

Multiple-Biometric Evaluation (MBE)
2010

Still Face Image Track
Concept, Evaluation Plan and API
Version 1.0.0

Patrick Grother

Image Group
Information Access Division
Information Technology Laboratory
National Institute of Standards and Technology



February 1, 2010

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

18
19
20
21

Status of this Document

NIST intends that the content of this document is fixed. However, NIST will update the document in response to specific technical issues. NIST may add background information.

Comments and questions should be submitted to mbe2010@nist.gov. A FAQ is maintained as Annex B.

Changes between this and the January 28-th version are shown in blue.

Timeline of the MBE-STILL Evaluation

| | |
|----------------------------|---|
| July to September 2010 | NIST documentation and reports released |
| February 1, 2010, v 1.0.0 | Added enrollment_directory as input to function call in Table 24. |
| January 28, 2010, v 0.9.3 | How-to-submit-implementations information added to Annex A |
| January 27 to May 14, 2010 | Still-face open submission period |
| January 11, 2010 | Clarified evaluation plan, v 0.9.1 |
| December 18, 2009 | Final evaluation plan, v 0.9 |
| December 16, 2009 | Comments period closes on second draft of this document. |
| December 15, 2009 | Release of sample data: http://face.nist.gov/mbe/NIST_SD32v01_MEDS_1_face.zip |
| December 09, 2009 | Second draft evaluation plan (revised version of this document) for public comment. |
| December 04, 2009 | MBGC Workshop, Washington DC |
| November 30, 2009 | Request that participants give non-binding no-commitment indication of whether they will participate in the test. |
| | Comments period closes on first draft of this document. |
| November 16, 2009 | Initial draft evaluation plan (this document) for public comment. |

| October 2009 | November 2009 | December 2009 | January 2010 | February 2010 |
|---|--|--|--|---|
| Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 |
| March 2010 | April 2010 | May 2010 | June 2010 | July 2010 |
| Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |

1 **Table of Contents**

2 1. MBE 6

3 1.1. Scope 6

4 1.2. Audience 6

5 1.3. Market drivers 6

6 1.4. Offline testing 7

7 1.5. Phased testing..... 7

8 1.6. Interim reports..... 7

9 1.7. Final reports..... 7

10 1.8. Application scenarios..... 8

11 1.9. Image source labels 8

12 1.10. Options for participation 9

13 1.11. Use of multiple images per person 9

14 1.12. Provision of photograph date information to the implementation 10

15 1.13. Provision of other metadata to the implementation 10

16 1.14. Core accuracy metrics..... 10

17 1.15. Generalized accuracy metrics 11

18 1.16. Reporting template size..... 11

19 1.17. Reporting computational efficiency 11

20 1.18. Exploring the accuracy-speed trade-space 11

21 1.19. Hardware specification 11

22 1.20. Threaded computations..... 12

23 1.21. Time limits 12

24 1.22. Test datasets..... 13

25 1.23. Compression study 13

26 1.24. Quality analysis 13

27 1.25. Ground truth integrity 14

28 2. Data structures supporting the API 14

29 2.1. Overview 14

30 2.2. Requirement 14

31 2.3. File formats and data structures..... 14

32 2.4. File structures for enrolled template collection 16

33 2.5. Data structure for result of an identification search 17

34 3. API Specification 17

35 3.1. Implementation identifiers 17

36 3.2. Maximum template size 18

37 3.3. 1:1 Verification without enrollment database..... 18

38 3.4. 1:N Identification 21

39 3.5. 1:1 Verification with enrollment database 26

40 3.6. Pose conformance estimation 27

41 3.7. Software and Documentation..... 28

42 3.8. Runtime behavior 30

43 4. References..... 30

44 Annex A Submission of Implementations to the MBE 2010 STILL 32

45 A.1 Submission of implementations to NIST..... 32

46 A.2 How to participate 32

47 A.3 Implementation validation 32

48 Annex B Frequently asked Questions..... 34

49

50 **List of Figures**

51 Figure 1- Organization and documentation of the MBE 6

52 Figure 2- Schematic of verification without enrollment database 19

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

List of Tables

Table 1 – Abbreviations..... 5

Table 2 – Subtests supported under the MBE still-face activity..... 8

Table 3 – MBE classes of participation 9

Table 4 - Summary of accuracy metrics 10

Table 5 – Number of threads allowed for each application 12

Table 6 – Processing time limits in milliseconds 12

Table 7 – Main image corpora (others will be used)..... 13

Table 8 – Labels describing types of images 14

Table 9 – Structure for a single face, with metadata 15

Table 10 – Structure for a set of images from a single person..... 15

Table 11 – Structure for a pair of eye coordinates..... 16

Table 12 - Enrollment dataset template manifest 16

Table 13 – Structure for a candidate..... 17

Table 14 – Implementation identifiers 17

Table 15 - Implementation template size requirements..... 18

Table 16 – Functional summary of the 1:1 application 18

Table 17 – SDK initialization 19

Table 18 – Template generation 19

Table 19 – Template matching 20

Table 20 – Procedural overview of the identification test 21

Table 21 – Enrollment initialization..... 22

Table 22 – Enrollment feature extraction 23

Table 23 – Enrollment finalization..... 24

Table 24 – Identification feature extraction initialization 24

Table 25 – Identification feature extraction..... 25

Table 26 - Identification initialization..... 25

Table 27 – Identification search 26

Table 28 – Verification against an enrolled identity..... 26

Table 29 - Initialization of the pose conformance estimation SDK 27

Table 30 - Pose conformance estimation 27

Table 31 - Implementation library filename convention..... 28

1 **Acknowledgements**

- 2 — The authors are grateful to the experts who made extensive comments on the first (November 16) version of this
 3 document.
 4 — The authors are grateful to the experts who made such rapid and extensive comments on the second (December 09)
 5 version of this document.

6 **Project History**

- 7 — February 1, 2010 - Single API change in Table 24. Fix API at version 1.0
 8 — January 28, 2010 - Addition of Annex A SDK submission information, version 0.9.3
 9 — January 14, 2010 - Minor technical modifications, version 0.9.2
 10 — December 18, 2009 - Release of the third public draft, version 0.9.
 11 — December 09, 2009 - Release of second public draft, version 0.7.
 12 — November 16, 2009 - Release of first public draft of the Multiple-Biometric Evaluation (MBE) 2010 - Still Face Image
 13 Concept, Evaluation Plan and API Version 0.5.
 14 — August 2009 - Brief large scale 1:N proposal to U. S. Government sponsors.

15 **Terms and definitions**

16 The abbreviations and acronyms of Table 1 are used in many parts of this document.

17 **Table 1 – Abbreviations**

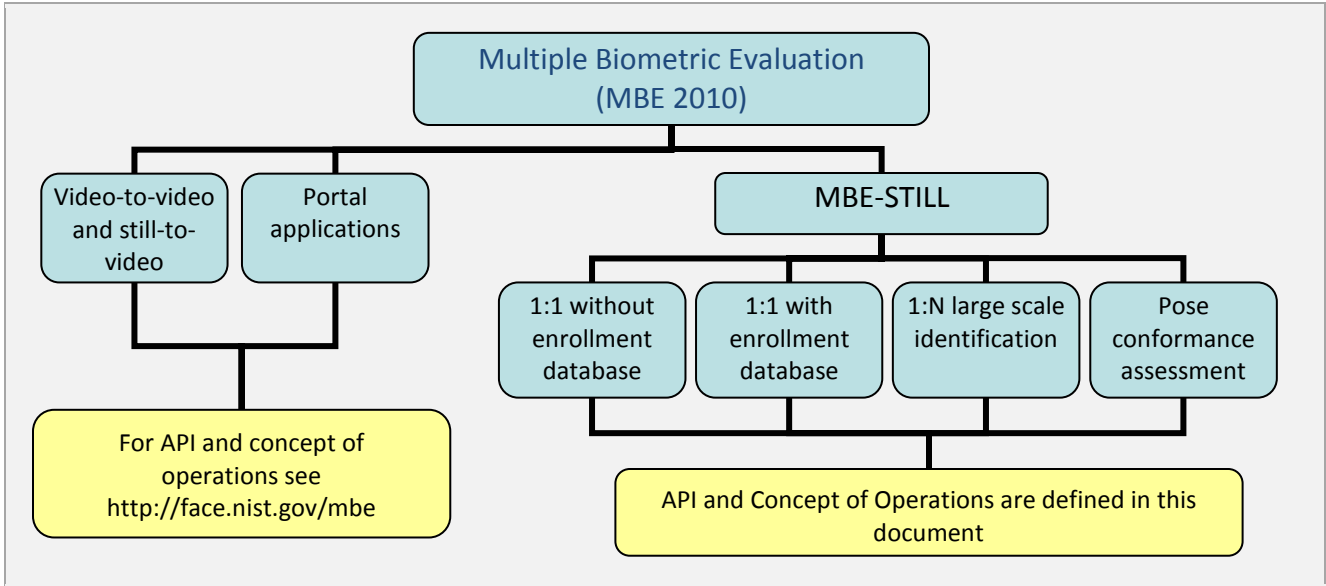
| | |
|-------------------|---|
| FNIR | False negative identification rate |
| FPIR | False positive identification rate |
| FMR | False match rate |
| FNMR | False non-match rate |
| GFAR | Generalized false accept rate |
| GFRR | Generalized false reject rate |
| DET | Detection error tradeoff characteristic: For verification this is a plot of FNMR vs. FMR (sometimes as normal deviates, sometimes on log-scales). For identification this is a plot of FNIR vs. FPIR. |
| INCITS | InterNational Committee on Information Technology Standards |
| ISO/IEC 19794 | ISO/IEC 19794-5: Information technology — Biometric data interchange formats — Part 5:Face image data. First edition: 2005-06-15. (See Bibliography entry). |
| I385 | INCITS 385:2004 - U.S. precursor to the 19794-5 international standard |
| ANSI/NIST Type 10 | The dominant container for facial images in the law enforcement world. |
| MBE | NIST's Multiple Biometric Evaluation program |
| NIST | National Institute of Standards and Technology |
| PIV | Personal Identity Verification |
| SC 37 | Subcommittee 37 of Joint Technical Committee 1 – developer of biometric standards |
| SDK | The term Software Development Kit refers to any library software submitted to NIST. This is used synonymously with the terms "implementation" and "implementation under test". |

18

1 **1. MBE**

2 **1.1. Scope**

3 This document establishes a concept of operations and an application programming interface (API) for evaluation of face
 4 recognition implementations submitted to NIST's Multiple Biometric Evaluation. This document covers only the
 5 recognition of two-dimensional still-images. As depicted in Figure 1, the recognition-from-video and face-iris portal tracks
 6 of the MBE program are documented elsewhere. See <http://face.nist.gov/mbe> for all MBE documentation.



7 **Figure 1- Organization and documentation of the MBE**

8 **1.2. Audience**

9 Universities and commercial entities with capabilities in following areas are invited to participate in the MBE still-face test.

- 10 – Identity verification with face recognition algorithms
- 11 – Large scale identification implementations.
- 12 – Organizations with a capability to assess pose orientation of a face in an image.

13 Organizations will need to implement the API defined in this document. Participation is open worldwide. There is no
 14 charge for participation. While NIST intends to evaluate technologies that could be readily made operational, the test is
 15 also open to experimental, prototype and other technologies.

16 **1.3. Market drivers**

17 This test is intended to support a plural marketplace of face recognition systems. While the dominant application, in
 18 terms of revenue, has been one-to-many search for driving licenses and visa issuance, the deployment of one-to-one face
 19 recognition has re-emerged with the advent of the e-Passport verification projects¹. In addition, there remains
 20 considerable activity in the use of FR for surveillance applications.

