR for discarding non-neutral face images. However, achieved reductions appear insignificant, in particular for the used open-source face recognition system. A possible explanation for this is that low scores are frequently assigned to images with smiling faces, while these images do not represent a challenge for face recognition.

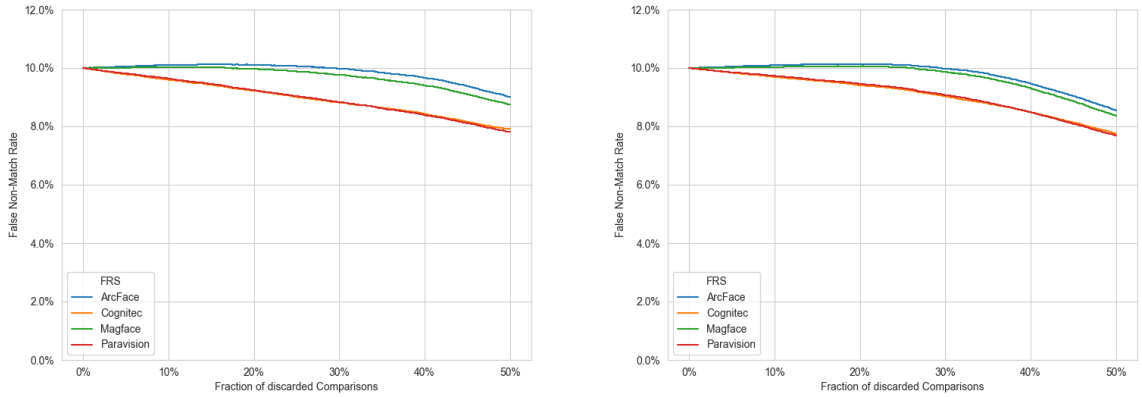

Figure 134: EDC curves for quality component Expression Neutrality with the algorithms DANRF and DMUERF on the VGGFace2 test set.

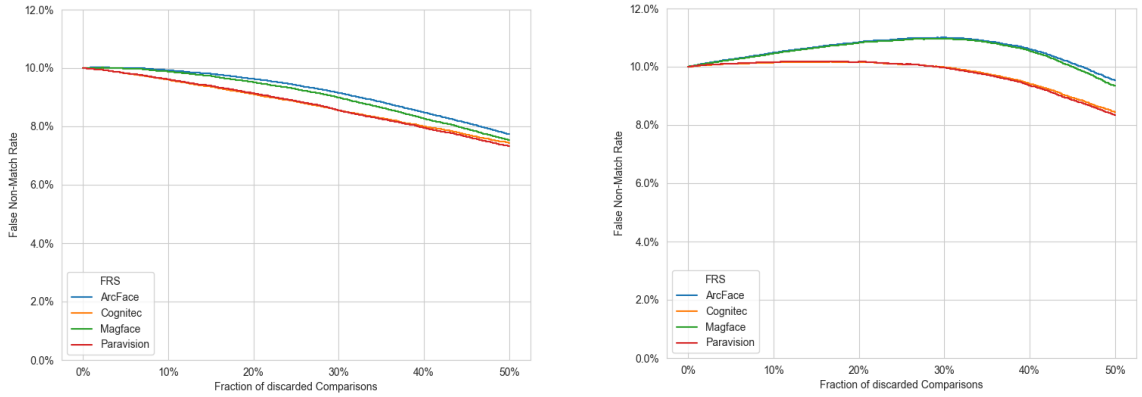

Figure 133: EDC curves for quality component Expression Neutrality with the algorithms DMUETrivial and HSEmotionWithTwoCNNs on the VGGFace2 test set.

7.11.4 Final Algorithm Selection

For the quality component Expression Neutrality, the algorithm HSEmotionWithTwoCNNs specified in Section 7.11.2.3 was selected and implemented in OFIQ. In the used ONNX models, the weights of the final layer (which are not needed by the algorithm) were all set to zero to prevent misuse of the implementation for emotion estimation and, hence, to avoid issues with regulations restricting such uses.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the function

$$Q = \mathrm{ROUND}(100 \cdot \mathrm{SIGMOID}(x, -5000, 5000))$$

is used, where ROUND and SIGMOID are defined as described in Section 5.4.

7.12 No Head Covering

7.12.1 Data Selection

Two test sets with facial images containing miscellaneous head coverings were compiled from publicly available datasets.

- CAS-PEAL-R1 database: 1,308 facial images with different head coverings were taken from the subfolder FRONTAL/Accessory containing in the filenames the keys “A4”, “A5” or “A6”. 1,039 facial images without head coverings were taken from the folder FRONTAL/Normal.
- CelebAMask-HQ database [24]: First, based on the labels provided with the database, 1068 facial images of subjects wearing head coverings (images for which the segmentation mask’s attribute "_hat" is available) and 1073 images with no head coverings were selected. Later, 50 of the selected with label “head covering” were excluded, because they either don’t show any head covering at all or a very unusual, partial head covering (e.g. diadem, ribbon, hood of chain mail). The sanitised test set is coined CelebAMaskHQ-V2.

Figure 135: Examples of face images with no head coverings (left) and with head coverings (right) from the test set CAS PEAL-R1.

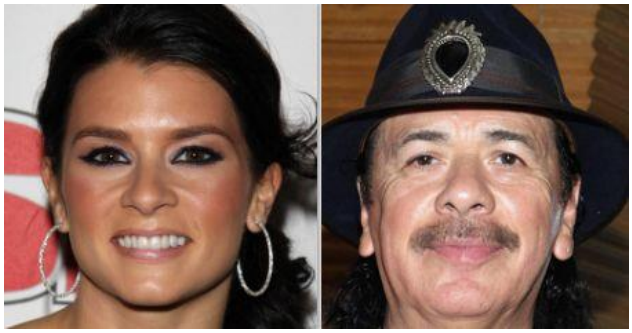

Figure 136: Examples of face images with no head coverings (left) and with head coverings (right) from the test set CelebAMask-HQ-V2.

The number of images and labels in the test sets is summarised in *Table 20*.

Table 20: Number of images in the test sets used for No Head Coverings.

Test Set	No Head Covering	Head Covering
CAS-PEAL-R1	1,039	1,308
CelebAMaskHQ-V2	1,073	1,018

It should be noted that the CNN used for face parsing (see Section 4.3.3.3), on which the implemented algorithms rely, was trained on the CelebAMaskHQ database from which the test set CelebAMaskHQ-V2 has been selected.

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.12.2 Selection and Prototyping of Candidate Algorithms

The implemented algorithms rely on the face parsing algorithm described in Section 4.3.2.1.

7.12.2.1 CD1 Algorithm

The first candidate algorithm was taken from ISO/IEC CD1 29794-5:2023 [6]. It determines the proportion of pixels in the face parsing segmentation map that are labelled as “hat”.

The algorithm takes as input the face parsing segmentation map *M* described in Section 4.3.2.1, and performs the following steps:

1. Count the number *n* of pixels in *M* with label 18 (“hat”).
2. Output *n*/*m,* where *m* is the number of pixels in *M*.

7.12.2.2 CD3 Algorithm

In ISO/IEC CD3 29794-5, the algorithm was improved:

- Only pixels in the area above the eyes are considered. The y-coordinates of the eyes is approximated based on their target coordinates used by the alignment algorithm.
- In addition to the label 18 (“hat”), also the label 16 (“clothing”) is considered to account for hoods and headscarves which are sometimes labelled by the face parsing algorithm as clothing.

