Face Recognition Vendor Test Ongoing

General Evaluation Specifications

Patrick Grother
Mei Ngan
Kayee Hanaoka
Information Access Division
Information Technology Laboratory

Contact via frvt@nist.gov

April 6, 2023

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Revision History

2

3

4

Date	Version	Description
April 1, 2019	1.0	Initial document
September 9, 2020	1.1	Update operating system to CentOS 8.2 and compiler to g++ 8.3.1 Adjust the legal similarity score range
February 14, 2022	1.2	Update operating system to Ubuntu 20.04.3 and compiler to g++ 9.3.0
August 29, 2022	2.0	Remove Section 8 and link to current/definitive data structures header file on GitHub
April 6, 2023	3.0	Add support for iris images; and make text generic for face or iris or multimodal

5

Table of Contents

32 **List of Tables**

33

1. Audience

Participation in FRVT is open to any organization worldwide. There is no charge for participation. The target audience is researchers and developers of FR algorithms. While NIST intends to evaluate stable technologies that could be readily made operational, the test is also open to experimental, prototype and other technologies. All algorithms **must** be submitted as implementations of the API defined in the specific test's API document.

2. Rules for Participation

2.1. Participation Agreement

A participant must properly follow, complete, and submit the FRVT Participation Agreement. This must be done once, either prior or in conjunction with the very first algorithm submission. It is not necessary to do this for each submitted implementation thereafter UNLESS there are major organizational changes to the submitting entity.

NOTE   If an organization updates their cryptographic signing key, they must send a new completed participation agreement submission for this evaluation, with the fingerprint of their public key.

2.2. Validation

Prior to submission, all participants must run their software through the provided corresponding validation package for the test they wish to enter. The validation package will be made available at https://github.com/usnistgov/frvt. The purpose of validation is to ensure consistent algorithm output between the participant's execution and NIST's execution.

2.3. Number and Schedule of Submissions

Participants may send one submission as often as every four calendar months from the last submission for evaluation. NIST will evaluate implementations on a first-come-first-served basis, and quickly publish results.

3. Reporting

Unless otherwise specified for a specific test, for all algorithms that complete the evaluations, NIST will post performance results on the NIST FRVT website. NIST will maintain an email list to inform interested parties of updates to the website. Artifacts will include a leaderboard highlighting the top performing submissions in various areas (e.g., accuracy, speed etc.) and individual implementation-specific report cards. NIST will maintain reporting on the two most recent algorithm submissions from any organization. In the event an algorithm is no longer operable (e.g., license expiration, etc.), that algorithm will be retired from the evaluation. Prior submission results will be archived but remain accessible via a public link.

Important: This is an open test in which NIST will identify the algorithm and the developing organization. Algorithm results will be attributed to the developer. Results will be machine generated (i.e. scripted) and will include timing, accuracy and other performance results. These will be posted alongside results from other implementations. Results will be expanded and modified as additional implementations are tested, and as analyses are implemented. Results may be regenerated on-the-fly, usually whenever additional implementations complete testing, or when new analysis is added.

NIST may additionally report results in workshops, conferences, conference papers and presentations, journal articles and technical reports.

3.1. Version Control

Developers must submit a version.txt file in the doc/ folder that accompanies their algorithm – see Section 6.4. The string in this file should allow the developer to associate results that appear in NIST reports with the submitted algorithm. This is intended to allow end-users to obtain productized versions of the prototypes submitted to NIST. NIST will publish the contents of version.txt. NIST has previously published MD5 hashes of the core libraries for this purpose.

4. Hardware specification

74 NIST intends to support high performance by specifying the runtime hardware beforehand. There are several types of
75 computer blades that may be used in the testing. Each machine has at least 128 GB of memory. We anticipate that 16
76 processes can be run without time slicing, though NIST will handle all multiprocessing work via `fork()`[1]. Participant-
77 initiated multiprocessing is not permitted.