21 These applications are differentiated by the population size (and other variables). In the driving license duplicate
 22 detection application, the enrollment database might exceed 10⁷ people. In the surveillance application, the watchlist
 23 size can readily extend to 10⁴.

¹ These match images acquired from a person crossing a border against the ISO/IEC 19794-5 facial image stored on the embedded ISO/IEC 7816 + ISO/IEC ISO 14443 chips.

1 **1.4. Offline testing**

2 While this set of tests is intended as much as possible to mimic operational reality, this remains an offline test executed
3 on databases of images. The intent is to assess the core algorithmic capability of face recognition algorithms. This test will
4 be conducted purely offline - it does not include a live human-presents-to-camera component. Offline testing is attractive
5 because it allows uniform, fair, repeatable, and efficient evaluation of the underlying technologies. Testing of
6 implementations under a fixed API allows for a detailed set of performance related parameters to be measured.

7 **1.5. Phased testing**

8 To support research and development efforts, this testing activity will embed multiple rounds of testing. These test
9 rounds are intended to support improved performance. Once the test commences, NIST will test implementations on a
10 first-come-first-served basis and will return results to providers as expeditiously as possible. Providers may submit
11 revised SDKs to NIST only after NIST provides results for the prior SDK. The frequency with which a provider may submit
12 SDKs to NIST will depend on the times needed for vendor preparation, transmission to NIST, validation, execution and
13 scoring at NIST, and vendor review and decision processes.

14 For the number of SDKs that may be submitted to NIST see section 1.10.

15 **1.6. Interim reports**

16 The performance of each SDK will be reported in a "score-card". This will be provided to the participant. While the score
17 cards may be used by the provider for arbitrary purposes, they are intended to allow development. The score cards will

- 18 – be machine generated (i.e. scripted),
- 19 – be provided to participants with identification of their implementation,
- 20 – include timing, accuracy and other performance results,
- 21 – include results from other implementations, but will not identify the other providers,
- 22 – be expanded and modified as revised implementations are tested, and as analyses are implemented,
- 23 – be generated and released asynchronously with SDK submissions,
- 24 – be produced independently of the other status of other providers' implementations,
- 25 – be regenerated on-the-fly, primarily whenever any implementation completes testing, or when new analysis is
26 added.

27 NIST does not intend to release these test reports publicly. NIST may release such information to the U.S. Government
28 test sponsors. While these reports are not intended to be made public, NIST can only request that agencies not release
29 this content.

30 **1.7. Final reports**

31 At some point NIST will terminate the testing rounds and will write one or more final public reports. NIST may publish

- 32 – Reports (typically as numbered NIST Interagency Reports),
- 33 – Publications in the academic literature,
- 34 – Presentations (typically PowerPoint).

35 Our intention is that the final test reports will publish results for the best-performing implementation from each
36 participant. Because "best" is ill-defined (accuracy vs. time vs. template size, for example), the published reports may
37 include results for other implementations. The intention is to report results for the most capable implementations (see
38 section 1.14, on metrics). Other results may be included (e.g. in appendices) to show, for example, examples of progress
39 or tradeoffs. **IMPORTANT: Results will be attributed to the providers.**

1 **1.8. Application scenarios**

2 The test will include one-to-one verification tests, and one-to-many identification tests². As described in Table 2, the test
3 is intended to represent:

- 4 – Close-to-operational use of face recognition technologies in identification applications in which the enrolled dataset
5 could contain images from up to three million persons.
- 6 – Verification scenarios in which still images are compared.
- 7 – Verification scenarios in which images are compared with entries in an enrolled database.

8 **Table 2 – Subtests supported under the MBE still-face activity**

| # | | A | B | C | D |
|----|--|---|---|---|--|
| 1. | Aspect | 1:1 verification | 1:1 verification | 1:N identification | Pose estimation |
| 2. | Enrollment dataset | None, application to single images. | N enrolled subjects | N enrolled subjects | None, application to single images, |
| 3. | Prior NIST test references | Equivalent to 1 to 1 matching in [FRVT 2006] | Equivalent to gallery normalization in [FRVT 2006] | | |
| 4. | Example application | Verification of e-Passport facial image against a live border-crossing image. | Verification of live capture against a central access control database after presentation of an ID credential | Open-set identification of an image against a central database, e.g. a search of a mugshot against a database of known criminals. | During capture, algorithm assesses whether face is frontal or not, or estimates pose. Frontal pose is required in formal standards because non-frontal pose eventually degrades face recognition accuracy. |
| 5. | Score or feature space normalization support | Vendor uses normalization techniques over SDK-internal datasets | Vendor applies normalization techniques against enrollment dataset and internal datasets | Any score or feature based statistical normalization techniques-are applied against enrollment database | |
| 6. | Intended number of subjects | Up to $O(10^5)$ | Up to $O(10^5)$ | Up to $O(10^7)$ but dependence on N will be computed. From $O(10^2)$ upwards. | Expected $O(10^3)$ |
| 7. | Number of images per individual | Variable, see section 1.11. | Variable, see section 1.11. | Variable, see section 1.11. | 1 |
| 8. | Metadata items | Sex, date of image, date of birth, race, height, weight. These may not be available to the SDK. NIST may run tests with and without this data. IMPORTANT: Metadata may be missing for some images. Metadata may be unreliable. | | | |

9
10 NOTE 1: The vast majority of images are color. The API supports both color and greyscale images.

11 NOTE 2: For the operational datasets, it is not known what processing was applied to the images before they were
12 archived. So, for example, we do not know whether gamma correction was applied. NIST considers that best practice,
13 standards and operational activity in the area of image preparation remains weak.

14 **1.9. Image source labels**

15 NIST may mix images from different source in an enrollment set. For example, NIST could combine N/2 mugshot images
16 and N/2 visa images into a single enrollment dataset. For this reason, in the data structure defined in clause 2.3.3, each
17 image is accompanied by a "label" which identifies the set-membership images. The legal values for labels are given in
18 clause 2.3.2.

² NIST has previously only modeled identification scenarios. The simplest simulation mimics a 1:N search by conducting N 1:1 comparisons.

1 **1.10. Options for participation**

2 The following rules apply:

- 3 — A participant must properly follow, complete and submit the Annex A Participation Agreement. This must be done
- 4 once. It is not necessary to do this for each submitted SDK.
- 5 — All participants shall submit at least one class A SDK. This shall be sent before, or concurrently with, any class B or C
- 6 SDKs.
- 7 — Class B, C and D SDKs are optional.
- 8 — Any SDK shall implement exactly one of the functionalities defined in clause 3. So, for example, the 1:1 functionality
- 9 of a class A SDK shall not be merged with that of a class C SDK.
- 10 — At any point in time, the maximum number of SDKs undergoing testing at NIST will be two. This is the total of class A
- 11 plus B plus C and D. NIST will invite submission of revised SDKs when testing of each prior SDK has been completed.
- 12 — A provider of an SDK may ask NIST not to repeat feature extraction and enrollment processes. This may expedite
- 13 testing of an SDK because NIST can proceed directly to identification and verification trials. NIST cannot conduct
- 14 surveys over runtime parameters - NIST must limit the extent to which participants are able to train on the test data.

16 **Table 3 – MBE classes of participation**

| Function | 1:1 verification without enrollment database | 1:1 verification with enrollment database | 1:N identification | Pose conformance estimation |
|------------------|--|---|--------------------|-----------------------------|
| Class label | A | B | C | D |
| API requirements | 3.1 + 3.2 + 3.3 | 3.1 + 3.2 + 3.5 + 3.4.2 + 3.4.3 + 3.4.4 | 3.1 + 3.2 + 3.4 | 3.1 + 3.6 |

17

18 Class A might be preferred by academic institutions because the API supports the elemental hypothesis testing

19 verification function "are the images from the same person or not?"

20 **1.11. Use of multiple images per person**

21 Some of the proposed datasets includes $K > 2$ images per person for some persons. This affords the possibility to model a

22 recognition scenario in which a new image of a person is compared against all prior images³. Use of multiple images per

23 person has been shown to elevate accuracy over a single image [FRVT2002b].

24 For this test, NIST will enroll $K \geq 1$ images under each identity. Normally the probe will consist of a single image, but NIST

25 may examine the case that it could consist of multiple images. Ordinarily, the probe images will be captured after the

26 enrolled images of a person⁴. The method by which the face recognition implementation exploits multiple images is not

27 regulated: The test seeks to evaluate vendor provided technology for multi-instance fusion. This departs from some prior

28 NIST tests in which NIST executed fusion algorithms ([e.g. [FRVT2002b], and sum score fusion, for example, [MINEX]).

29 This document defines a template to be the result of applying feature extraction to a set of $K \geq 1$ images. That is, a

30 template contains the features extracted from one or more images, not generally just one. An SDK might internally fuse K

31 feature sets into a single representation or maintain them separately - In any case the resulting template is a single

32 proprietary block of data. All verification and identification functions operate on such multi-image templates.

33 The number of images per person will depend on the application area:

³ For example, if a banned driver applies for a driving license under a new name, and the local driving license authority maintains a driving license system in which all previous driving license photographs are enrolled, then the fraudulent application might be detected if the new image matched any of the prior images. This example implies one (elemental) method of using the image history.

⁴ To mimic operational reality, NIST intends to maintain a causal relationship between probe and enrolled images. This means that the enrolled images of a person will be acquired before all the images that comprise a probe.

- 1 — In civil identity credentialing (e.g. passports, driving licenses) the images will be acquired approximately uniformly
- 2 over time (e.g. five years for a Canadian passport). While the distribution of dates for such images of a person might
- 3 be assumed uniform, a number of factors might undermine this assumption⁵.
- 4 — In criminal applications the number of images would depend on the number of arrests⁶. The distribution of dates for
- 5 arrest records for a person (i.e. the recidivism distribution) has been modeled using the exponential distribution, but
- 6 is recognized to be more complicated. NIST currently estimates that the number of images will never exceed 100.
- 7 NIST will not use this API for video data.

8 **1.12. Provision of photograph date information to the implementation**

9 Due to face ageing effects, the utility of any particular enrollment image is dependent on the time elapsed between it and

10 the probe image. In MBE, NIST intends to use the most recent image as the probe image, and to use the remaining prior

11 images under a single enrolled identity.

12 **1.13. Provision of other metadata to the implementation**

13 For any given image, NIST may provide biographical metadata to the SDK. The list of variables is given in the Table 9 face

14 image data-structure. Note that such data is often collected operationally and may be unreliable. NIST will attempt to

15 quantify the utility of providing metadata by running facial recognition accuracy tests with and without the metadata.

16 **1.14. Core accuracy metrics**

17 Notionally the error rates for verification applications will be false match and false non-match error rates, FMR and FNMR.

18 For identification testing, the test will target open-universe applications such as benefits-fraud and watch-lists. It will not

19 address the closed-set task because it is operationally uncommon.

20 While some one-to-many applications operate with purely rank-based metrics, this test will primarily target score-based

21 identification metrics. Metrics are defined in Table 4. The analysis will survey over various rank and thresholds. Plots of

22 the two error rates, parametric on threshold, will be the primary reporting mechanism.