The resulting algorithm takes as input the face parsing segmentation map *M* described in Section 4.3.2.1, and performs the following steps:

1. Crop M from the bottom by 204 pixels.
2. Count the number *n* of pixels in *M* having either label 16 (“clothing”) and label 18 (“hat”).
3. Output *n*/*m,* where *m* is the number of pixels in *M*.

7.12.3 Evaluation

The DET curves for both algorithms, referred to as CD1 and CD3, respectively, on the modified test set CelebAMaskHQ-V2 and on the test set CAS-PEAL-R1 are shown in Figure 137. Slightly better performance rates are achieved compared to the CD3 algorithm with respect to false positives.

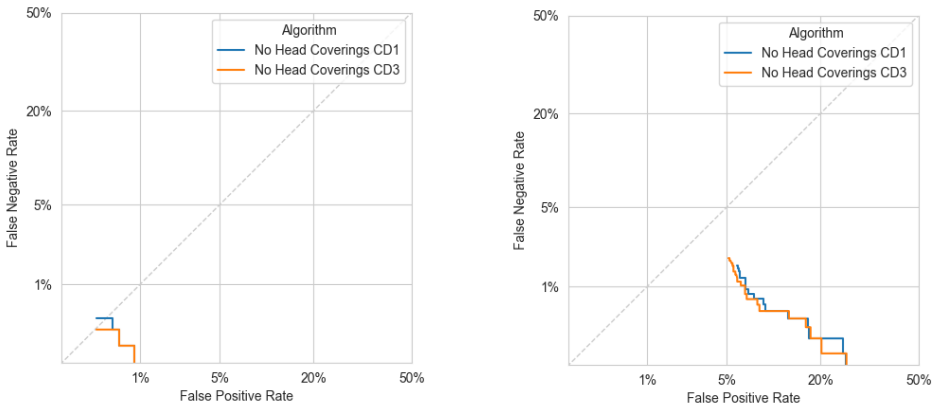

Figure 137: DET curve of the algorithms for the quality component No Head Coverings on the test sets CAS-PEAL-R1 (left) and CelebAMask-HQ-V2 (right).

The EDC curves of the algorithms are shown in Figure 138. As expected from the DET curves, it can be observed that for relevant discard rates, the obtained results are similar.

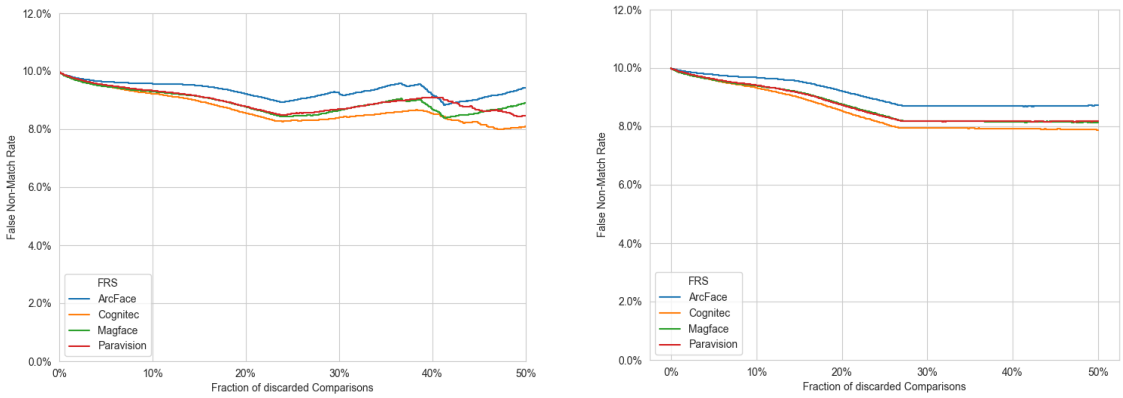

Figure 138: EDC curves for the quality component No Head Coverings using the CD1 (left) and CD3 (right) algorithms on the VGGFace2 test set.

7.12.4 Final Algorithm Selection

For the quality component No Head Covering, the CD3 algorithm specified in Section 7.12.2.2 was selected and implemented in OFIQ.

For the mapping of the native quality measure q to the quality component value Q in the target range [0,100], the following procedure is used, where ROUND and SIGMOID are defined as described in Section 5.4:

1. Let $T_0 = 0$, $T_1 = 0.95$
2. If $q \leq T_0$ set Q=100
3. If $q \geq T_1$ set Q=0
4. Otherwise, set Q = ROUND(100 $h(q)$ / $h(T_0)$) with

   $h(x)$ = SIGMOID(T_1, 0.02, 0.1) - SIGMOID(x, 0.02, 0.1)

7.13 Motion Blur Prevention

7.13.1 Data Selection

7.13.1.1 Training Set

For the training of a machine learning algorithm for motion blur estimation, a sufficiently large training and validation data set with numeric labels (specifying the displacement of the head from the beginning to end of the motion, measured in pixels) is needed. Since manual annotation of a large number of real images with such labels is too time consuming, we generated semi-synthetic data. To this end, we applied linear motion blur with random kernel sizes between 1 and 29 and random directions (with an integer angle value uniformly chosen from [0,179]) on 529 sharp images from the FERET database [39] and 231 reasonably sharp images of the FRGCv2 database [34]. In total, 4,979 synthetically motion-blurred images were generated for the images from FERET and 4,109 for the images from FRGCv2, resulting in 9,088 images with synthetic motion blur. Note, that the simulated motion blurs are only linear, while in reality non-linear motion blur (accelerated and potentially not straight) motion may occur. Example images of synthetic motion blurs are shown in following figure.

Figure 139: Example images of synthetic motion blur with kernel size / direction, from left to right: 8/137°, 16/126°, 19/157° and 22/85°.

As numeric labels for the images with synthetic motion blur, we use the size of the applied kernel, which equals the distance in pixels over which the face is smudged in the image. This choice of the labels is in accordance with the definition used in the NIST FATE Quality SIDD track [16]: the extent of motion blur is defined as the displacement of the head from beginning to end of the motion, measured in pixels.

We complement these semi-synthetic motion-blurred images by a large variety of images without motion blur:

- 1,364 full-frontal sharp images from the FERET database, aligned according to the ICAO requirements for passport images and resized to an image height of 900.
- The 1,363 sharp images from FERET (see previous bullet point) with Gaussian blur filter of randomly chosen (odd) size between 15 and 79 applied.
- The 1,363 blurred images from FERET (see previous bullet point) with a 3x3 sharpening filter applied.[49]
- 703 reasonably sharp images from the FRGCv2 database.
- 531 images from FRGCv2 with considerably defocus.

[49] For sharpening, a convolutional 3x3 filter with centre value set to 5 and the other values in the centre row and column set to -1 is used. The motivation of sharpening some of the images was to introduce artefacts similar to those that might occur due to camera noise or postprocessing.