78 All implementations shall use 64 bit addressing.

79 NIST intends to support highly optimized algorithms by specifying the runtime hardware. We use multiple computers
80 including the following:

81 — Intel® Xeon® Gold 6254 CPU @ 3.10GHz

82 — Intel® Xeon® E5-2630 v4 CPU @ 2.20GHz[2]

83 — Intel® Xeon® E5-2680 v4 CPU @ 2.4GHz[2]

84 — Intel® Xeon® Gold 6140 CPU @ 2.30GHz[3]

85 All timing tests will be measured on Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz. FRVT tests will not support the use of
86 Graphics Processing Units (GPUs).

5. Operating system, compilation, and linking environment

88 The operating system that the submitted implementations shall run on will be released as a downloadable file accessible
89 from https://nigos.nist.gov/evaluations/ubuntu-20.04.3-live-server-amd64.iso which is the 64-bit version of Ubuntu
90 20.04.3 LTS (Focal Fossa).

91 For this test, Windows machines will not be used. Windows-compiled libraries are not permitted. All software must run
92 under Ubuntu 20.04.3.

93 NIST will link the provided library file(s) to our C++ language test drivers. Participants are required to provide their library
94 in a format that is dynamically-linkable using the C++17 compiler g++ version 9.3.0.

95 A typical link line might be

96 ```
g++ -I. -Wall -m64 -o frvt11 frvt11.cpp -L. -lfrvt_11_acme_007
```

97 The Standard C++ library should be used for development.  Header files containing API function prototypes will be
98 provided separately for each FRVT track and documented in each of the corresponding API documents.

99 The header files will be made available to implementers at https://github.com/usnistgov/frvt.  All algorithm submissions
100 will be compiled against the officially published header files – developers should not alter the header files when compiling
101 and building their libraries.

102 All compilation and testing will be performed on x86_64 platforms.  Thus, participants are strongly advised to verify
103 library-level compatibility with g++ (on an equivalent platform) prior to submitting their software to NIST to avoid linkage
104 problems later on (e.g. symbol name and calling convention mismatches, incorrect binary file formats, etc.).

105

---

[1] http://man7.org/linux/man-pages/man2/fork.2.html

[2] cat /proc/cpuinfo returns fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf
eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe
popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch ida arat epb pln pts dtherm tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm rdseed adx smap xsaveopt cqm_llc
cqm_occup_llc

[3] cat /proc/cpuinfo returns fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse
sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc aperfmperf
eagerfpu pni pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe
popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch ida arat epb pln pts dtherm tpr_shadow vnmi
flexpriority ept vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx avx512f rdseed adx smap clflushopt
avx512cd xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc

## 106    6.  Software and Documentation

### 107    6.1.    Library and Platform Requirements

108    Participants shall provide NIST with binary code only (i.e. no source code).  The implementation should be submitted in
109    the form of a dynamically-linked library file.

110    The core library shall be named according to "Implementation Library Filename" documented in each API document.  The
111    library name will generally follow the convention: *libfrvt_<track>_<provider>_<sequence>.so*.  Additional supplemental
112    libraries may be submitted that support this "core" library file (i.e. the "core" library file may have dependencies
113    implemented in these other libraries).  Supplemental libraries may have any name, but the "core" library must be
114    dependent on supplemental libraries in order to be linked correctly. The **only** library that will be explicitly linked to the
115    test driver is the "core" library.

116    Intel Integrated Performance Primitives (IPP) ® libraries are permitted if they are delivered as a part of the developer-
117    supplied library package. It is the provider's responsibility to establish proper licensing of all libraries.  The use of IPP
118    libraries shall not prevent running on CPUs that do not support IPP.  Please take note that some IPP functions are
119    multithreaded and threaded implementations are prohibited.

120    Developers may obviously use common deep learning frameworks (e.g. Caffe, TensorFlow, etc.) and should submit those
121    dependencies as supplemental libraries.  NIST has successfully received and run implementations leveraging such deep
122    learning frameworks in other evaluations with no issues.

123    Do not include any standard libraries (e.g., libc.so, libgcc.so, etc.) that come with the operating system and/or compilation
124    environment in your submission.  The NIST test harness will handle all image I/O, so do not include JPEG or PNG libraries