23 **Table 4 - Summary of accuracy metrics**

| | Application | Metric |
|---|--|---|
| A | 1:1 Verification | FMR = Fraction of impostor comparisons that produce a similarity score greater than a threshold value |
| | | FNMR = Fraction of genuine comparisons that produce a similarity score less than some threshold value |
| B | 1:N Identification Primary identification metric. | FPIR = Fraction of searches that do not have an enrolled mate for which one or more candidate list entries exceed a threshold |
| | | FNIR = Fraction of searches that have an enrolled mate for which the mate is below a threshold |
| C | 1:N Identification (with rank criteria) Secondary identification metric | FPIR = Fraction of searches that do not have an enrolled mate for which one or more candidate list entries exceed a threshold |
| | | FNIR = Fraction of searches that have an enrolled mate for which the mate is not in the best R ranks <i>and</i> at or above a threshold |

24

25 NOTE: The metric on line B is a special case of the metric on line C: the rank condition is relaxed ($R \rightarrow N$). Metric B is the

26 primary metric of interest because the target application does not include a rank criterion.

27 FPIR and FMR will usually be estimated using probe images for which there is no enrolled mate.

⁵ For example, a person might skip applying for a passport for one cycle (letting it expire). In addition, a person might submit identical images (from the same photography session) to consecutive passport applications at five year intervals.

⁶ A number of distributions have been considered to model recidivism, see "Random parameter stochastic process models of criminal careers." In Blumstein, Cohen, Roth & Visher (Eds.), *Criminal Careers and Career Criminals*, Washington, D.C.: National Academy of Sciences Press, 1986.

1 NIST will extend the analysis in other areas, with other metrics, and in response to the experimental data and results.

2 **1.15. Generalized accuracy metrics**

3 Under the ISO/IEC 19795-1 biometric testing and reporting standard, a test must account for "failure to acquire" and
4 "failure to enroll" events (e.g. elective refusal to make a template, or fatal errors). The effect of these is application-
5 dependent.

6 For verification, the appropriate metrics reported in MBE 2010 will be generalized error rates (GFAR, GFRR). When single
7 images are compared, (GFAR,GFRR) and (FMR,FNMR) will be equivalent if no failures are observed.

8 Similarly for identification, generalized error rates will be reported.

9 **1.16. Reporting template size**

10 Because template size is influential on storage requirements and computational efficiency, this API supports
11 measurement of template size. NIST will report statistics on the actual sizes of templates produced by face recognition
12 implementations submitted to MBE. NIST may report statistics on runtime memory usage. Template sizes were reported
13 in the NIST Iris Exchange test⁷.

14 **1.17. Reporting computational efficiency**

15 As with other tests, NIST will compute and report recognition accuracy. In addition, NIST will also report timing statistics
16 for all core functions of the submitted SDK implementations. This includes feature extraction, and 1:1 and 1:N
17 recognition. For an example of how efficiency can be reported, see the final report of the NIST Iris Exchange test⁷.

18 Note that face recognition applications optimized for pipelined 1:N searches may not demonstrate their efficiency in pure
19 1:1 comparison applications.

20 **1.18. Exploring the accuracy-speed trade-space**

21 Organizations may enter two SDKs per class. This is intended to allow an exploration of accuracy vs. speed tradeoffs for
22 face recognition algorithms running on a fixed platform. NIST will report both accuracy and speed of the implementations
23 tested. While NIST cannot force submission of "fast vs. slow" variants, participants may choose to submit variants on
24 some other axis (e.g. "experimental vs. mature") implementations. NIST encourages "fast-less-accurate vs. slow-more-
25 accurate" with a factor of three between the speed of the fast and slow versions.

26 **1.19. Hardware specification**

27 NIST intends to support high performance by specifying the runtime hardware beforehand. NIST will execute the test on
28 high-end PC-class computers. These machines have 4-cpus, each of which has 4 cores. These blades are labeled Dell
29 M905 equipped with 4x Quad Core AMD Opteron 8376HE processors⁸ running at 2.3GHz. Each CPU has 512K cache. The
30 bus runs at 667 Mhz. The main memory is 192 GB Memory as 24X8GB modules. We anticipate that 16 processes can be
31 run without time slicing.

32 All submitted implementations shall run on either

- 33 — Red Hat Enterprise Linux Server release 5.1 (Tikanga) (late linux 2.6 kernels), or
- 34 — Windows Server 2003 Enterprise R2 x64

35 The Linux option is preferred by NIST, but providers should choose whichever platform suits them. Providers are
36 cautioned that their choice of operating system may have some impact on efficiency. NIST will provide appropriate
37 caveats to the test report. NIST will respond to prospective participants' questions on the hardware, by amending this
38 section.

⁷ See the IREX (Iris Exchange) test report: NIST Interagency Report 7629, linked from <http://iris.nist.gov/irex>

⁸ `cat /proc/cpuinfo` returns `fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rdtscp lm 3wext 3dnow constant_tsc nonstop_tsc pni cx16 popcnt lahf_lm cmp_legacy svm extapic cr8_legacy altmovcr8 abm sse4a misalignsse 3dnowprefetch osvw`

1 NIST is recommending use of 64 bit implementations throughout. This will support large memory allocation - this seems
 2 necessary for the 1:N identification task with image counts in the millions. NIST will allow 32 bit operation for 1:1.
 3 If all templates were to be held in memory, the 192GB capacity implies a limit of 20KB per template, for a 10 million
 4 image enrollment. The API allows read access of the disk during the 1:N search.
 5 Some of the section 3 API calls allow the SDK to write persistent data to hard disk. The amount of data shall not exceed
 6 200 kilobytes per enrolled image.

7 **1.20. Threaded computations**

8 Table 5 shows the limits on the numbers of threads an face recognition implementation may use. In many cases
 9 threading is not permitted (i.e. T=1) because NIST will parallelize the test by dividing the workload across many cores and
 10 many machines. For the functions where we allow multi-threading, e.g. in the 1:N test, NIST requires the provider to
 11 disclose the maximum number of threads to us. If that number is T, NIST will run the largest integer number of processes,
 12 P, in parallel such that TP ≤ 16.

13 **Table 5 – Number of threads allowed for each application**

| | 1 | 2 | 3 | 4 |
|---|--|---|--------------------|-----------------------------|
| Function | 1:1 verification without enrollment database | 1:1 verification with enrollment database | 1:N identification | Pose conformance estimation |
| Feature extraction | 1 | 1 | 1 | 1 |
| Verification | 1 | 1 | NA | |
| Finalize enrollment (before 1:1 or 1:N) | NA | 1 ≤ T ≤ 16 | 1 ≤ T ≤ 16 | |
| Identification | NA | NA | 1 ≤ T ≤ 16 | |

14
 15 NIST will not run an SDK from participant X and an SDK from participant Y on the same machine at the same time.
 16 For single-threaded libraries, NIST will run up to 16 processes concurrently. NIST's calling applications are single-threaded.

17 **1.21. Time limits**

18 The elemental functions of the implementations shall execute under the time constraints of Table 6. These times limits
 19 apply to the function call invocations defined in section 3. Assuming the times are random variables, NIST cannot regulate
 20 the maximum value, so the time limits are 90-th percentiles. This means that 90% of all operations should take less than
 21 the identified duration.

22 The time limits apply per image. When K images of a person are present, the time limits shall be increased by a factor K.

23 **Table 6 – Processing time limits in milliseconds**

| | 1 | 2 | 3 | 4 | 5 |
|---|---|--|---|-------------------------------------|-----------------------------|
| Function | | 1:1 verification without enrollment database | 1:1 verification with enrollment database | 1:N identification | Pose conformance estimation |
| Feature extraction enrollment | | 1000 (1 core) | 1000 (1 core) | 1000 (1 core) | 500 (1 core) |
| Feature extraction for verification or identification | | 1000 (1 core) | 1000 (1 core) | 1000 (1 core) | |
| Verification | | 5 (1 core) | 10 (1 core) | NA | |
| Identification of one search image against 1,000,000 single-image MULTIFACE records. | | NA | NA | 10000 (16 cores) or 160000 (1 core) | |

24
 25 In addition the enrollment finalization procedure is subject to a time limit, as follows. For an enrollment of one million
 26 single-image **MULTIFACES**, the total time shall be less than 7200 seconds. The implementation can use up to 16 cores. This
 27 limit includes disk IO time.

1 **1.22. Test datasets**

2 This section is under development. The data has, in some cases, been estimated from initial small partitions. The
 3 completion of this section depends on further work. The information is subject to change. We intend to update this
 4 section as fully as possible.

5 NIST is likely to use other datasets, in addition.

6 **Table 7 – Main image corpora (others will be used)**

| | Laboratory | FRVT 2002+2006 / HCINT | Multiple Encounter Database |
|-------------------------|--|---|--|
| Collection, environment | See the Error! Reference source not found. for details on these images. See also FRVT 2006 Report, Phillips et al. NIST IR 7408. | Visa application process | Law enforcement booking |
| Live scan, Paper | | Live | Live, few paper |
| Documentation | | See NIST IR 6965 [FRVT2002] | See NIST Special Database 32 Volume 1, available 12/09 ⁹ . |
| Compression | | JPEG mean size 9467 bytes. See [FRVT2002b] | JPEG ~ 20:1 |
| Maximum image size | | 300 x 252 | Mixed, some are 640x480 others are 768x960, some are smaller. |
| Minimum image size | | 300 x 252 | |
| Eye to eye distance | | Median = 71 pixels | mean=156, sd=46 |
| Frontal | | Yes, well controlled | Moderately well controlled Profile images will be included and labeled as such. |
| Full frontal geometry | | Yes, in most cases. Faces may have small background than ISO FF requires. | Mostly not. Varying amounts of the torso are visible. |
| Intended use | 1:1 | 1:1 and 1:N | 1:N |

7 **1.23. Compression study**

8 The effect of compression on accuracy will be studied by applying the JPEG and JPEG 2000 compression algorithms to
 9 uncompressed image corpora maintained at NIST. The studies will be conducted on images conforming to the full-frontal
 10 or token frontal geometries of the ISO/IEC 19794-5 standard. Towards this end, images will be compressed and resized
 11 till the algorithms break.

12 The results might refine the guidance given in ISO/IEC 19794-5:2005 Face Image Interchange standard, Annex A.

13 This study will be conducted in verification trials (i.e. class A SDKs).

14 **1.24. Quality analysis**

15 NIST will examine the effectiveness of quality scores in predicting recognition accuracy. A quality score is computed from
 16 an input record during feature extraction. The default method of analysis will be the error vs. reject analysis document in
 17 P. Grother and E. Tabassi, *Performance of biometric quality measures*, IEEE Trans. PAMI, 29:531–543, 2007.

18 The default use-case is that the enrollment image is assumed to be pristine (in conformance with the ISO standard, for
 19 example), and quality is being used *during* a verification or identification transaction to select the image most likely to
 20 match the reference image. The reference image is assumed to be unavailable for matching during the collection.

21 For reasons of operational realism, metadata, such as a date of birth, will not normally be provided to the quality
 22 computation.

23 Analyses other than for the default case may be conducted.

⁹ NIST Special Database 32, Volume 1, is available at: http://face.nist.gov/mbe/NIST_SD32v01_MEDS_I_face.zip. This link is temporary. The database will ultimately be linked from <http://face.nist.gov/mbe>.

1 **1.25. Ground truth integrity**

- 2 Some of the test databases will be derived from operational systems. They may contain ground truth errors in which
- 3 — a single person is present under two different identifiers, or
 - 4 — two persons are present under one identifier, or
 - 5 — in which a face is not present in the image.

6 If these errors are detected, they will be removed. NIST will use aberrant scores (high impostor scores, low genuine scores) to detect such errors. This process will be imperfect, and residual errors are likely. For comparative testing, identical datasets will be used and the presence of errors should give an additive increment to all error rates. For very accurate implementations this will dominate the error rate. NIST intends to attach appropriate caveats to the accuracy results. For prediction of operational performance, the presence of errors gives incorrect estimates of performance.