- 200 unprocessed images from the FEI database [73].
- 131 unprocessed images from the Utrecht ECVP database [74].
- 285 unprocessed images from the H-BRS multispectral SWIR/RGB Database [75].
- 284 images from the H-BRS multispectral SWIR/RGB Database with a 3x3 sharpening filter applied.
- 1,498 unprocessed images from the Flickr-Faces-HQ Dataset (FFHQ) [18]. 998 of these images were chosen from the images with the highest feature values, while the remaining 500 images were chosen at random.
- The 998 images from FFHQ chosen from the images with the highest feature values with a 3x3 sharpening filter applied.
- 539 unprocessed images from CelebA-HQ [19], chosen from the images with the highest feature values and manually filtered for visible artefacts resulting from the super resolution CNN.
- 1,277 unprocessed sharp images captured in e-gates and self-enrolment kiosks.
- 597 of the images captured in e-gates (see previous bullet point) with a 3x3 sharpening filter applied.

For all of these 11,133 training images, the labels were set to 0 (indicating no motion blur). Together with the images with synthetic motion blur, this results in a training set of 20,221 images.

7.13.1.2 Test Sets

The first test set *internal* was compiled from 365 face images captured from volunteering employees at secunet using secunet's easykiosk and easygate systems:

- 134 face images with motion blur, where the acquired subjects were moving during the capturing process,
- 124 sharp images and
- 107 images showing considerable out-of-focus blur but no motion blur.

None of the images was contained in the training set. Example images containing motion blur due to the movement of subjects are illustrated in Figure 140.

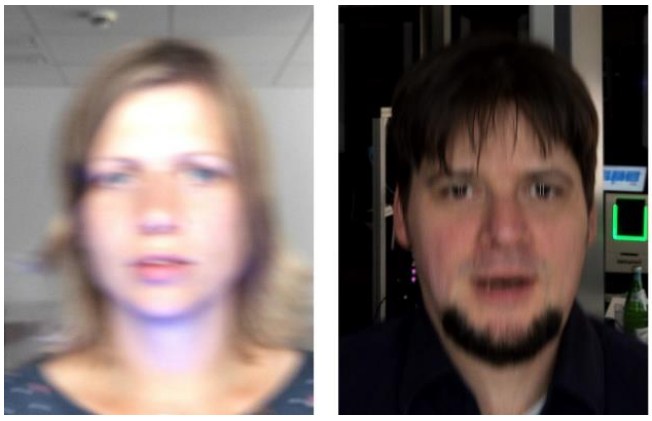

Figure 140: Example images with motion blur (cropped to the facial region) of the internal test dataset.

A second test set *HDA-Motion-Blur* was compiled in cooperation with Hochschule Darmstadt. Using 3 different cameras (Kodak DX6490, Canon EOS 50D and Canon PowerShot SX200 IS) 1341 images were captured of 35 subjects using a uniform background. For 1171 images, the subjects were asked to move their head during capture, and for 170 images, they were required to stand still. The images were manually

labelled with a motion blur score ranging – in steps of 10 – from 10 (meaning extreme motion blur) to 90 (denoting no motion blur). Example images are shown in Figure 141.

For the evaluation, the 96 images with score 80 were excluded as the motion blur therein was quite limited, and all other images were labelled with binary labels *Motion Blur* (images with score ≤ 70) and *No Motion Blur* (score = 90).

Figure 141: Images from the test set HDA-Motion-Blur with motion blur score 80 (left), 50 (middle) and 10 (right).

For evaluations with EDC curves, we use the same subset of VGGFace2 as for evaluation of the Unified Quality Score (see Section 5.1).

7.13.2 Selection and Prototyping of Candidate Algorithms

The selected algorithm is based on the same general image processing methods as the algorithms for Sharpness specified in Section 6.6.2.2. Precisely, a regression algorithm is trained to estimate the amount of motion blur measured as the number of pixels of the visible movement of the head based on features computed with edge detection filters and from the difference of the image with its blurred variant. In contrast to the algorithms for Sharpness specified in Section 6.6.2.2, un-isotropically applied Sobel filters are used to capture directional blur. The selection of the features has been inspired by the algorithms proposed for (non-motion) blur detection in ISO/IEC WD5 29794-5:2022 [3].

First, the same pre-processing is applied to the image as in the algorithms specified in Section 6.6.2.2, i.e. the image is cropped to the minimal bounding box of the landmarks, scaled so that its longer side is 250 and converted to grey scale. Furthermore, the landmarked region is computed as segmentation mask for feature computation (details below).

Then the following features are computed of the cropped grayscale image I:

- **Horizontal Sobel filter.** For each of the kernel sizes s=1,3,5, the horizontal Sobel filter S_s^{x} is computed on the pre-processed image using the OpenCV function Sobel() with parameters dx=1, dy=0 and ksize=s. For each s, the arithmetic mean and standard deviation of the absolute values of the outputs in the landmarked region R are computed as feature, i.e. $\mu\left(\left|S_s^{\mathrm{x}}(I_{i,j})\right| \mid R_{i,j}=1\right)$ and $\sigma\left(\left|S_s^{\mathrm{x}}(I_{i,j})\right| \mid R_{i,j}=1\right)$ for s=1,3,5.
- **Vertical Sobel filter.** For each of the kernel sizes s=1,3,5, the vertical Sobel filter S_s^{y} is computed on the pre-processed image using the OpenCV function Sobel() with parameters dx=0, dy=1 and ksize=s. For each s, the arithmetic mean and standard deviation of the absolute values of the outputs in the landmarked region R are computed as feature, i.e. $\mu\left(\left|S_s^{\mathrm{y}}(I_{i,j})\right| \mid R_{i,j}=1\right)$ and $\sigma\left(\left|S_s^{\mathrm{y}}(I_{i,j})\right| \mid R_{i,j}=1\right)$ for s=1,3,5.

- **Horizontal Sobel filter on rotated image.** The horizontal Sobel filter features are also computed on the image I', resulting from a rotation of image I by 45°. For each s, the arithmetic mean and standard deviation of the absolute values of the outputs in the landmarked region are computed as feature, i.e. $\mu\left(\left|S_s^{\mathrm{x}}(I'_{i,j})\right| \mid R_{i,j}=1\right)$ and $\sigma\left(\left|S_s^{\mathrm{x}}(I'_{i,j})\right| \mid R_{i,j}=1\right)$ for s=1,3,5.
- **Vertical Sobel filter on rotated image.** The vertical Sobel filter features are also computed on the image I', resulting from a rotation of image I by 45°. For each s, the arithmetic mean and standard deviation of the absolute values of the outputs in the landmarked region are computed as feature, i.e.$\mu\left(\left|S_s^{\mathrm{y}}(I'_{i,j})\right| \mid R_{i,j}=1\right)$ and $\sigma\left(\left|S_s^{\mathrm{y}}(I'_{i,j})\right| \mid R_{i,j}=1\right)$for s=1,3,5.
- **Laplacian filters.** First, a Gaussian blur filter G_3 with 3x3 kernel is applied to the pre-processed image to remove noise. Then, for each of the kernel sizes s=1,3,5,7,9, the Laplacian Filter L_s with kernel size s is computed on the blurred image using the OpenCV function Laplacian() with parameter ksize=s. For each s, the arithmetic mean and standard deviation of the absolute values of the outputs in the landmarked region R are computed as feature, i.e. $\mu\left(|L_s(G_3(I))| \mid R_{i,j}=1\right)$ and $\sigma\left(|L_s(G_3(I))| \mid R_{i,j}=1\right)$ for s=1,3,5,7,9.
- **Difference to mean-filtered image.** This feature is based on the method defined in ISO/IEC WD5 29794-5:2022 [3]. For each of the kernel sizes s=3,5,7 a mean filter M_s (OpenCV function cv2.blur) is applied to the image and the resulting image is subtracted from the original one. The mean and standard deviation of the absolute values of this difference in the landmarked region R, i.e., $\mu\left(\left|I_{i,j}-M_s(I)_{i,j}\right| \mid R_{i,j}=1\right)$ and $\sigma\left(\left|I_{i,j}-M_s(I)_{i,j}\right| \mid R_{i,j}=1\right)$, are returned as feature value.

