Face Recognition Vendor Test
Ongoing

Face Recognition AE (Age Estimation)

Application Programming Interface (API)

VERSION 0.8

DRAFT FOR PUBLIC COMMENT

EMAIL COMMENTS TO FRVT@NIST.GOV BY AUGUST 03, 2023

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

Contact via frvt@nist.gov

July 03, 2023

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

Table of Contents

List of Tables

1. FRVT AE

1.1. Scope

Facial age verification has recently been mandated in legislation in a number of jurisdictions. These laws are typically intended to protect minors from various harms by verifying that the individual is above a certain age. Less commonly some applications extend benefits to groups below a certain age. Further use-cases seek only to determine actual age. The mechanism for estimating age is usually not specified in legislation. Face analysis using software is one approach, and is attractive when a photograph is available or can be captured.

This track of the Face Recognition Vendor Test - FRVT Age Estimation (AE) – is an ongoing evaluation of software algorithms that inspect photos and videos of a face to produce an age estimate. The output of this track is a set of reports on the accuracy and computational efficiency of algorithms. As with other FRVT evaluations, this track of FRVT is open to a worldwide community of developers.

This document establishes a concept of operations and an application programming interface (API) for evaluation of facial age estimation implementations submitted to NIST. Separate API documents are published for other FRVT tracks.

This track of FRVT does not involve identity determination or recognition.

1.2. General FRVT Evaluation Specifications

General and common information shared between all Ongoing FRVT tracks are documented in the FRVT General Evaluation Specifications document - https://pages.nist.gov/frvt/api/FRVT_common.pdf. This includes rules for participation, hardware and operating system environment, software requirements, reporting, and common data structures that support the APIs.

1.3. Reporting

For all algorithms that complete the evaluation

- NIST will publish on its website the name of the developer's organization, the algorithm identifiers, and the performance results with attribution to the developer.
- NIST may provide other performance results to the participating organization.
- NIST may additionally report and share results with partner government agencies and interested parties, and in workshops, conferences, conference papers, presentations, and technical reports.

Important: The developer's name will be published alongside the results.

Results will be machine generated (i.e., scripted) and will include timing, accuracy and other performance results. These will be provided alongside results from other implementations. Results will be expanded and modified as additional implementations are tested, as new image datasets become available, and as analyses are implemented. Results may be updated, extended and regenerated after an initial publication for example whenever additional implementations complete testing, or when new analyses are added.

1.4. Time limits

The elemental functions of the implementations shall execute under the time constraints of Table 1. These time limits apply to the function call invocations defined in section 4. The median duration will be compared with the time limit meaning that the median duration should be less than that given in the Table.

The time limits apply per image.

Table 1 – Processing time limits in milliseconds for a single 640 x 480 image of one face

Function	1:1 verification
estimateAge() Table 3 verifyAge()	1000 (1 core)
estimageAge() Table 4 estimateAgeDifference()	2000 (1 core)

1.5. Accuracy Metrics

This test will evaluate algorithmic ability to estimate the age of a person in an image or a video. NIST will measure age estimation accuracy across various imaging conditions and various demographic groups. NIST will compute and report:

- Statistics of the difference between the actual and estimated ages, e.g., Mean Absolute Error
- False Positive Rate (FPR) – The proportion of samples for which the estimated age is at or above an age threshold, but the actual age is not.
- False Negative Rate (FNR) – The proportion of samples for which the estimated age is at or below an age threshold, but the actual age is not.
- Statements of the estimated age which, if used as a threshold, would give a zero error rate on determining whether the subject is below all age of interest.

We will report failure to process counts.

2. Data structures supporting the API

The data structures supporting this API are documented in the FRVT - General Evaluation Specifications document, with corresponding header file named *frvt_structs.h* published at https://github.com/usnistgov/frvt.

3. Implementation Library Filename

The core library shall be named as libfrvt_ae_<***provider***>_<***sequence***>.so, with

- provider: single word, non-infringing name of the main provider. Example: acme
- sequence: a three digit decimal identifier to start at 000 and incremented by 1 every time a library is sent to NIST. Example: 007

Example core library names: libfrvt_ae_acme_000.so, libfrvt_ae_mycompany_006.so.

Important: Public results will be attributed with the provider's name and the 3-digit sequence number.

4. API Specification

FRVT AE participants shall implement the relevant C++ prototyped interfaces in Section 4.3 . C++ was chosen in order to make use of some object-oriented features.

4.1. Header File

The prototypes from this document will be written to a file named **frvt_ae.h** and will be available to implementers at https://github.com/usnistgov/frvt.

4.2. Namespace

All supporting data structures will be declared in the `FRVT` namespace. All API interfaces/function calls for this track will be declared in the `FRVT_AE` namespace.

4.3. API

4.3.1. Interface

The software under test must implement the interface `Interface` by subclassing this class and implementing each method specified therein.

	C++ code fragment	Remarks
1.	`class Interface`	
2.	`{` `public:`	
3.	`virtual ReturnStatus initialize(` `const std::string &configDir ) = 0;`	Supports algorithm initialization

4.	`virtual ReturnStatus estimateAge(`
	`const Media &face,`
	`double &age) = 0;`
		Age (in years) estimation, given media of a single still image, or a sequence of video frames.