11 **2. Data structures supporting the API**

12 **2.1. Overview**

13 This section describes separate APIs for the core face recognition applications described in section 1.8. All SDK's submitted to MBE shall implement the functions required by the rules for participation listed before Table 3.

15 **2.2. Requirement**

16 MBE participants shall submit an SDK which implements the relevant "C" prototyped interfaces of clause 3.

17 **2.3. File formats and data structures**

18 **2.3.1. Overview**

19 In this face recognition test, an individual is represented by $K \geq 1$ two-dimensional facial images, and by subject and image-specific metadata.

21 **2.3.2. Dictionary of terms describing images**

22 Images will be accompanied by one of the labels given in Table 8. Face recognition implementations submitted to MBE should tolerate images of any category.

24 **Table 8 – Labels describing types of images**

| | Label as "C" char * string | Primary test area | Meaning |
|----|--------------------------------------|-------------------|--|
| 1. | "unknown" | | Either the label is unknown or unassigned. |
| 2. | "laboratory frontal controlled" | 1:1 | Frontal with controlled illumination |
| 3. | "laboratory frontal uncontrolled" | 1:1 | Any illumination |
| 4. | "laboratory nonfrontal controlled" | 1:1 | NOTE: There is no hyphen "-" |
| 5. | "laboratory nonfrontal uncontrolled" | 1:1 | Any illumination, pose is unknown and could be frontal |
| 6. | "visa" | 1:N | Either a member of the FRVT 2002/2006 HCINT corpus or one of similar properties. |
| 7. | "mugshot" | 1:N | Either a member of the Multi-encounter law enforcement database or one of similar properties. The image is nominally frontal - See NIST Special Database 32. |
| 8. | "profile" | 1:N | The image is a profile image taken from the multi-encounter law enforcement database. |

25
26 As of December 2009, the claims that "profile" images are useful for facial recognition are very few. Profile images are
27 not likely to be used in MBE 2010. The option is being maintained in this API to support future tests and demonstrations.

1 **2.3.3. Data structures for encapsulating multiple images**

2 The standardized formats for facial images are the ISO/IEC 19794-5:2005 and the ANSI/NIST ITL 1-2007 type 10 record.
 3 The ISO record can store multiple images of an individual in a standalone binary file. In the ANSI/NIST realm, K images of
 4 an individual are usually represented as the concatenation of one Type 1 record + K Type 10 records. The result is usually
 5 stored as an EFT file.

6 For the current test, neither ANSI/NIST Type 10 nor ISO/IEC 19794-5 is used because they do not encode metadata
 7 information such as capture date and sex¹⁰.

8 An alternative method of representing K images of an individual is to define a structure containing an image filename and
 9 metadata fields. Each file contains a standardized image format, e.g. PNG (lossless) or JPEG (lossy).

10 **Table 9 – Structure for a single face, with metadata**

| | "C" code fragment | Remarks |
|-----|------------------------|---|
| 1. | typedef struct sface | |
| 2. | { | |
| 3. | uint16_t image_width; | Number of pixels horizontally |
| 4. | uint16_t image_height; | Number of pixels vertically |
| 5. | uint16_t image_depth; | Number of bits per pixel. Legal values are 8 and 24. |
| 6. | uint8_t format; | Flag indicating native format of the image 0x01 = JPEG (i.e. compressed data) 0x02 = PNG (i.e. never compressed data) |
| 7. | uint8_t *data; | Pointer to raster scanned data. Either RGB color or intensity. If image_depth == 24 this points to 3WH bytes RGBRGBRGB... If image_depth == 8 this points to WH bytes IIIIIII |
| 8. | char *description; | Single description of the image. The allowed values for this string are given in Table 8. |
| 9. | | |
| 10. | uint16_t dob; | Day of birth on [1,31]; 0 indicates unknown |
| 11. | uint16_t mob; | Month of birth [1-12]; 0 indicates unknown |
| 12. | uint16_t yob; | Year of birth [0- 2099]; 0 indicates unknown |
| 13. | uint16_t day; | Day of image capture [1-31]; 0 indicates unknown |
| 14. | uint16_t month; | Month of image capture [1-12]; 0 indicates unknown |
| 15. | uint16_t year; | Year of image capture [0-2099]; 0 indicates unknown |
| 16. | uint8_t sex; | This field uses the ISO/IEC 19794-5 values: Unspecified = 0x00; Male = 0x01; Female = 0x02; Unknown 0xFF |
| 17. | uint8_t race; | This field will use these values: 0x00 - Unassigned or unknown 0x01 -- American Indian or Alaska Native 0x02 -- Asian 0x03 -- Black or African American 0x04 -- Hispanic or Latino 0x05 -- Native Hawaiian or Other Pacific Islander 0x06 -- White |
| 18. | uint16_t weight; | Body weight in kilograms: 0x00 - unassigned or unknown |
| 19. | uint16_t height; | Height in centimeters: 0x00 - unassigned or unknown |
| 20. | } ONEFACE; | |

11 **Table 10 – Structure for a set of images from a single person**

| | "C" code fragment | Remarks |
|----|------------------------|---|
| 1. | typedef struct mface | |
| 2. | { | |
| 3. | unsigned int numfaces; | The number of accessible files, F, such that the last element is faces[F-1] |

¹⁰ In ANSI/NIST such content is routinely transmitted in Type 2, but it's not uniformly standardized, and in case overly complicated for the purpose of testing.

| | | |
|----|------------------|---|
| 4. | ONEFACE **faces; | Pointers to F pre-allocated face images of the same person. |
| 5. | } MULTIFACE; | |

1
2
3
4
5
6
7
8
9

2.3.4. Data structure for eye coordinates

SDKs should return eye coordinates of each enrolled facial image. This function, while not necessary for a recognition test, will assist NIST in assuring the correctness of the test database. The primary mode of use will be for NIST to inspect images for which eye coordinates are not returned, or differ between vendor SDKs.

The eye coordinates shall follow the placement semantics of the ISO/IEC 19794-5:2005 standard - the geometric midpoints of the endocanthion and exocanthion (see clause 5.6.4 of the ISO standard).

Sense: The label "left" refers to subject's left eye (and similarly for the right eye), such that xright < xleft.

Table 11 – Structure for a pair of eye coordinates

| | "C" code fragment | Remarks |
|----|---------------------|---|
| 1. | typedef struct ohos | |
| 2. | { | |
| | uint8_t failed; | If the eye coordinates have been computed and assigned, this value should be set to 0 otherwise it should be set on [1,255]. |
| 3. | int16_t xleft; | X and Y coordinate of the center of the subject's left eye. Out-of-range values (e.g. x < 0 or x >= width) indicate the implementation believes the eye center is outside the image. |
| 4. | int16_t yleft; | |
| 5. | int16_t xright; | X and Y coordinate of the center of the subject's right eye. Out-of-range values (e.g. x < 0 or x >= width) indicate the implementation believes the eye center is outside the image. |
| 6. | int16_t yright; | |
| 7. | } EYEPAIR; | |

10

2.3.5. Data type for similarity scores

Identification and verification functions shall return a measure of the similarity between the face data contained in the two templates. The datatype shall be an eight byte double precision real. The legal range is [0, DBL_MAX], where the DBL_MAX constant is larger than practically needed and defined in the <limits.h> include file. Larger values indicate more likelihood that the two samples are from the same person.

Providers are cautioned that algorithms that natively produce few unique values (e.g. integers on [0,127]) will be disadvantaged by the inability to set a threshold precisely, as might be required to attain a false match rate of exactly 0.0001, for example.

2.4. File structures for enrolled template collection

An SDK converts a MULTIFACE into a template, using, for example the "convert_multiface_to_enrollment_template" function of section 3.4.3. To support the class B verification and class C identification functions of Table 3, NIST will concatenate enrollment templates into a single large file. This file is called the EDB (for enrollment database). The EDB is a simple binary concatenation of proprietary templates. There is no header. There are no delimiters. The EDB may extend to hundreds of gigabytes in length

This file will be accompanied by a manifest; this is an ASCII text file documenting the contents of the EDB. The manifest has the format shown as an example in Table 12. If the EDB contains N templates, the manifest will contain N lines. The fields are space (ASCII decimal 32) delimited. There are three fields, all containing numeric integers. Strictly speaking, the third column is redundant.

25
26
27
28
29

Table 12 - Enrollment dataset template manifest

| Field name | Template ID | Template Length | Position of first byte in EDB |
|---|--------------------------|--------------------------|-------------------------------|
| Datatype required | Unsigned decimal integer | Unsigned decimal integer | Unsigned decimal integer |
| Datatype length required | 4 bytes | 4 bytes | 8 bytes |
| Example lines of a manifest file appear to the right. Lines 1, 2, 3 and N appear. | 90201744 | 1024 | 0 |
| | 163232021 | 1536 | 1024 |
| | 7456433 | 512 | 2560 |

| | |
|--|---|
| | ... |
| | 183838 1024 307200000 |

1
2 This modification of the API (since version 0.5) avoids file system overhead associated with storing millions of individual
3 files.

4 **2.5. Data structure for result of an identification search**

5 All identification searches shall return a candidate list of a NIST-specified length. The list shall be sorted with the most
6 similar matching entries list first with lowest rank. The data structure shall be that of Table 13.

7 **Table 13 – Structure for a candidate**

| | "C" code fragment | Remarks |
|----|---|---|
| 1. | <code>typedef struct candidate</code> | |
| 2. | <code>{</code> | |
| 3. | <code> uint8_t failed;</code> | If the candidate computation failed, this value is set on [1,255]. If the candidate is valid it should be set to 0. |
| 4. | <code> uint32_t template_id;</code> | The Template ID integer from the enrollment database manifest defined in clause 2.4. |
| 5. | <code> double similarity_score;</code> | Measure of similarity between the identification template and the enrolled candidate. Higher scores mean more likelihood that the samples are of the same person. An algorithm is free to assign any value to a candidate. The distribution of values will have an impact on the appearance of a plot of false-negative and false-positive identification rates. |
| 6. | <code> double probability;</code> | An estimate of the probability that the biometric data and candidate belong to <i>different</i> persons, i.e. the probability that a score this large would be observed given that the pair of images are from different people = P(SCORE IMPOSTOR). This value shall be on [0:1]. This is one minus the integral of the expected impostor distribution from 0 to the similarity score, i.e. the expected false match rate. |
| 7. | <code>} CANDIDATE;</code> | |

8

9 **3. API Specification**

10 **3.1. Implementation identifiers**

11 All implementations shall support the self-identification function of Table 14. This function is required to support internal
12 NIST book-keeping. The version numbers should be distinct between any versions which offer different algorithmic
13 functionality.

14 **Table 14 – Implementation identifiers**

| | | |
|-------------------|--|---|
| Prototype | <code>int32_t get_pid(char *sdk_identifier, char *email_address);</code> | A vendor-assigned ID. This shall be different for each submitted SDK. |
| | | Output |
| Description | This function retrieves a point-of-contact email address from the implementation under test. | |
| Output Parameters | <code>sdk_identifier</code> | Version ID code as hexadecimal integer printed to null terminated ASCII string. NIST will allocate exactly 5 bytes for this. This will be used to identify the SDK in the results reports. This value should be changed every time an SDK is submitted to NIST. The value is vendor assigned - format is not regulated by NIST. EXAMPLE: "011A" |
| | <code>email_address</code> | Point of contact email address as null terminated ASCII string. NIST will allocate at least 64 bytes for this. SDK shall not allocate. |
| Return Value | 0 | Success |
| | Other | Vendor-defined failure |

1 **3.2. Maximum template size**

2 All implementations shall report the maximum expected template sizes. These values will be used by the NIST test
 3 harnesses to pre-allocate template data. The values should apply to a single image. For a **MULTIFACE** containing K images,
 4 NIST will allocate K times the value returned. The function call is given in Table 15.