In summary, this results in 34 features, which are used to train a machine learning (regression) algorithm to infer the extent of motion blur. The regression algorithm is trained on the features and numeric labels of the semi-synthetic training and validation data set described beforehand.

For all 20,221 images of the training data, these features were extracted, and the numeric labels (indicating the motion length) were scaled with the factor used for resizing of the images during pre-processing. As regression algorithms, Support Vector Machine (SVM) and Random Forest were implemented in Python using the Machine Learning module of OpenCV (Classes *ml::SVM* and *ml::RTree*). For both regression algorithms, the available hyperparameters (including the kernel type of the SVM) were optimised sequentially using a 3-fold subsampling where 70% of the data was used for training and 30% for testing in each iteration. As evaluation metric, the Mean Squared Error (MSE) was used. For SVM, the features and labels of the training set were scaled to mean 0 and standard deviation 1; using the scaling parameters determined on the training set, the features of the test set were scaled and the outputs of the SVM were un-scaled (because the SVM was trained to output scaled labels).

The following hyperparameters were found as optimal:

- For SVM: SVM type *C-Support Vector classification*, kernel="rbf" (radial basis function), EPS-regression with gamma=0.6, C=100, P=0.01 and TermCriteria = (2,0,0.5). This results in an average mse of 1.8.
- For Random Forest: ActiveVarCount=3, MinSampleCount=12, MaxDepth=60 and TermCriteria=(2,0,0.1). This results in an average mse of 3.5.

Hence, it was decided to use an SVM with the aforementioned parameters. To this end, the SVM was trained on the scaled features and labels of the complete training set. The trained model and the scaling parameters (for features and labels) were saved to files.

The resulting algorithm takes as input the image I (un-aligned) in BGR colour channel order with 8 bits per channel and the landmarks L output by any of the landmark extraction algorithms in Section 4.2.1 . For initialisation, the trained SVM model and the scaling parameters for the features and the labels (computed on the training set) are loaded from file. Then the following steps are executed:

1. Using the landmarks L, compute the landmarked region segmentation mask R using the algorithm specified in Section 4.3.1.1 with parameter α=0.

2. Crop *I* and *R* to the minimal upright bounding box of the landmarked region (pixels *i,j* with $R_{i,j}$=1).
3. Scale *I* and *R* so that the longer side is 250, i.e. by a factor q, where $q = 250/\max(w, h)$ and *w, h* are the width and height of the cropped image, respectively.
4. Convert *I* to grey scale.
5. Compute the features on *I* restricted to *R* as described above to obtain a 34-dimensional feature vector *f*; the mean and standard deviation of the filter outputs and mean difference values is computed on the landmarked region only.
6. Scale *f* with the feature scaling parameters m_1, …, m_{34} (mean values) and s_1, …, s_{34}) (standard deviation values) by setting $f'_i = (f_i - m_i)/s_i$.
7. Feed the scaled feature vector *f'* into the trained SVM model to obtain a scaled prediction value *x'*.
8. Un-scale *x'* using the label scaling parameters m_0 (mean value) and s_0 (standard deviation value) by setting $x = x' \cdot s_0 + m_0$
9. Compute the dislocation estimation *d* by scaling *x* to the original image size: $d = x/q$.
10. Output *d*.

Further, an algorithm based on Cepstrum analysis [76] was implemented. The use of Cepstrum analysis for the detection and estimation of motion blur was first proposed in [77].

The algorithm uses the parameters $\sigma_{\text{high}} = 2.5$, $\sigma_{\text{low}} = 1.5$, *w*=80 and *s*=40, and takes as input the aligned image *I* and the transformed landmarks *L'* output by the algorithm in Section 4.4.1, and performed the following steps:

1. Determine a rectangle (given by its upper left corner (a,b) and its lower right corner (c,d)) within the facial region as follows:
    a. Set *b* to the minimum y-coordinate of all landmarks
    b. Set *d* to the maximum y-coordinate of the outer mouth contour landmarks 78-87.
    c. From the landmarks 0-32 marking the face contour, select the subset *U* of landmarks having an y-coordinate smaller than *b*+12.
    d. Compute X as the mean x-coordinate of all landmarks in *U*.
    e. Set U_{l} to the set of landmarks from *U* having x-coordinate smaller than X, and U_{r} to the set of landmarks in *U* having x-coordinate greater than X.
    f. Set *a* to the x-coordinate of the landmark in U_{l} having the highest y-coordinate.
    g. Set *c* to the x-coordinate of the landmark in U_{r} having the highest y-coordinate.
2. Crop *I* to the rectangle given by (*a*,*b*,*c*,*d*).
3. If $w > d - b$ or $w > c - a$, set $w = \min(d - b, c - a)$.
4. If $s < (d - b)/2$ or $s < (c - a)/2$, set $s = \min(d - b, c - a)/2$.
5. Perform the following steps to apply a Difference of Gaussian (DoG) on *I* to obtain an edge array *E*:
    a. Apply a Gaussian Blur filter with kernel size $8 \cdot \sigma_{\text{low}} + 1$ to *I* to obtain a blurred image I_1.
    b. Apply a Gaussian Blur filter with kernel size $8 \cdot \sigma_{\text{high}} + 1$ to *I* to obtain a blurred image I_2.
    c. Set $E = I_1 - I_2$, where *E* is represented as an array of floating-point values.
6. Compute a Hann window *H* of the same size as *E.*
7. Multiply *E* with *H* to obtain a masked edge array E'.

8. For all $0 \leq i \leq \lfloor (d-b-w)/s \rfloor + 1$ and for all $0 \leq j \leq \lfloor (c-a-w)/s \rfloor + 1$, execute the following steps:
    a. Select the subarray $A(i,j)$ of E' of size $w \times w$ ranging from the upper left corner $(i \cdot s + 1,\ j \cdot s + 1)$ to the lower left corner $(i \cdot s + w,\ j \cdot s + w)$.
    b. Compute the 2-dimensional discrete Fourier Transform $F(i,j)$ of $A(i,j)$.
9. Set $\bar{F}$ to the component-wise mean absolute value of all $F(i,j)$, i.e. set $\bar{F}_{x,y} = \sum_{i,j} |F(i,j)_{x,y}|$ for all $1 \leq x, y \leq w$, where the absolute value of a complex number is defined as $|x + iy| = \sqrt{x^2 + y^2}$.
10. Compute the power spectrum P as the element-wise natural logarithm of $\bar{F}$.
11. Compute the 2-dimensional inverse discrete Fourier Transform D of P.
12. Obtain the Cepstrum C by shifting D so that the centre corresponds to the "quefrency"[50] 0.
13. Obtain C' by setting all positive values in C to zero, i.e. set $C'_{p,q} = \min(0, C_{p,q})$ for all p, q.
14. For $n = \lfloor \sqrt{w^2/2} \rfloor$, compute a monotonously decreasing sequence $S_0, \ldots, S_n$, where S_r is the sum of all $C'_{i,j}$ having a distance greater or equal than r to the centre $(w/2, w/2)$ of C':

$$S_r = \sum_{i,j:\ \left\| (i,j) - (\frac{w}{2}, \frac{w}{2}) \right\|_2 \geq r} C'_{i,j} \text{ for } 0 \leq r \leq \lfloor \sqrt{w^2/2} \rfloor.$$

15. Compute the sequence D negative derivatives of S as $D_r = S_r - S_{r+1}$, for $0 \leq r \leq n-1$.
16. Find the index k of the most prominent peak in D using the Scipy function find_peaks[51].
17. Return k+1 as an estimation of the motion blur length.