5.	`virtual ReturnStatus estimateAge(`
	`const Media &faceOne,`
	`const double &ageOne,`
	`const Media &faceTwo,`
	`double &ageTwo) = 0;`
		Estimate the age (in years) from media input of a single still image, or a sequence of video frames of the person in faceTwo.
		faceOne and ageOne are provided as inputs for additional information of the same person in faceTwo.

6.	`virtual ReturnStatus estimateAgeDifference(`
	`const Media &faceOne,`
	`const Media &faceTwo,`
	`double &ageDifference) = 0;`
		Absolute value of an estimated age difference (in years), given two faces of the same person.

7.	`virtual ReturnStatus verifyAge(`
	`const Media &face,`
	`const double &ageThreshold,`
	`bool &isAboveThreshold) = 0;`
		A binary decision on whether the face in the image, or sequence of video frames is compliant with the provided age threshold.

8.	`static std::shared_ptr<Interface> getImplementation();`	Factory method to return a managed pointer to the `Interface` object. This function is implemented by the submitted library and must return a managed pointer to the `Interface` object.
8.	`};`	

There is one class (static) method declared in `Interface.getImplementation()` which must also be implemented by the implementation. This method returns a shared pointer to the object of the interface type, an instantiation of the implementation class. A typical implementation of this method is also shown below as an example.

	C++ code fragment	Remarks

	`#include "frvt_ae.h"`
	
	`using namespace FRVT_AE;`
	
	`NullImpl::NullImpl () { }`
	
	`NullImpl::~NullImpl () { }`
	
	`std::shared_ptr<Interface>`
	`Interface::getImplementation()`
	`{`
	`return std::make_shared<NullImpl>();`
	`}`
	
	`// Other implemented functions`	

4.3.2. Initialization

The NIST test harness will call the initialization function in Table 2 before calling any of the quality assessment functions of this API. This function will be called BEFORE any calls to `fork()`[1] are made.

Table 2 – Initialization

Prototype	ReturnStatus initialize(const string &configDir);	 Input
Description	This function initializes the implementation under test. It will be called by the NIST application before any calls the quality assessment functions of this API. The implementation under test should set all parameters. This function will be called N=1 times by the NIST application, prior to parallelizing M >= 1 calls to any other functions via `fork()`.	
Input	configDir	A read-only directory containing any developer-supplied configuration parameters or run-time data files. The name of this directory is assigned by NIST, not hardwired by the provider. The names of the files in this directory are hardwired in the implementation and are unrestricted.
Output	none	
Return Value	See General Evaluation Specifications document for all valid return code values.	

4.3.3. Estimate age from an input media of a person

Table 3 – Estimate age from an input media

Prototypes	ReturnStatus estimateAge(const Media &face, double &age);	 Input Output
Description	This function estimates the age of a person from media of a still image, or sequence of video frames of exactly one person. An estimated age should be returned in years.	
Input	face	Input media of a single still image or a sequence of video frames of one person.
Output	age	Indication of the age (in years) of the person.
Return Value	See General Evaluation Specifications document for all valid return code values.	

4.3.4. Estimate age from an input media given ground truth from separate media

Table 4 – Estimate age given ground truth from separate media

Prototypes	ReturnStatus estimateAge(const Media &faceOne, const double &ageOne, const Media &faceTwo, double &ageTwo);	 Input Input Input Output
Description	This function estimates the age of the person in faceTwo. This function allows implementation to either estimate age as in section 4.3.3, or to exploit additional information provided with faceOne image. Developer should not assume that faceTwo is collected at a later date than faceOne.	
Input	faceOne	Input media of a single still image or a sequence of video frames of one person.
Input	ageOne	Age of the person in faceOne at the time it was taken.
Input	faceTwo	Input media of a single still image or a sequence of video frames of the same person in faceOne.
Output	ageTwo	A value indicating the estimated age of the face in faceTwo.
Return Value	See General Evaluation Specifications document for all valid return code values.	

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

4.3.5. Estimate age difference between two images of one person

NIST specifically invites comments on whether this function is needed. NIST would like to avoid the additional computational cost unless there is an operational use-case that cannot be met by making two calls to Table 3 function.

Table 5 – Estimate age difference between two input media

Prototypes	ReturnStatus estimateAgeDifference(		
	const Media &faceOne,		Input
	const Media &faceTwo,		Input
	double &ageDifference);		Output
Description	This function estimates the age difference between two face images of the same person.		
Input	faceOne	Input media of a single still image or a sequence of video frames of one person.	
Input	faceTwo	Input media of a single still image or a sequence of video frames of the same person in faceOne.	
Output	ageDifference	A value indicating the age difference (in years). Positive values indicate faceTwo image was collected later than faceOne. Negative values indicate the opposite.	
Return Value	See General Evaluation Specifications document for all valid return code values.		

4.3.6. Age Compliance

Table 6 – Verification that subject in input media is above an age threshold

Prototypes	ReturnStatus verifyAge(		
	const Media &face,		Input
	const double &ageThreshold,		Input
	bool &isAboveThreshold);		Input
Description	This function returns a binary decision on whether the face in the image is above an age threshold. This function prototype allows an implementation to invoke specialized processing for certain age groups. We anticipate calling this function with ageThreshold values in {12, 18, 21, and 70}. We may use other values also.		
Input	face	Input media of a single still image or a sequence of video frames of one person.	
Input	ageThreshold	Input age of interest.	
Output	isAboveThreshold	True if the estimated age of the input media is above the age threshold; False otherwise.	
Return Value	See General Evaluation Specifications document for all valid return code values.		