5 **Table 15 - Implementation template size requirements**

| | | | |
|-------------------|--|---|--------|
| Prototype | int32_t get_max_template_sizes(uint32_t *max_enrollment_template_size, uint32_t *max_recognition_template_size) | | Output |
| | | | Output |
| Description | This function retrieves the maximum template size needed by the feature extraction routines. | | |
| Output Parameters | max_enrollment_template_size | The maximum possible size, in bytes, of the memory needed to store feature data from a single enrollment image. | |
| | max_recognition_template_size | The maximum possible size, in bytes, of the memory needed to store feature data from a single verification or identification image. | |
| Return Value | 0 | Success | |
| | Other | Vendor-defined failure | |

6

7 **3.3. 1:1 Verification without enrollment database**

8 **3.3.1. Overview**

9 The 1:1 testing will proceed in three phases: preparation of enrolment templates; preparation of verification templates;
 10 and matching. These are detailed in Table 16.

11 **Table 16 – Functional summary of the 1:1 application**

| Phase | # | Name | Description | Performance Metrics to be reported by NIST |
|----------------------------|----|---------------------|--|---|
| Initialization | I1 | Initialization | Function to allow implementation to read configuration data, if any. | None |
| Enrollment | E1 | Serial enrolment | Given $K \geq 1$ input images of an individual, the implementation will create a proprietary enrollment template. NIST will manage storage of these templates. NIST requires that these operations may be executed in a loop in a single process invocation, or as a sequence of independent process invocations, or a mixture of both. | Statistics of the time needed to produce a template. Statistics of template size. Rate of failure to produce a template and rate of erroneous function. |
| Verification | V1 | Serial verification | Given $K \geq 1$ input images of an individual, the implementation will create a proprietary verification template. NIST will manage storage of these templates. NIST requires that these operations may be executed in a loop in a single process invocation, or as a sequence of independent process invocations, or a mixture of both. | Statistics of the time needed to produce a template. Statistics of template size. Rate of failure to produce a template and rate of erroneous function. |
| Matching (i.e. comparison) | C1 | Serial matching | Given one proprietary enrollment template and one proprietary verification template, compare these and produce a similarity score. NIST requires that these operations may be executed in a loop in a single process invocation, or as a sequence of independent process invocations, or a mixture of both. | Statistics of the time taken to compare two templates. Accuracy measures, primarily reported as DETs. |

12

13

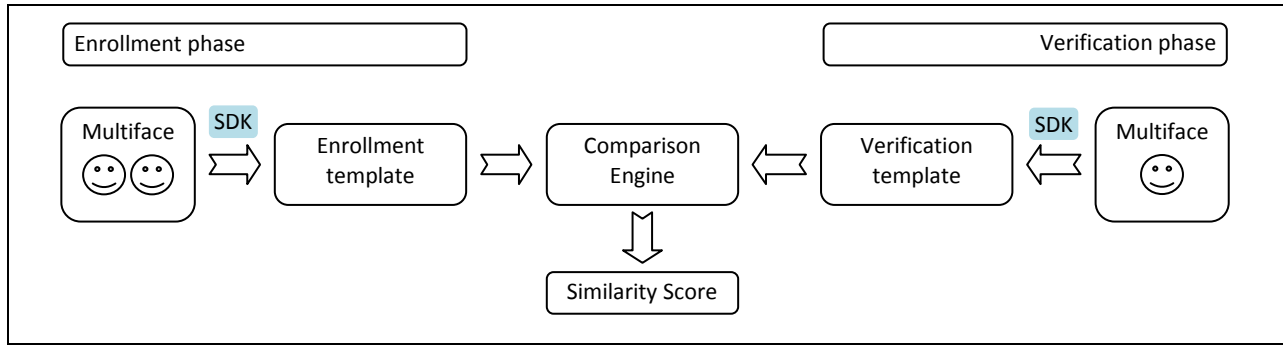


Figure 2- Schematic of verification without enrollment database

1

2 **3.3.2. API**

3 **3.3.2.1. Initialization of the implementation**

4 Before any template generation or matching calls are made, the NIST test harness will make a call to the initialization of
 5 the function in Table 17.

6

Table 17 – SDK initialization

| | | | |
|-------------------|---|---|-------------------------|
| Prototype | int32_t initialize_verification(const char *configuration_location, const char **descriptions, const uint8_t num_descriptions); | | Input Input Input |
| Description | This function initializes the SDK under test. It will be called by the NIST application before any call to the Table 18 functions convert_multiface_to_enrollment_template or convert_multiface_to_verification_template. The SDK under test should set all parameters. | | |
| Input Parameters | configuration_location | A read-only directory containing any vendor-supplied configuration parameters or runtime data files. The name of this directory is assigned by NIST. It is not hardwired by the provider. The names of the files in this directory are hardwired in the SDK and are unrestricted. | |
| | descriptions | A lexicon of labels one of which will be assigned to each image. EXAMPLE: The descriptions could be {"mugshot", "visa", "unknown"}. These labels are provided to the SDK so that it knows to expect images of these kinds. | |
| | num_descriptions | The number of items in the description. In the example above this is 3. | |
| Output Parameters | none | | |
| Return Value | 0 | Success | |
| | 2 | Vendor provided configuration files are not readable in the indicated location. | |
| | 8 | The descriptions are unexpected, or unusable. | |
| | Other | Vendor-defined failure | |

7 **3.3.2.2. Template generation**

8 The functions of Table 18 support role-specific generation of a template data. The format of the templates is entirely
 9 proprietary.

10

Table 18 – Template generation

| | | |
|------------|---|---------------------------|
| Prototypes | int32_t convert_multiface_to_enrollment_template(const MULTIFACE *input_faces, uint32_t *template_size, uint8_t *proprietary_template); | Input Output Output |
| | int32_t convert_multiface_to_verification_template(const MULTIFACE *input_faces, uint32_t *template_size, | Input Output |

| | | |
|-------------------|---|---|
| | uint8_t *proprietary_template, uint8_t *quality); | Output Output |
| Description | This function takes a MULTIFACE , and outputs a proprietary template. The memory for the output template is allocated by the NIST test harness before the call i.e. the implementation shall not allocate memory for the result. In all cases, even when unable to extract features, the output shall be a template record that may be passed to the match_templates function without error. That is, this routine must internally encode "template creation failed" and the matcher must transparently handle this. | |
| Input Parameters | input_faces | An instance of a Table 10 structure. Implementations must alter their behavior according to the number of images contained in the structure. |
| Output Parameters | template_size | The size, in bytes, of the output template |
| | proprietary_template | The output template. The format is entirely unregulated. NIST will allocate a KT byte buffer for this template: The value K is the number of images in the MULTIFACE ; the value T is output by the maximum template size functions of Table 15. |
| | quality | An assessment of image quality. This is optional. The legal values are <ul style="list-style-type: none"> – [0,100] - The value should have a monotonic decreasing relationship with false non-match rate anticipated for this sample if it was compared with a pristine image of the same person. So, a low value indicates high expected FNMR. – 255 - This value indicates a failed attempt to calculate a quality score. – 254 - This values indicates the value was not assigned. |
| Return Value | 0 | Success |
| | 2 | Elective refusal to process this kind of MULTIFACE |
| | 4 | Involuntary failure to extract features (e.g. could not find face in the input-image) |
| | 6 | Elective refusal to produce a template (e.g. insufficient pixels between the eyes) |
| | 8 | Cannot parse input data (i.e. assertion that input record is non-conformant) |
| | Other | Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test. |

1

2 **3.3.2.3. Matching**

3 Matching of one enrollment against one verification template shall be implemented by the function of Table 19.

4

Table 19 – Template matching

| | | |
|-------------------|---|---|
| Prototype | int32_t match_templates(const uint8_t *verification_template, const uint32_t verification_template_size, const uint8_t *enrollment_template, const uint32_t enrollment_template_size, double *similarity); | Input Input Input Input Output |
| Description | This function compares two opaque proprietary templates and outputs a similarity score which need not satisfy the metric properties. NIST will allocate memory for this parameter before the call. When either or both of the input templates are the result of a failed template generation (see Table 18), the similarity score shall be -1 and the function return value shall be 2. | |
| Input Parameters | verification_template | A template from convert_multiface_to_verification_template(). |
| | verification_template_size | The size, in bytes, of the input verification template $0 \leq N \leq 2^{32} - 1$ |
| | enrollment_template | A template from convert_multiface_to_enrollment_template(). |
| | enrollment_template_size | The size, in bytes, of the input enrollment template $0 \leq N \leq 2^{32} - 1$ |
| Output Parameters | similarity | A similarity score resulting from comparison of the templates, on the range [0,DBL_MAX]. See section 2.3.5. |
| Return Value | 0 | Success |
| | 2 | Either or both of the input templates were result of failed feature extraction |
| | Other | Vendor-defined failure |

1 **3.4. 1:N Identification**

2 **3.4.1. Overview**

3 The 1:N application proceeds in two phases, enrollment and identification. The identification phase includes separate
4 pre-search feature extraction stage, and a search stage.

5 The design reflects the following *testing* objectives for 1:N implementations.

- support distributed enrollment on multiple machines, with multiple processes running in parallel
- allow recovery after a fatal exception, and measure the number of occurrences
- allow NIST to copy enrollment data onto many machines to support parallel testing
- respect the black-box nature of biometric templates
- extend complete freedom to the provider to use arbitrary algorithms
- support measurement of duration of core function calls
- support measurement of template size

6 **Table 20 – Procedural overview of the identification test**

| Phase | # | Name | Description | Performance Metrics to be reported by NIST |
|------------|----|---------------------|---|--|
| Enrollment | E1 | Initialization | <p>Give the implementation advance notice of the number of individuals and images that will be enrolled.</p> <p>Give the implementation the name of a directory where any provider-supplied configuration data will have been placed by NIST. This location will otherwise be empty.</p> <p>The implementation is permitted read-write-delete access to the enrollment directory during this phase. The implementation is permitted read-only access to the configuration directory.</p> <p>After enrollment, NIST may rename and relocate the enrollment directory - the implementation should not depend on the name of the enrollment directory.</p> | |
| | E2 | Parallel Enrollment | <p>For each of N individuals, pass multiple images of the individual to the implementation for conversion to a combined template. The implementation will return a template to the calling application.</p> <p>The implementation is permitted read-only access to the enrollment directory during this phase. NIST's calling application will be responsible for storing all templates as binary files. These will not be available to the implementation during this enrollment phase.</p> <p>Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on different computers. The same person will not be enrolled twice.</p> | <p>Statistics of the times needed to enroll an individual.</p> <p>Statistics of the sizes of created templates.</p> <p>The incidence of failed template creations.</p> |
| | E3 | Finalization | <p>Permanently finalize the enrollment directory. This supports, for example, adaptation of the image-processing functions, adaptation of the representation, writing of a manifest, indexing, and computation of statistical information over the enrollment dataset.</p> <p>The implementation is permitted read-write-delete access to the enrollment directory during this phase.</p> | <p>Size of the enrollment database as a function of population size N and the number of images.</p> <p>Duration of this operation. The time needed to execute this function shall be reported with the preceding enrollment times.</p> |
| Pre-search | S1 | Initialization | <p>Tell the implementation the location of an enrollment directory. The implementation could look at the enrollment data.</p> <p>The implementation is permitted read-only access to the enrollment directory during this phase. Statistics of the time needed for this operation.</p> | <p>Statistics of the time needed for this operation.</p> |

| | | | | |
|--------|----|----------------------|---|--|
| | S2 | Template preparation | For each probe, create a template from a set of input images. This operation will generally be conducted in a separate process invocation to step S2. The implementation is permitted no access to the enrollment directory during this phase. The result of this step is a search template. | Statistics of the time needed for this operation. Statistics of the size of the search template. |
| Search | S3 | Initialization | Tell the implementation the location of an enrollment directory. The implementation should read all or some of the enrolled data into main memory, so that searches can commence. The implementation is permitted read-only access to the enrollment directory during this phase. | Statistics of the time needed for this operation. |
| | S4 | Search | A template is searched against the enrolment database. The implementation is permitted read-only access to the enrollment directory during this phase. | Statistics of the time needed for this operation. Accuracy metrics - Type I + II error rates. Failure rates. |

1

2 **3.4.2. Initialization of the enrollment session**

3 Before any enrollment feature extraction calls are made, the NIST test harness will call the initialization function of Table
4 21.