7.13.3 Evaluation

The evaluation on the internal test set reveal mixed results, see Figure 142. When distinguishing images with motion blur from sharp images, Cepstrum Analysis achieves a moderate detection performance associated with an EER of approximately 20%. In contrast, the SVM-based classifier achieves a good EER of 2.5%. When trying to distinguish the images with motion blur from images with de-focus blur in the internal test set, for both the SVM based classifier and the Cepstrum Analysis, the EER is close to 50%, which is equivalent to guessing.

[50] The quefrencies of the cepstrum correspond to the frequencies of the Frourrier Transform.
[51] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

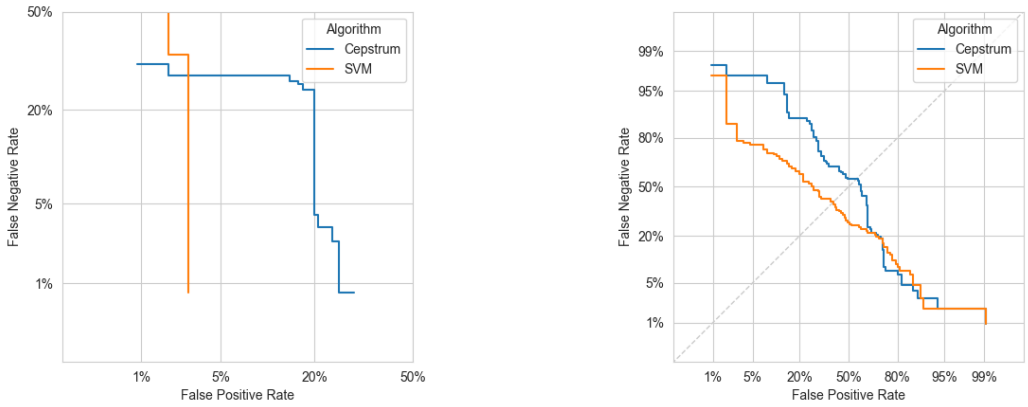

Figure 142: DET curves of the quality component Motion Blur Prevention on the internal test set when using sharp images as (left) and images with de-focus as negative samples (right). Note, that the right plot uses a larger scale.

The detection performance results on the HDA-Motion-Blur test set are depicted in Figure 143. Like on the other test set, the SVM-based classifier significantly outperforms the Cepstrum analysis approach. Here, an EER of about 10% and 24% are achieved for the SVM-based classifier and the Cepstrum analysis, respectively.

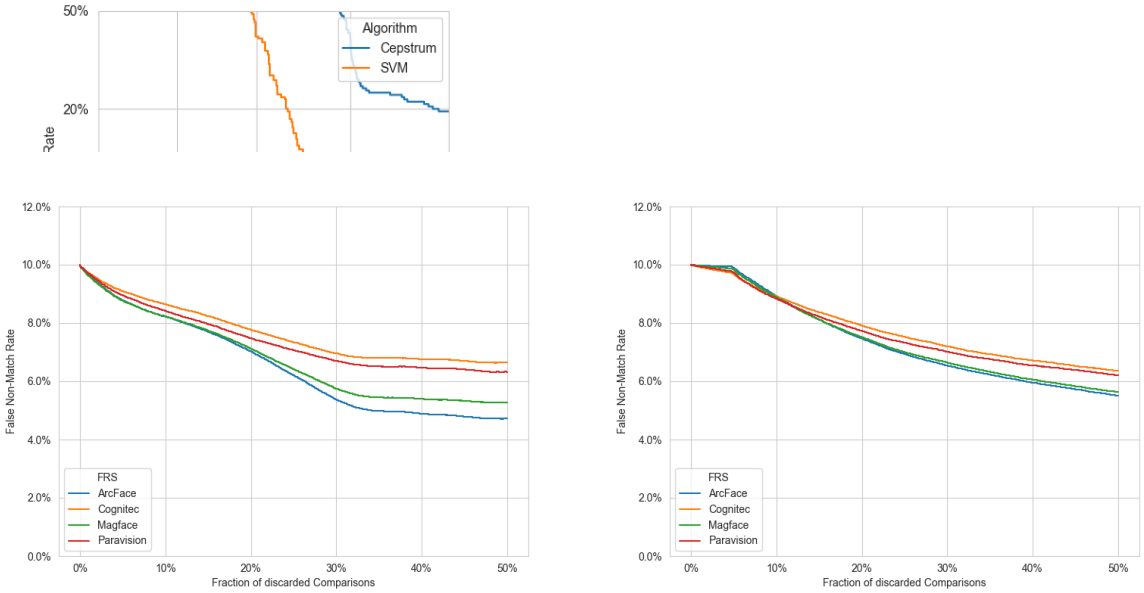

Fig
Blu

Figure 144: EDC curves of the quality component Motion Blur Prevention using the Cepstrum analysis algorithm (left) and the SVM-based classifier algorithm (right) on the VGGFace2 test set.

The corresponding EDC curves are depicted in Figure 144. Interestingly, the method based on Cepstrum analysis achieves greater reductions in terms of false non-match rate are achieved, in particular for small discard fractions, e.g. 10%. For the method based on the SVM classifiers, no significant reduction in false non-match rate is achieved until a discard rate of about 5%.

The algorithm based on Cepstrum Analysis was implemented in the submission secunet_003 to the NIST FATE Quality SIDD track [16]. The test set consists of images with synthetically generated motion blur of various amounts. As visible in Figure 145, the algorithm doesn't show a good detection performance.