125    (i.e., libjpg.so, libpng.so) in your submission.  If you need to include those libraries for other reasons, please contact NIST
126    prior to your submission.  NIST will report the size of the supplied libraries.

127    **Important:** Public results will be attributed with the provider name and the 3-digit sequence number in the submitted
128    library name.

### 129    6.2.    Configuration and developer-defined data

130    The implementation under test may be supplied with configuration files and supporting data files.  NIST will report the
131    size of the supplied configuration files.

### 132    6.3.    A Note on Training

133    NIST and the FRVT program do not train biometric recognition algorithms.  We do not provide training data to software,
134    and software is prohibited from adapting to any data we pass to the algorithms.  Training of biometric recognition
135    algorithms is not a turn-key operation; instead it is typically an extended process involving researchers curating suitable
136    training sets, establishing architectures and hyperparameters, and running trials over days or weeks, and then evaluating
137    the output.  The result of such a process, which is often iterative, is usually a "trained model" i.e. static data and
138    parameters that can be saved and provided to NIST as an integral part of the black-box recognition engine.  NIST does not
139    support training, because our tests seek to mimic operational reality and, there, algorithms are almost always shipped
140    and used "as is" without any training or adaptation to customer data.   The representation of the biometric characteristic,
141    as described by the "model", is fixed until the software is upgraded.

### 142    6.4.    Submission folder hierarchy

143    Participant submissions shall contain the following folders at the top level

144    —   lib/ - contains all participant-supplied software libraries

145    —   config/ - contains all configuration and developer-defined data, e.g., trained models

146    —   doc/ - contains version.txt, which documents versioning information for the submitted software and any other
147        participant-provided documentation regarding the submission

148    —   validation/ - contains validation output

### 6.5.    Installation and Usage

The implementation shall be installable using simple file copy methods. It shall not require the use of a separate installation program and shall be executable on any number of machines without requiring additional machine-specific license control procedures or activation.  The implementation shall not use nor enforce any usage controls or limits based on licenses, number of executions, presence of temporary files, etc.  The implementation shall remain operable for at least six months from the submission date.

### 6.6.    Documentation

Participants shall provide documentation of additional functionality or behavior beyond that specified here.

### 6.7.    Modes of operation

Implementations shall not require NIST to switch "modes" of operation or algorithm parameters. For example, the use of two different feature extractors must either operate automatically or be split across two separate library submissions.

## 7.  Runtime behavior

### 7.1.    Interactive behavior, stdout, logging

The implementation will be tested in non-interactive "batch" mode (i.e. without terminal support). Thus, the submitted library shall:

— Not use any interactive functions such as graphical user interface (GUI) calls, or any other calls which require terminal interaction e.g. reads from "standard input".

— Run quietly, i.e. it should not write messages to "standard error" and shall not write to "standard output".

— Only if requested by NIST for debugging, include a logging facility in which debugging messages are written to a log file whose name includes the provider and library identifiers and the process PID.

### 7.2.    Exception Handling

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

### 7.3.    External communication

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

### 7.4.    Stateless behavior

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

### 7.5.    Single-thread Requirement/Parallelization

Implementations must run in single-threaded mode, because NIST will parallelize the test by dividing the workload across many cores and many machines.  Implementations must ensure that there are no issues with their software being parallelized via the `fork()` function. Developers should take caution with checking threading when using third-party frameworks (e.g., TensorFlow, MXNet, etc.).

189 ## 8. Data structures supporting the API

190 The common data structures used to support the C++ API functions for the various FRVT tasks are published on GitHub at
191 https://github.com/usnistgov/frvt/blob/master/common/src/include/frvt_structs.h.  The actual C++ API function
192 prototypes themselves are documented separately for each test and are available on the website for each track.