5

Table 21 – Enrollment initialization

| | | |
|-------------------|--|---|
| Prototype | int32_t initialize_enrollment_session(const char *configuration_location, const char *enrollment_directory, const uint32_t num_persons, const uint32_t num_images, const char **descriptions, const uint8_t num_descriptions); | |
| | | Input |
| | | Input |
| | | Input |
| | | Input |
| | | Input |
| Description | This function initializes the SDK under test and sets all needed parameters. This function will be called N=1 times by the NIST application immediately before any $M \geq 1$ calls to convert_multiface_to_enrollment_template. The SDK should tolerate execution of $P > 1$ processes on the same machine each of which may be reading and writing to the enrollment directory. This function may be called P times and these may be running simultaneously and in parallel. | |
| Input Parameters | configuration_location | A read-only directory containing any vendor-supplied configuration parameters or run-time data files. |
| | enrollment_directory | The directory will be initially empty, but may have been initialized and populated by separate invocations of the enrollment process. When this function is called, the SDK may populate this folder in any manner it sees fit. Permissions will be read-write-delete. |
| | num_persons | The number of persons who will be enrolled $0 \leq N \leq 2^{32} - 1$ (e.g. 1million) |
| | num_images | The total number of images that will be enrolled, summed over all identities $0 \leq M \leq 2^{32} - 1$ (e.g. 1.8 million) |
| | descriptions | A lexicon of labels one of which will be assigned to each enrollment image. EXAMPLE: The descriptions could be {"mugshot", "visa"}. NOTE: The identification search images may or may not be labeled. An identification image may carry a label not in this set of labels. |
| | num_descriptions | The number of items in the description. In the example above this is 2. |
| Output Parameters | none | |
| Return Value | 0 | Success |
| | 2 | The configuration data is missing, unreadable, or in an unexpected format. |
| | 4 | An operation on the enrollment directory failed (e.g. permission, space). |

| | | |
|--|-------|---|
| | 6 | The SDK cannot support the number of persons or images. |
| | 8 | The descriptions are unexpected, or unusable. |
| | Other | Vendor-defined failure |

1
2
3
4

3.4.3. Enrollment

A **MULTIFACE** is converted to a single enrollment template using the function of Table 22.

Table 22 – Enrollment feature extraction

| | | |
|-------------------|--|--|
| Prototypes | int32_t convert_multiface_to_enrollment_template(const MULTIFACE *input_faces, EYEPAIR *const *output_eyes, uint32_t *template_size, uint8_t *proprietary_template); | |
| | | Input |
| | | Output |
| | | Output |
| | | Output |
| Description | <p>This function takes a MULTIFACE, and outputs a proprietary template. The memory for the output template is allocated by the NIST test harness before the call i.e. the implementation shall not allocate memory for the result.</p> <p>If the function executes correctly (i.e. returns a zero exit status), the NIST calling application will store the template. The NIST application will concatenate the templates and pass the result to the enrollment finalization function (see section 3.4.4).</p> <p>If the function gives a non-zero exit status:</p> <ul style="list-style-type: none"> – If the exit status is 8, NIST will debug, otherwise – the test driver will ignore the output template (the template may have any size including zero) – the event will be counted as a failure to enroll. Such an event means that this person can never be identified correctly. <p>IMPORTANT. NIST's application writes the template to disk. The implementation must not attempt writes to the enrollment directory (nor to other resources). Any data needed during subsequent searches should be included in the template, or created from the templates during the enrollment finalization function of section 3.4.4.</p> | |
| Input Parameters | input_faces | An instance of a Table 10 structure. Implementations must alter their behavior according to the number of images contained in the structure. |
| Output Parameters | output_eyes | For each input image in the MULTIFACE the function shall return the estimated eye centers. The calling application will pre-allocate the correct number of EYEPAIR structures (i.e. one for each image in the MULTIFACE). |
| | template_size | The size, in bytes, of the output template |
| | proprietary_template | The format is entirely unregulated. NIST will allocate a KT byte buffer for this template: The value K is the number of images in the MULTIFACE ; the value T is output by the maximum enrollment template size function of Table 15. |
| Return Value | 0 | Success |
| | 2 | Elective refusal to process this kind of MULTIFACE |
| | 4 | Involuntary failure to extract features (e.g. could not find face in the input-image) |
| | 6 | Elective refusal to produce a template (e.g. insufficient pixels between the eyes) |
| | 8 | Cannot parse input data (i.e. assertion that input record is non-conformant) |
| | Other | Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test. |

5
6
7
8
9
10
11
12

3.4.4. Finalize enrollment

After all templates have been created, the function of Table 23 will be called. This freezes the enrollment data. After this call the enrollment dataset will be forever read-only. This API does not support interleaved enrollment and search phases.

The function allows the implementation to conduct, for example, statistical processing of the feature data, indexing and data re-organization. The function may alter the file structure. It may increase or decrease the size of the stored data. No output is expected from this function, except a return code.

1

Table 23 – Enrollment finalization

| | | |
|-------------------|--|--|
| Prototypes | int32_t finalize_enrollment (| |
| | const char *enrolment_directory, | Input |
| | const char *edb_name, | Input |
| | const char *edb_manifest_name); | Input |
| Description | <p>This function takes the name of the top-level directory where enrollment database (EDB) and its manifest have been stored. These are described in section 2.4. The enrollment directory permissions will be read + write.</p> <p>The function supports post-enrollment vendor-optional book-keeping operations and statistical processing. The function will generally be called in a separate process after all the enrollment processes are complete.</p> <p>This function should be tolerant of being called two or more times. Second and third invocations should probably do nothing.</p> | |
| Input Parameters | enrollment_directory | The top-level directory in which enrollment data was placed. This variable allows an implementation to locate any private initialization data it elected to place in the directory. |
| | edb_name | The name of a single file containing concatenated templates, i.e. the EDB of section 2.4. While the file will have read-write-delete permission, the SDK should only alter the file if it preserves the necessary content, in other files for example. The file may be opened directly. It is not necessary to prepend a directory name. |
| | edb_manifest_name | The name of a single file containing the EDB manifest of section 2.4. The file may be opened directly. It is not necessary to prepend a directory name. |
| Output Parameters | None | |
| Return Value | 0 | Success |
| | 2 | Cannot locate the input data - the input files or names seem incorrect. |
| | 4 | An operation on the enrollment directory failed (e.g. permission, space). |
| | 6 | One or more template files are in an incorrect format. |
| | Other | Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test. |

2 **3.4.5. Pre-search feature extraction**

3 **3.4.5.1. Initialization**

4 Before MULTIFACES are sent to the identification feature extraction function, the test harness will call the initialization
5 function in Table 24.

6

Table 24 – Identification feature extraction initialization

| | | |
|-------------------|---|--|
| Prototype | int32_t initialize_feature_extraction_session(| |
| | const char * configuration_location, | Input |
| | const char * enrolment_directory); | Input |
| Description | <p>This function initializes the SDK under test and sets all needed parameters. This function will be called N=1 times by the NIST application immediately before any $M \geq 1$ calls to convert_multiface_to_identification_template. The SDK should tolerate execution of $P > 1$ processes on the same machine each of which can read the configuration directory. This function may be called P times and these may be running simultaneously and in parallel.</p> <p>The implementation has read-only access to its prior enrollment data.</p> | |
| Input Parameters | configuration_location | A read-only directory containing any vendor-supplied configuration parameters or run-time data files. |
| | enrollment_directory | The top-level directory in which enrollment data was placed and then finalized by the implementation. The implementation can parameterize subsequent template production on the basis of the enrolled dataset. |
| Output Parameters | none | |
| Return Value | 0 | Success |
| | 2 | The configuration data is missing, unreadable, or in an unexpected format. |

| | | |
|--|-------|--|
| | 4 | An operation on the enrollment directory failed (e.g. permission). |
| | Other | Vendor-defined failure |

1
2
3
4
5

3.4.5.2. Feature extraction

A **MULTIFACE** is converted to an atomic identification template using the function of Table 25. The result may be stored by NIST, or used immediately. The SDK shall not attempt to store any data.

Table 25 – Identification feature extraction

| | | |
|-------------------|---|--|
| Prototypes | int32_t convert_multiface_to_identification_template(| |
| | const MULTIFACE *input_faces, | Input |
| | EYEPAIR * const *output_eyes, | Output |
| | uint32_t *template_size, | Output |
| | uint8_t *identification_template); | Output |
| Description | <p>This function takes a MULTIFACE, and outputs a proprietary template. The memory for the output template is allocated by the NIST test harness before the call i.e. the implementation shall not allocate memory for the result.</p> <p>If the function executes correctly, it returns a zero exit status. The NIST calling application may commit the template to permanent storage, or may keep it only in memory (the vendor implementation does not need to know). If the function returns a non-zero exit status, the output template will be not be used in subsequent search operations.</p> <p>The function shall not have access to the enrollment data, nor shall it attempt access.</p> | |
| Input Parameters | input_faces | An instance of a Table 10 structure. Implementations must alter their behavior according to the number of images contained in the structure. |
| Output Parameters | output_eyes | For each input image in the MULTIFACE the function shall return the estimated eye centers. The calling application will pre-allocate the correct number of EYEPAIR structures (i.e. one for each image in the MULTIFACE). |
| | template_size | The size, in bytes, of the output template |
| | identification_template | The output template for a subsequent identification search. The format is entirely unregulated. NIST will allocate a KT byte buffer for this template: The value K is the number of images in the input MULTIFACE ; the value T is output by the maximum enrollment template size function of Table 15. |
| Return Value | 0 | Success |
| | 2 | Elective refusal to process this kind of MULTIFACE |
| | 4 | Involuntary failure to extract features (e.g. could not find face in the input-image) |
| | 6 | Elective refusal to produce a template (e.g. insufficient pixels between the eyes) |
| | 8 | Cannot parse input data (i.e. assertion that input record is non-conformant) |
| | Other | Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test. |

3.4.6. Initialization

The function of Table 26 will be called once prior to one or more calls of the searching function of Table 27. The function might set static internal variables so that the enrollment database is available to the subsequent identification searches.