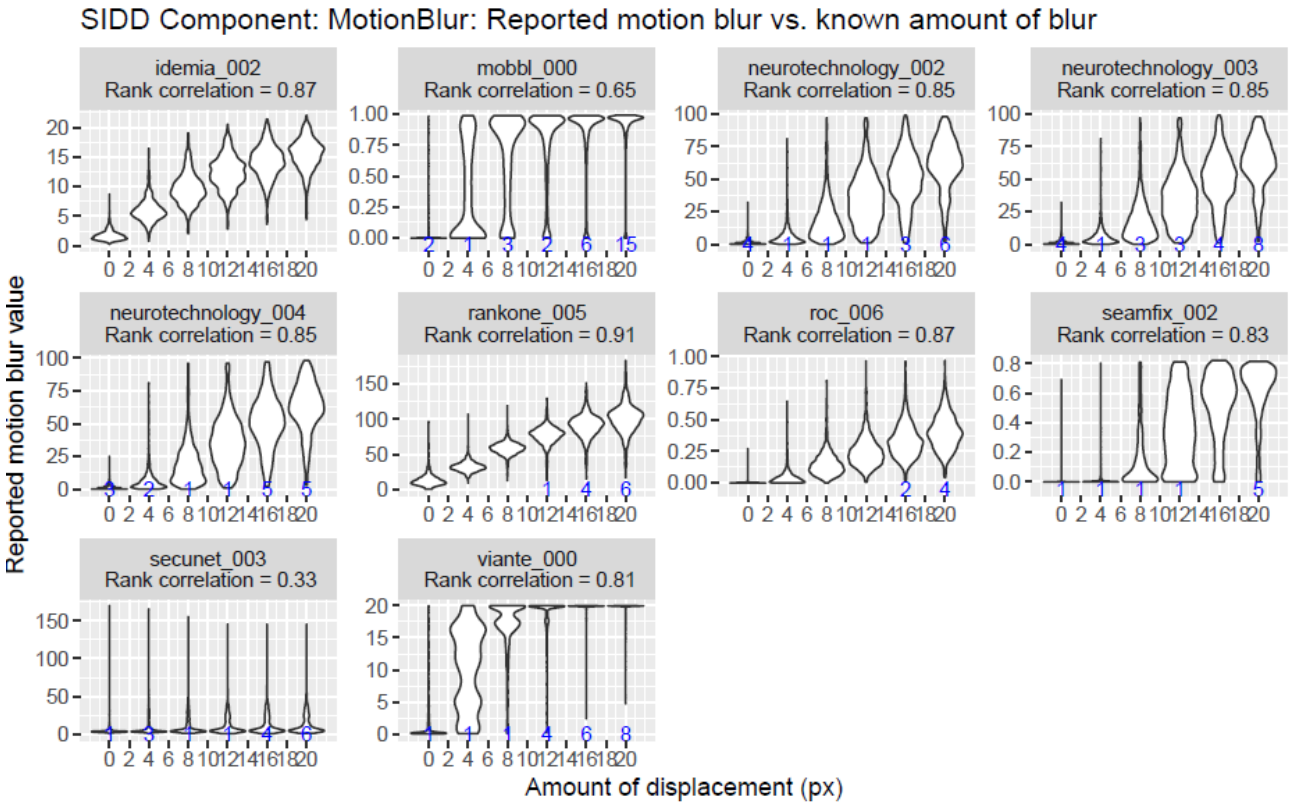

Figure 145: Distribution of the outputs of the algorithms for the component Motion Blur in the NIST FATE Quality SIDD report [16] per amount of (synthetic) motion blur. The algorithm based on Cepstrum Analysis is labelled as secunet_003.

7.13.4 Final Algorithm Selection

The quality component Motion Blur Prevention was discarded in ISO/IEC DIS 29794-5:2024 [9] and in OFIQ, because no sufficiently effective algorithm was available.

8 Summary and Outlook

The OFIQ framework provides an open-source reference implementation of the standard ISO/IEC 29794-5, consisting of state-of-the-art algorithms that allow commercial use. The ISO/IEC 29794-5 standard was developed by international experts in the field of face recognition. In parallel, OFIQ was implemented and findings obtained in two prototyping and evaluation phases were fed into the corresponding standard, which was co-authored by OFIQ developers. The OFIQ framework contains algorithms to assess the unified quality of a face image as well as capture- and subject-related quality components that have been identified as relevant for face image quality assessment by international experts. This set of quality components makes it possible to provide explainable and actionable feedback to users and operators of face recognition systems.

The OFIQ framework fulfils requirements relevant for practical applications, such as platform compatibility. Throughout the development of OFIQ, resource consumption has been minimised. Hence, OFIQ is expected to become a de-facto standard that allows a transparent transnational assessment of face image quality. This is of utmost importance for large-scale face recognition systems like in the upcoming EES.

Future works towards improving OFIQ may focus on several relevant aspects. On the one hand, quality components that have not been integrated in the current version of OFIQ may be added, e.g. Motion blur prevention. On the other hand, future advances related to the algorithms contained in OFIQ can be leveraged for improvements. To facilitate the use of OFIQ in environments with limited computing power, a lightweight version of OFIQ should be provided. This can be achieved by further optimizing current algorithms achieving a reasonable trade-off between accuracy and computational workload. Finally, the algorithms contained in OFIQ need to be thoroughly analysed with respect to fairness, i.e., for demographic bias, in order to avoid potential discrimination of certain groups. Some results have already shown a demographic bias of OFIQ algorithms [78], [79].

9 Glossary

Term	Description
DET	Detection Error Trade-off
Detection Error Trade-off	A detection error trade-off plot is a graphical plot of error rates for binary classification systems, plotting the false rejection rate vs. false acceptance rate
ECDF	Empirically Cumulated Distribution Function
EDC	Error-versus-Discard Characteristics
Empirically Cumulative Distribution Function	The cumulative distribution function of a real-valued variable *X* evaluated at *x*, is the probability that *X* will take a value less than or equal to *x*.
Error-versus-Discard Characteristics	The false non-match error vs discard-ratio characteristic is a graphical presentation of the performance of quality assessment algorithms, plotting the dependence of the false non-match rate at a fixed comparison decision threshold on the percentage of low-quality reference and probe samples discarded.
Face Detection	The task of finding and localising faces in images.
Facial Landmark	Key point (salient point) specifying the location of a certain face part, e.g. a corner of the eye.
Face Segmentation	The task of separating the parts in the image where the face and, optionally, individual parts and accessories of the face, e.g. nose, mouth, eyes, ears, hair, beards, sunglasses, ear rings, are visible.
FNMR	False Non-Match Rate
FRVT	Face Recognition Vendor test. A program of the National Institute for Standards and Technology (NIST) aiming to evaluate algorithms for face biometrics. FRVT comprises several projects, each of which focussing on a certain task for which the algorithms are evaluated. One of these projects is FRVT Quality, focussing on algorithms assessing the quality of facial images (https://pages.nist.gov/frvt/html/frvt_quality.html).
FTX	Failure-to-Extract
Failure-to-Extract	An error occurred when processing a biometric sample such that no biometric features could be extracted
IED	Inter-Eye Distance
MAE	Mean Absolute Error
SIDD	Specific Image Defect Detection. A track within NIST FATE Quality evaluating the ability of facial image QA algorithms to detect certain quality issues. See https://pages.nist.gov/frvt/api/FRVT_ongoing_quality_sidd_api.pdf
Subject Segmentation	The task of separating the parts in the image where the subject and, optionally, individual parts and accessories of the subject, e.g. face, neck, hair, sunglasses, hats, clothing, are visible.

10 Bibliography

[1] ISO/IEC, “39794-5: Extensible biometric data interchange formats - Part 5: Face image data,” 2019.

[2] ISO/IEC, “1st WD 29794-5 - Biometric sample quality - Part 5: Face image,” 2020.

[3] ISO/IEC, “4th WD 29794-5 Biometric sample quality - Part 5: Face image data,” 2022.

[4] ISO/IEC, “5th WD 29794-5 - Biometric sample quality - Part 5: Face image data,” 2022.

[5] ISO/IEC, “6th WD 29794-5 - Biometric sample quality - Part 5: Face image data,” 2023.

[6] ISO/IEC, “1st CD 29794-5 - Biometric sample quality - Part 5: Face image data,” 2023.

[7] ISO/IEC, “2nd CD 29794-5: Biometric sample quality - Part 5: Face image data,” 2023.

[8] ISO/IEC, “3rd CD 29794-5 - Biometric sample quality - Part 5: face image data,” 2023.

[9] ISO/IEC, “DIS 29794-5 - Biometric sample quality - Part 5: Face image data,” 2024.

[10] ISO/IEC, “FDIS 29794-5 - Biometric sample quality - Part 5: Face image data,” 2024.

[11] ISO/IEC, „19794-5:2011 - Biometric data interchange formats - Part 5: Face image data,“ 2011.

[12] ISO/IEC, „TR 29794-5 Biometric sample quality - Part 5: Face image data,“ 2010.

[13] Federal Office for Information Security, „BSI Technical Guideline TR-03121-3 - Biometrics for Public Sector Applications - Part 3: Application Profiles, Function Modules and Processes - Volume 1: Border Control (BCL) - Version 6.0,“ 2021.

[14] J. Merkle, C. Rathgeb, B. Tams, D.-P. Lou, A. Dörsch und P. Drozdowski, „Facial Metrics for EES (EESFM) - State of the Art of Quality Assessment of Facial Images,“ arXiv 2211.08030, Version 1.2, 2022.

[15] Q. Cao, L. Shen, W. Xie, O. M. Parkhi and A. Zisserman, “VGGFace2: A dataset for recognising faces across pose and age,” in *Face & Gesture (FG)*, 2018.

[16] J. Yang, P. Grother, M. Ngan, K. Hanaoka and A. Hom, “Face Analysis Technology Evaluation (FATE) Part 11: Face Image Quality Vector Assessment - Specific Image Defect Detection,” National Institute of Standards and Technology, NIST IR 8485, April, 2024.

[17] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran and M. Grundmann, “BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs,” *CORR,* vol. abs/1907.05047, 2019.

[18] J. Deng, J. Guo, Y. Zhou, J. Yu, I. Kotsia and S. Zafeiriou, “RetinaFace: Single-stage Dense Face Localisation in the Wild,” *CORR,* vol. abs/1905.00641, 2019.

[19] X. Zhu, Z. Lei, X. Liu, H. Shi and S. Li, “Face Alignment Across Large Poses: A 3D Solution,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA,, 2016.

[20] V. Kazemi and J. Sullivan, “One Millisecond Face Alignment with an Ensemble of Regression Trees,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, 2014.

[21] Y. Kartynnik, A. Ablavatski, I. Grishchenko and M. Grundmann, “Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs,” *CoRR,* vol. abs/1907.06724, 2019.

[22] Y. Huang, H. Yang, C. Li, J. Kim and F. Wei, “ADNet: Leveraging error-bias towards normal direction in face alignment,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[23] National Institute of Standards and Technology, “Face Analysis Technology Evaluation (FATE) Quality Assessment - Specific Image Defect Detection,” 04-23-2024. [Online]. Available: https://pages.nist.gov/frvt/api/FRVT_ongoing_quality_sidd_api.pdf. [Accessed 09-23-2024].

[24] C.-H. Lee, Z. Liu, L. Wu and P. Luo, “MaskGAN: Towards Diverse and Interactive Facial Image Manipulation,” in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020.

[25] X. Yin, D. Huang and L. Chen, “FaceOcc: A Diverse, High-quality Face Occlusion Dataset for Human Face Extraction,” in *Traitement et Analyse de l’Information Méthodes et Applications (TAIMA'2022)*, Hammamet, Tunisia. , 2022. https://arxiv.org/abs/2201.08425, Code: https://github.com/face3d0725/FaceExtraction.

[26] X. Yin, D. Huang and L. Chen, “Non-Deterministic Face Mask Removal Based on 3D Priors,” in *2022 IEEE International Conference on Image Processing (ICIP)*, Bordeaux, 2022. Video presentation: https://www.youtube.com/watch?v=pspJsAq8rww.

[27] X. Yin, D. Huang, Z. Fu, Y. Wang and L. Chen, “Segmentation-Reconstruction-Guided Facial Image De-occlusion,” in *17th IEEE Intl. Conference on Automatic Face and Gesture Recognition 2023 (FG’2023)*, Waikoloa Beach, HI, USA, 2023. Video presentation: https://www.youtube.com/watch?v=meQHBwWM2i0.

[28] X. Yin, D. Huang, Z. Fu, Y. Wang and L. Chen, “Weakly-Supervised Photo-realistic Texture Generation for 3D Face Reconstruction,” in *IEEE 17th International Conference on Automatic Face and Gesture Recognition (FG)*, Waikoloa Beach, HI, USA, 2023. Video presentation: https://www.youtube.com/watch?v=PPdLKDI-xyk.

[29] P. J. Rousseeuw, “Least median of squares regression,” *Journal of the American statistical association,* vol. 79, no. 388, pp. 871-880, 1984.

[30] “Levenberg–Marquardt algorithm - Wikipedia,” Wikimedia Foundation, Inc., 02-28-2023. [Online]. Available: https://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm. [Accessed 03-09-2023].

[31] ISO/IEC, “29794-1 - Biometric sample quality - Part 1: Framework,” 2024.

[32] Q. Meng, S. Zhao, Z. Huang and F. Zhou, “MagFace: A Universal Representation for Face Recognition and Quality Assessment,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[33] M. Kim, A. K. Jain and X. Liu, “AdaFace: Quality Adaptive Margin for Face Recognition,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[34] P. Phillips, P. Flynn, T. Scruggs, K. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min and W. Worek, “Overview of the face recognition grand challenge,” in *Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[35] A. M. Martinez and R. Benavente, “The AR Face Database. CVC Technical Report #24,” 1998.

[36] R. Gross, I. Matthews, J. Cohn, T. Kanade and S. Baker, “Multi-PIE,” in *8th IEEE International Conference on Automatic Face & Gesture Recognition*, Amsterdam, Netherlands, 2008.

[37] H. Zhang, M. Grimmer, R. Raghavendra, K. Raja and C. Busch, "On the Applicability of Synthetic Data for Face Recognition," in *International Workshop on Biometrics and Forensics (IWBF 2021)*, Rome, Italiy, 2021.

[38] W. Gao, B. Cao, S. Shan, D. Zhou, X. Zhang and D. Zhao, "The CAS-PEAL Large-Scale Chinese Face Database and Baseline Evaluations," 2004.

[39] J. Phillips, H. Wechsler, J. Huang and P. J. Rauss, "The FERET database and evaluation procedure for face-recognition algorithms," *Image and Vision Computing,* vol. 16, no. 5, pp. 295-306, 1998.

[40] A. Horé and D. Ziou, "Image Quality Metrics: PSNR vs. SSIM," in *20th International Conference on Pattern Recognition*, 2010.

[41] L. Jonientz, "Detektion von Kompressionsartefakten in biometrischen Gesichtsbildern mittels neuronaler Netze," Department of Computer Science and Media of the Brandenburg University of Applied Science, Bachelor Thesis, 2023.

[42] K. Xiao, J. M. Yates, F. Zardawi, S. Sueeprasan, N. Liao, L. Gill, C.Li and S. Wuerger, "Characterising the variations in ethnic skin colours: a new calibrated data base for human skin," *Skin Research and Technology,* vol. 23, no. 1, 2017.

[43] Y. C. Shih, W.-S. Lai and C.-K. Liang, "Distortion-free wide-angle portraits on camera phones," *ACM Trans. Graph.,* vol. 38, no. 4, 2019.