Table 26 - Identification initialization

| | | |
|------------------|---|---|
| Prototype | int32_t initialize_identification_session(| |
| | const char *configuration_location, | Input |
| | const char *enrollment_directory); | Input |
| Description | This function reads whatever content is present in the enrollment_directory, for example a manifest placed there by the finalize_enrollment function. | |
| Input Parameters | configuration_location | A read-only directory containing any vendor-supplied configuration parameters or run-time data files. |
| | enrollment_directory | The top-level directory in which enrollment data was placed. |
| Return Value | 0 | Success |
| | Other | Vendor-defined failure |

6
7
8
9

1
2
3
4
5

3.4.7. Search

The function of Table 27 compares a proprietary identification template against the enrollment data and returns a candidate list.

Table 27 – Identification search

| | | | |
|-------------------|---|--|--------|
| Prototype | int32_t identify_template(const uint8_t *identification_template, const uint32_t identification_template_size, const uint32_t candidate_list_length, CANDIDATE * const *candidate_list); | | |
| | | | Input |
| | | | Input |
| | | | Output |
| Description | This function searches a template against the enrollment set, and outputs a list of candidates. NIST will allocate memory for the candidates before the call. | | |
| Input Parameters | identification_template | A template from convert_multiface_to_identification_template() - If the value returned by that function was non-zero the contents of identification_template will not be used and this function (i.e. identify_template) will not be called. | |
| | identification_template_size | The size, in bytes, of the input identification template $0 \leq N \leq 2^{32} - 1$ | |
| | candidate_list_length | The number of candidates the search should return | |
| Output Parameters | candidate_list | An array of "candidate_list_length" pointers to candidates. The datatype is defined in section 2.5. Each candidate shall be populated by the implementation. The candidates shall appear in descending order of similarity score - i.e. most similar entries appear first. | |
| Return Value | 0 | Success | |
| | 2 | The input template was defective. | |
| | Other | Vendor-defined failure | |

6
7
8
9

NOTE: Ordinarily the calling application will set the input candidate list length to operationally typical values, say $0 \leq L \leq 200$, and $L \ll N$. However, there is interest in the presence of mates much further down the candidate list. We may therefore extend the candidate list length such that L approaches N.

3.5. 1:1 Verification with enrollment database

For verification with an enrollment database, the sequence of operations and the enrollment functions are identical to those given in sections 3.4.2, 3.4.3, 3.4.4 and 3.4.6. The only difference lies in the actual recognition step: As shown in Table 28, the verification call accepts an explicit claim of identity.

NOTE: We will not use a class B enrollment database in a class C identification search, and vice-versa.

Table 28 – Verification against an enrolled identity

| | | | |
|------------------|---|---|--------|
| Prototype | int32_t verify_template(const uint8_t *verification_template, const uint32_t verification_template_size, const uint32_t enrolled_identity_claim, double *similarity); | | |
| | | | Input |
| | | | Input |
| | | | Output |
| Description | This function compares a template against a specified entry of the enrollment set, and outputs a similarity score. The returned similarity is a similarity score. When either the input template or the enrolled data for the claimed identity are the result of the failed template generation, the similarity score shall be -1 and the function return value shall be 2. | | |
| Input Parameters | verification_template | A template from convert_multiface_to_identification_template() which will have been initialized with initialize_feature_extraction_session(). | |
| | verification_template_size | The size, in bytes, of the input verification template $0 \leq N \leq 2^{32} - 1$ | |
| | enrolled_identity_claim | A Template ID that exists in the EDB. The EDB was passed into the | |

| | | |
|-------------------|------------|---|
| | | finalize_enrollment function of section 3.4.4. The ID appears in column 1 of Table 12. This value is NOT an integer index into the enrollment set. The face represented by the verification template is claimed to be that of this enrolled identity. |
| Output Parameters | similarity | A similarity score resulting from comparison of the templates, on the range [0,DBL_MAX]. See section 2.3.5. |
| Return Value | 0 | Success |
| | 2 | The input template was result of failed feature extraction |
| | 4 | Assertion that the input Template ID did not exist in the EDB. |
| | Other | Vendor-defined failure |

1 **3.6. Pose conformance estimation**

2 **3.6.1. Overview**

3 The functions of this section support testing of whether a face in an image has frontal pose. This supports conformance
 4 testing of, for example, the Full Frontal specification of the ISO standard [ISO]. The goal is to support a marketplace of
 5 products for acquisition time assessment of pose. This is important because pose is arguably the most influential
 6 covariate on face recognition error rates, and is not generally controllable by design of the acquisition system.

7 NIST encourages participants in this study to implement real-time video rate implementations, and also slower more
 8 accurate methods. The test and API is not intended to cover multi-frame techniques (e.g. from video) nor tracking.

9 The functional specification here supports a DET analysis in which false-rejection of actually frontal images can be traded
 10 off against false acceptance of non-frontal images via a frontal-conformance parameter, t. The exact meaning of the
 11 "frontality" value returned by this function is not regulated by the NIST specification. However a reasonable
 12 implementation would embed a monotonic relationship between the output value and non-frontal angle (i.e. compound
 13 rotation involving azimuthal head yaw and pitch).

14 The formal ISO requirement is for five degree rotation in pitch and yaw. While the ISO standard establishes an eight
 15 degree limit on roll angle, this is of less importance. NIST will not consider roll angle.

16 **3.6.2. API**

17 Table 30 provides a function for computing a pose conformance measurement from an image. Although the function
 18 makes use of the **MULTIFACE** structure (for consistency with the rest of the API), the function will ordinarily be invoked with
 19 just a single image. The initialization function of Table 29 will be called before one or more calls to the pose conformance
 20 assessment function.

21 **Table 29 - Initialization of the pose conformance estimation SDK**

| | | |
|-------------------|--|--|
| Prototype | int32_t initialize_frontal_pose_estimation(const char *configuration_location); | Input |
| Description | This function initializes the SDK under test. It will be called by the NIST application before any call to the Table 30 functions. The SDK under test should set all parameters. | |
| Input Parameters | configuration_location | A read-only directory containing any vendor-supplied configuration parameters or run-time data files. The name of this directory is assigned by NIST. It is not hardwired by the provider. The names of the files in this directory are hardwired in the SDK and are unrestricted. |
| Output Parameters | none | |
| Return Value | 0 | Success |
| | 2 | Vendor provided configuration files are not readable in the indicated location. |
| | Other | Vendor-defined failure |

22

23

Table 30 - Pose conformance estimation

| | | |
|------------|--|--|
| Prototypes | int32_t estimate_frontal_pose_conformance(| |
|------------|--|--|

| | | |
|-------------------|--|--|
| | const MULTIFACE *input_faces, double *non_frontalities); | Input Output |
| Description | This function takes a MULTIFACE , and outputs a non-frontality value for each image. The images of the MULTIFACE will be independent - i.e. they will generally not be frames from a video. The non-frontality value should increase with larger deviations from frontal pose. | |
| Input Parameters | input_faces | An instance of a Table 10 structure. |
| Output Parameters | non-frontalities | For the i-th input image, the i-th output value will indicate how from frontal the head pose is. The value should be on the range [0,1]. |
| Return Value | 0 | Success |
| | 2 | Elective refusal to process this kind of MULTIFACE |
| | 4 | Involuntary failure to extract features (e.g. could not find face in the input-image) |
| | 8 | Cannot parse input data (i.e. assertion that input record is non-conformant) |
| | Other | Vendor-defined failure. Failure codes must be documented and communicated to NIST with the submission of the implementation under test. |

1 **3.7. Software and Documentation**

2 **3.7.1. SDK Library and Platform Requirements**

3 Participants shall provide NIST with binary code only (i.e. no source code). Header files (".h") are allowed, but these shall
 4 not contain intellectual property of the company nor any material that is otherwise proprietary. It is preferred that the
 5 SDK be submitted in the form of a single static library file (ie. ".lib" for Windows or ".a" for Linux). However, dynamic and
 6 shared library files are permitted.

7 The core library shall be named according to Table 31. Additional dynamic or shared library files may be submitted that
 8 support this "core" library file (i.e. the "core" library file may have dependencies implemented in these other libraries).

9 **Table 31 - Implementation library filename convention**

| Form | libMBE_provider_class_sequence.ending | | | | |
|--|--|--|--|--|-------------------------|
| Underscore delimited parts of the filename | libMBE | provider | classes | sequence | ending |
| Description | First part of the name, required to be this. | Single word name of the main provider EXAMPLE: Acme | Function classes supported in Table 3. EXAMPLE: C | A two digit decimal identifier to start at 00 and increment by 1 every time any SDK is sent to NIST. EXAMPLE: 07 | One of .so .a .dll .lib |
| Example | libMBE_Acme_C_07.a | | | | |

10

11 NIST will report the size of the supplied libraries.

12 **3.7.2. Configuration and vendor-defined data**

13 The implementation under test may be supplied with configuration files and supporting data files. The total size of the
 14 SDK, that is all libraries, include files, data files and initialization files shall be less than or equal to 1 073 741 824 bytes =
 15 1024³ bytes.

16 NIST will report the size of the supplied configuration files.

17 **3.7.3. Software environment and linking**

18 NIST will link the provided library file(s) to various ISO 98/99 "C/C++" language test driver applications developed by NIST.
 19 Participants are required to provide their library in a format that is linkable using "gcc" with the NIST test driver, which is
 20 compiled with gcc version 4.X. These use libc. The link command might be:

21 `gcc -I. -Wall -m64 -o mbetest mbetest.c -L. -lMBE_Acme_C_07 -lpthread`

1 On request, NIST will allow use of "g++" for linking, but the API must have "C" linkage. The Standard C++ library is
 2 available¹¹. The prototypes of this document will be written to a file "mbe.h" which will be included via

```
extern "C"
{
#include <mbe.h>
}
```

3 NIST will handle all input of images via the JPEG and PNG libraries, sourced, respectively from <http://www.ijg.org/> and see
 4 <http://libpng.org>.

5 All compilation and testing will be performed on x86 platforms. Thus, participants are strongly advised to verify library-
 6 level compatibility with gcc (on an equivalent platform) prior to submitting their software to NIST to avoid linkage
 7 problems later on (e.g. symbol name and calling convention mismatches, incorrect binary file formats, etc.).

8 NIST will attempt to support Intel Integrated Performance Primitives (Intel IPP) and "icc" compiled libraries. See the
 9 processor specifications in section 1.19.

10 Windows libraries will be linked using gcc or g++ running under the MinGW layer¹². This supports 64 bit binaries.
 11 Windows-compiled libraries have been successfully linked in prior NIST tests.

12 Dependencies on external dynamic/shared libraries such as compiler-specific development environment libraries are
 13 discouraged. If absolutely necessary, external libraries must be provided to NIST upon prior approval by the Test Liaison.

14 3.7.4. Installation and Usage

15 The SDK must install easily (i.e. one installation step with no participant interaction required) to be tested, and shall be
 16 executable on any number of machines without requiring additional machine-specific license control procedures or
 17 activation.

18 The SDK shall be installable using simple file copy methods. It shall not require the use of a separate installation program.

19 The SDK shall neither implement nor enforce any usage controls or limits based on licenses, number of executions,
 20 presence of temporary files, etc. The SDKs shall remain operable until April 30 2011.

21 Hardware (e.g. USB) activation dongles are not acceptable.

22 3.7.5. Hard disk space

23 MBE participants should inform NIST if their implementations require more than 100K of persistent storage, per enrolled
 24 image on average.

25 3.7.6. Documentation

26 Participants shall provide complete documentation of the SDK and detail any additional functionality or behavior beyond
 27 that specified here. The documentation must define all (non-zero) vendor-defined error or warning return codes.