[44] H. Zhao, X. Ying, Y. Shi, X. Tong, J. Wen and H. Zha, "DCFace: Radial Distortion Correction for Face Recognition," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[45] C.-H. Chao, P.-L. Hsu, H.-Y. Lee and Y.-C. F. Wang, "Self-Supervised Deep Learning for Fisheye Image Rectification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[46] X. Li, B. Zhang, P. V. Sander and J. Liao, "Blind Geometric Distortion Correction on Images Through Deep Learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[47] T. L. Pessot, "Detection of distortion in facial images," NTNU, 2023.

[48] F. Song, X. Tan, X. Liu and S. Chen, "Eyes Closeness Detection from Still Images with Multi-scale Histograms of Principal Oriented Gradients," *Pattern Recognition,* vol. 47, no. 9, 2014.

[49] Intelligent Systems Laboratory (ISL) at the Rensselaer Polytechnic Institute (RPI), "RPI ISL Eye Database," 12-26-2008. [Online]. Available: https://sites.ecse.rpi.edu/~cvrl/database/ISL_IR_Eye_Database.htm. [Accessed 06-09-2023].

[50] A. Fogelton and W. Benesova, "Eye blink detection based on motion vectors analysis," *Computer Vision and Image Understanding,* vol. 148, pp. 23-33, 2016.

[51] W. Gan, J. Xue, K. Lu, Y. Yan, P. Gao and J. Lyu, "FEAFA+: an extended well-annotated dataset for facial expression analysis and 3D facial animation," in *Fourteenth International Conference on Digital Image Processing (ICDIP 2022)*, 2022.

[52] A. Al Redhaei, Y. Albadawi, S. Mohamed and A. Alnoman, "Realtime Driver Drowsiness Detection Using Machine Learning," in *2022 Advances in Science and Engineering Technology International Conferences (ASET)*, 2022.

[53] D. S. Ma, J. Correll and B. Wittenbrink, "The Chicago face database: A free stimulus set of faces and norming data," *Behavior research methods,* vol. 47, pp. 1122-1135, 2015.

[54] R. Min, N. Kose and J.-L. Dugelay, “KinectFaceDB: A Kinect Database for Face Recognition,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* 2014.

[55] X.-P. Burgos-Artizzu, P. Perona and P. Dollar, “Robust face landmark estimation under occlusion,” in *International Conference on Computer Vision (ICCV)*, 2013.

[56] F. Adorf, “Detektion von Verdeckungen zur biometrischen Qualitätsbewertung von Gesichtsbildern,” Ruhr-University Bochum, Master Thesis, 2023.

[57] J. Guo, X. Zhu, Y. Yang, F. Yang, Z. Lei and S. Z. Li, “Towards Fast, Accurate and Stable 3D Dense Face Alignment,” in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XIX*, 2020.

[58] C.-Y. Wu, Q. Xu and U. Neumann, “Synergy between 3DMM and 3D Landmarks for Accurate 3D Facial Geometry,” in *International Conference on 3D Vision, 3DV 2021, London, United Kingdom, December 1-3, 2021*, 2021.

[59] N. Gourier, D. Hall and J. L. Crowley, “Estimating Face Orientation from Robust Detection of Salient Facial Features,” in *Cambridge, UK*, 2004, Proceedings of Pointing 2004, ICPR, International Workshop on Visual Observation of Deictic Gestures.

[60] G. G. Slabaugh, “Computing Euler angles from a rotation matrix,” 08-06-1999. [Online]. Available: http://eecs.qmul.ac.uk/~gslabaugh/publications/euler.pdf. [Accessed 07-12-2023].

[61] A. Cobo, R. Valle, J. M. Buenaposada and L. Baumela, “On the representation and methodology for wide and short range head pose estimation,” *Pattern Recognition,* vol. 149, 2024.

[62] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar and I. Matthews, “The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, San Francisco, USA, 2010.

[63] P. Ekman, W. Friesen and J. Hager, “Facial Action Coding System: The Manual,” A Human Face, Salt Lake City, 2002.

[64] T. Karras, S. Laine and T. Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks,” *CORR,* vol. abs/1812.04948, 2018.

[65] Kaggle Inc., “Kaggle - ffhq_emotion_label,” Kaggle Inc., 2022. [Online]. Available: https://www.kaggle.com/datasets/tom99763/ffhq-emotion-label. [Accessed 05-30-2023].

[66] N. Aifanti, C. Papachristou and A. Delopoulos, “The MUG Facial Expression Database,” in *Proc. 11th Int. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, Desenzano, Italy, 2010.

[67] S. M. Mavadati, M. H. Mahoor, K. Bartlett, P. Trinh and J. F. Cohn, “DISFA: A spontaneous facial action intensity database,” *IEEE Transactions on Affective Computing,* vol. 4, no. 2, pp. 151-160, 2013.

[68] Z. Wen, W. Lin, T. Wang and G. Xu, "Distract Your Attention: Multi-head Cross Attention Network for Facial Expression Recognition," *CORR,* vol. abs/2109.07270, 2021.

[69] J. She, Y. Hu, H. Shi, J. Wang, Q. Shen and T. Mei, "DMUE: Dive into Ambiguity: Latent Distribution Mining and Pairwise Uncertainty Estimation for Facial Expression Recognition," in *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[70] M. Grimmer, R. N. J. Veldhuis and C. Busch, “Efficient Expression Neutrality Estimation with Application to Face Recognition Utility Prediction,” *CORR,* vol. abs/2402.05548, 2024.

[71] A. Mollahosseini, B. Hasani and M. H. Mahoor, “AffectNet: A Database for Facial Expression, Valence, and Arousal Computing in the Wild,” *CORR,* vol. abs/1708.03985, 2017.

[72] A. Savchenko, “Facial Expression Recognition with Adaptive Frame Rate based on Multiple Testing Correction,” in *Fortieth International Conference on Machine Learning (ICML)*, Honolulu, Hawaii, USA, 2023.

[73] C. E. Thomaz and G. A. Giraldi, “A new ranking method for principal components analysis and its application to face image analysis,” *Image Vis. Comput.,* vol. 28, p. 902–913, 2010.

[74] P. Hancock, “2D Face Sets,” University of Stirling, 06-04-2012. [Online]. Available: http://pics.stir.ac.uk/2D_face_sets.htm. [Accessed 04-13-2023].

[75] H. Steiner, S. Sporrer, A. Kolb and N. Jung, “Design of an Active Multispectral SWIR Camera System for Skin Detection and Face Verification,” *J. Sensors,* vol. 2016, p. 9682453:1–9682453:16, 2016.

[76] R. B. Randall, “A history of cepstrum analysis and its application to mechanical problems,” *Mechanical Systems and Signal Processing,* vol. 97, pp. 3-19, 2017.

[77] R. Fabian and D. Malah, “Robust identification of motion and out-of-focus blur parameters from blurred and noisy images,” *CVGIP: graphical models and image processing,* vol. 53, no. 5, pp. 403-412, 1991.

[78] S. Gentric, „OFIQ Evaluation - Demographic factors,“ Idemia, Technical Report, 2024.

[79] A. Dörsch, C. Rathgeb, M. Grimmer und C. Busch, „Detection and Mitigation of Bias in Under Exposure Estimation for Face Image Quality Assessment,“ in *23rd International Conference of the Biometrics Special Interest Group (BIOSIG 2024)*, Darmstadt, 2024.