28 3.7.7. Modes of operation

29 Individual SDKs provided shall not include multiple "modes" of operation, or algorithm variations. No switches or options
 30 will be tolerated within one library. For example, the use of two different "coders" by an feature extractor must be split
 31 across two separate SDK libraries, and two separate submissions.

32 3.7.8. Watermarking of images

33 The SDK functions shall not watermark or otherwise steganographically mark up the images.

¹¹ This includes the compiler that installs with RedHat, which is Target: x86_64-redhat-linux configured with: `./configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --enable-shared --enable-threads=posix --enable-checking=release --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-libgjc-multifile --enable-languages=c,c++,objc,obj-c++,java,fortran,ada --enable-java-awt=gtk --disable-dssi --enable-plugin --with-java-home=/usr/lib/jvm/java-1.4.2-gcj-1.4.2.0/jre --with-cpu=generic --host=x86_64-redhat-linux Thread model: posix gcc version 4.1.2 20070626 (Red Hat 4.1.2-14)`

The libraries are what shipped with RH 5.1: `/usr/lib64/libstdc++.so.6.0.8 lib/libc.so.6 -> libc-2.5.so /lib/libm.so.6 -> libm-2.5.so`

¹² According to <http://www.mingw.org/> MinGW, a contraction of "Minimalist GNU for Windows", is a part of the GNU Compiler Collection (GCC), and GNU Binutils, for use in the development of native Microsoft Windows applications.

1 **3.8. Runtime behavior**

2 **3.8.1. Interactive behavior**

3 The SDK will be tested in non-interactive “batch” mode (i.e. without terminal support). Thus, the submitted library shall
 4 not use any interactive functions such as graphical user interface (GUI) calls, or any other calls which require terminal
 5 interaction e.g. reads from “standard input”.

6 **3.8.2. Error codes and status messages**

7 The SDK will be tested in non-interactive “batch” mod, without terminal support. Thus, the submitted library shall run
 8 quietly, i.e. it should not write messages to "standard error" and shall not write to “standard output”. An SDK may write
 9 debugging messages to a log file - the name of the file must be declared in documentation.

10 **3.8.3. Exception Handling**

11 The application should include error/exception handling so that in the case of a fatal error, the return code is still
 12 provided to the calling application.

13 **3.8.4. External communication**

14 Processes running on NIST hosts shall not side-effect the runtime environment in any manner, except for memory
 15 allocation and release. Implementations shall not write any data to external resource (e.g. server, file, connection, or
 16 other process), nor read from such. If detected, NIST will take appropriate steps, including but not limited to, cessation of
 17 evaluation of all implementations from the supplier, notification to the provider, and documentation of the activity in
 18 published reports.

19 **3.8.5. Stateful behavior**

20 All components in this test shall be stateless, except as noted. This applies to face detection, feature extraction and
 21 matching. Thus, all functions should give identical output, for a given input, independent of the runtime history. NIST
 22 will institute appropriate tests to detect stateful behavior. If detected, NIST will take appropriate steps, including but not
 23 limited to, cessation of evaluation of all implementations from the supplier, notification to the provider, and
 24 documentation of the activity in published reports.

25 **4. References**

| | |
|-----------------|--|
| FRVT 2002 | Face Recognition Vendor Test 2002: Evaluation Report, NIST Interagency Report 6965, P. Jonathon Phillips, Patrick Grother, Ross J. Micheals, Duane M. Blackburn, Elham Tabassi, Mike Bone |
| FRVT 2002b | Face Recognition Vendor Test 2002: Supplemental Report, NIST Interagency Report 7083, Patrick Grother |
| FRVT 2006 | P. Jonathon Phillips, W. Todd Scruggs, Alice J. O’Toole, Patrick J. Flynn, Kevin W. Bowyer, Cathy L. Schott, and Matthew Sharpe. "FRVT 2006 and ICE 2006 Large-Scale Results." NISTIR 7408, March 2007. |
| AN27 | NIST Special Publication 500-271: American National Standard for Information Systems — <i>Data Format for the Interchange of Fingerprint, Facial, & Other Biometric Information – Part 1</i> . (ANSI/NIST ITL 1-2007). Approved April 20, 2007. |
| MINEX | P. Grother et al., <i>Performance and Interoperability of the INCITS 378 Template</i> , NIST IR 7296 http://fingerprint.nist.gov/minex04/minex_report.pdf |
| MOC | P. Grother and W. Salamon, <i>MINEX II - An Assessment of ISO/IEC 7816 Card-Based Match-on-Card Capabilities</i> http://fingerprint.nist.gov/minex/minexII/NIST_MOC_ISO_CC_interop_test_plan_1102.pdf |
| PERFSTD INTEROP | ISO/IEC 19795-4 — Biometric Performance Testing and Reporting — Part 4: Interoperability Performance Testing. Posted as document 37N2370 . The standard was published in 2007. It can be purchased from ANSI at http://webstore.ansi.org/ . |
| ISO STD05 | ISO/IEC 19794-5:2005 — <i>Information technology — Biometric data interchange formats — Part 5: Face image data</i> . The standard was published in 2005, and can be purchased from ANSI at http://webstore.ansi.org/ Multipart standard of "Biometric data interchange formats". This standard was published in 2005. It was amended twice to include guidance to photographers, and then to include 3D information. Two corrigenda were published. All these changes and new material is currently being incorporated in revision of the standard. Publication is likely in early 2011. The |

documentary history is as follows.

1. ISO/IEC 19794-5: Information technology — Biometric data interchange formats — Part 5:Face image data. First edition: 2005-06-15.
2. International Standard ISO/IEC 19794-5:2005 Technical Corrigendum 1: Published 2008-07-01
3. International Standard ISO/IEC 19794-5:2005 Technical Corrigendum 2: Published 2008-07-01
4. Information technology — Biometric data interchange formats — Part 5: Face image data AMENDMENT 1: Conditions for taking photographs for face image data. Published 2007-12-15
5. Information technology — Biometric data interchange formats — Part 5: Face image data AMENDMENT 2: Three dimensional image data.
6. JTC 1/SC37/N3303. FCD text of the second edition. Contact pgrother AT nist DOT gov for more information.

1
2

Annex A

Submission of Implementations to the MBE 2010 STILL

1

2

A.1 Submission of implementations to NIST

4 NIST requires that all software, data and configuration files submitted by the participants be signed and encrypted.
 5 Signing is done with the participant's private key, and encryption is done with the NIST public key. The detailed
 6 commands for signing and encrypting are given here: http://iris.nist.gov/irex/crypto_protection.pdf [Link is correct Jan
 7 28 2010].

8 NIST will validate all submitted materials using the participant's public key, and the authenticity of that key will be verified
 9 using the key fingerprint. This fingerprint must be submitted to NIST by writing it on the signed participation agreement.

10 By encrypting the submissions, we ensure privacy; by signing the submission, we ensure authenticity (the software
 11 actually belongs to the submitter). **NIST will not accept into MBE any submission that is not signed and encrypted. NIST
 12 accepts no responsibility for anything that is transmitted to NIST that is not signed and encrypted with the NIST public
 13 key.**

A.2 How to participate

15 Those wishing to participate in MBE testing must do all of the following, on the schedule listed on Page 2.

- 16 – IMPORTANT: Follow the instructions for cryptographic protection of your SDK and data here.
 17 http://iris.nist.gov/irex/crypto_protection.pdf
- 18 – Send a signed and fully completed copy of the *Application to Participate in the Multiple Biometric Evaluation (MBE)*
 19 *2010 Still Face Track*. This is available at This must identify, and include signatures from, the Responsible Parties as
 20 defined in section **Error! Reference source not found.** The properly signed MBE Application to Participate shall be
 21 sent to NIST as a PDF.
- 22 – Provide an SDK (Software Development Kit) library which complies with the API (Application Programmer Interface)
 23 specified in this document.
 - 24 • Encrypted data and SDKs below 20MB can be emailed to NIST at mbe2010@nist.gov
 - 25 • Encrypted data and SDKS above 20MB shall be
 - 26 § Made available as a file.zip.gpg or file.zip.asc download from a generic webserver¹³, or:
 - 27 § Mailed as a file.zip.gpg or file.zip.asc on CD / DVD to NIST at this address:

| | |
|---|--|
| MBE Test Liaison (A203) 100 Bureau Drive A203/Tech225/Stop 8940 NIST Gaithersburg, MD 20899-8940 USA | In cases where a courier needs a phone number please use NIST shipping and handling on: 301 -- 975 -- 6296. |
|---|--|

28

A.3 Implementation validation

30 Registered Participants will be provided with a small validation dataset and program available on the website
 31 <http://face.nist.gov/mbe> at http://face.nist.gov/mbe/validation_2.tar.gz

32 The validation test programs shall be compiled by the provider. The output of these programs shall be submitted to NIST.

¹³ NIST will not register, or establish any kind of membership, on the provided website.

MBE 2010

- 1 Prior to submission of the SDK and validation data, the Participant must verify that their software executes on the
- 2 validation images, and produces correct similarity scores and templates.
- 3 Software submitted shall implement the MBE API Specification as detailed in the body of this document.
- 4 Upon receipt of the SDK and validation output, NIST will attempt to reproduce the same output by executing the SDK on
- 5 the validation imagery, using a NIST computer. In the event of disagreement in the output, or other difficulties, the
- 6 Participant will be notified.
- 7
- 8

Annex B

Frequently asked Questions

1
2

3 **Q1: Can you provide additional information about controlled and uncontrolled still images.**

4 A1: Controlled stills are facial images nominally taken under studio lighting. Representative examples are all the images in
5 the FERET dataset and FRGC images from Experiment 1. Uncontrolled stills are images that are not taken under studio
6 lighting. Uncontrolled stills maybe taken in hallways and atriums with ambient lighting, and images taken outside.

7 **Q2. What will the image sizes be in FRVT 2006?**

8 A2: A maximum of 4288 by 2848 pixels. The images can either be in portrait or landscape mode. One of the goals of the
9 MBE 2010 still face track is to measure the effects of compression and geometric normalization on performance. Towards
10 this end, images will be compressed and resized till the algorithms break.

11 **Q3. What is the range in the size of the face in the still images?**

12 A3: What is important is not the absolute size of the face, but rather the ratio of the size of the image to the size of the
13 face. For still images, the size of the face will be measured in the number of pixels between the centers of the eyes. For
14 calculations to this answer, the size of the image will be measured by the smaller of the two dimensions. For controlled
15 still images, the largest faces could have a size ratio of 2.0, and the smallest could have a size ratio of 15.0. For
16 uncontrolled still images, the largest faces could have a size ratio of 4.0, and the smallest could have a size ratio of 30.0.

17 **Q4. What is the range of roll (in plane rotation) of the faces?**

18 A4: The images can be in either landscape or portrait mode. The roll (in plane rotation) is measured as the angle between
19 the line through the centers of the eyes and the x-axis (horizontal axis). For controlled images, the mean angle is
20 approximately 0 degrees, standard deviation of approximately 5.0 degrees, and limits of rotation of (approximately)
21 plus/minus 20 degrees. For uncontrolled images, the mean angle is approximately 0 degrees, standard deviation of
22 approximately 6.0 degrees, and limits of rotation of (approximately) plus/minus 20 degrees.

23 **Q5. Your previous answer of face size in still images was expressed as a ratio of face size to image size. Can
24 you please provide absolute numbers for the face size?**

25 A5: For controlled still images, the minimum size face will be 10 pixels, and the maximum will be 800 pixels. For
26 uncontrolled still images, the minimum size face will be 10 pixels, and the maximum size face will be 350 pixels. The size
27 of a face is measured as the number of pixels between the centers of the eyes.