Iris Exchange (IREX) 10: Ongoing Evaluation of Iris Recognition Identification Track

Concept, Evaluation Plan, and API Overview

Version 0.3

George W. Quinn
Information Technology Laboratory, NIST

Questions and Comments to irex@nist.gov

The latest version of this document is available at:
https://www.nist.gov/programs-projects/irex-10-identification-track

September 17, 2019

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

Release Notes

The latest version of this document can be downloaded from
https://www.nist.gov/programs-projects/irex-10-identification-track.

Inquiries and comments may be submitted to irex@nist.gov. Subscribe to the IREX mailing list by sending an email to irex+subscribe@list.nist.gov.

The submission procedure, implementation, and runtime requirements for IREX 10 are similar to those of prior IREX evaluations. However, the API has undergone significant changes.

Timeline

July 26th, 2019	Version 0.1 of this document is open for comment.
Aug 6th, 2019	Version 0.2 of this document is released for comment.
Aug 26th, 2019	Version 0.3 of this document is released for comment.
Sep 17th, 2019	**CONOPS & API document frozen.**
Sep 30th, 2019 onwards	NIST will be accepting applications.

Contents

1 Overview **1**
- 1.1 Introduction . . . 1
- 1.2 Publication of Results . . . 1
- 1.3 Performance Metrics . . . 1

2 Participation Requirements **2**
- 2.1 Who Can Participate? . . . 2
- 2.2 Implementation Requirements . . . 2
  - 2.2.1 Linking . . . 2
  - 2.2.2 Installation . . . 3
  - 2.2.3 Runtime . . . 3
- 2.3 Submission Procedure . . . 4
- 2.4 Software Validation . . . 5

3 Test Environment **6**
- 3.1 Hardware . . . 6
- 3.2 Software . . . 6
- 3.3 Iris Datasets . . . 6

4 API Overview **6**

5 References **8**

1 Overview

1.1 Introduction

This document establishes a concept of operations (CONOPS) and application programming interface (API) specification for the *Iris Exchange (IREX) 10 Ongoing Evaluation of Iris Recognition, Identification Track*. Administered offline at NIST's Biometrics Research Lab (BRL), developers submit their matching software for testing over sequestered iris data. As an ongoing evaluation, developers may submit at any time (unless otherwise specified on the IREX 10 website[1]).

The *Identification Track* of IREX 10 assesses performance for one-to-many (aka identification mode) applications. Most flagship deployments of iris recognition operate in one-to-many mode, providing services ranging from prison management, border security, expedited processing, distribution of resources, and database de-duplication.

IREX 10 is the latest installment of the IREX umbrella program. IREX was initiated by NIST to support the standardization, development, and deployment of iris-based technology. Although IREX 10 is the first ongoing IREX evaluation, it does not preclude the possibility that NIST will conduct future one-off evaluations as part of IREX. The latest information on IREX can be found on the IREX website[2].

1.2 Publication of Results

NIST will post performance results for each submitted implementation to the IREX 10 website, in periodic NIST interagency reports, and possibly other places as well.

1.3 Performance Metrics

Performance is assessed for one-to-many matching. One-to-many systems are tasked with searching individuals against an enrolled population and returning the searched individual's correct identity if found. The identification step involves producing a list of candidate identities, each with a comparison score quantifying the degree of similarity (or dissimilarity) of the candidate to the searched individual.

Offline software tests such as IREX 10 cannot mimic all aspects of a deployed system. Nevertheless, general performance metrics can provide insight into the capabilities and potential of specific biometric technologies. Below is a list of performance factors that may be assessed in IREX 10:

Matching Accuracy Accuracy will be presented in the form of Detection Error Trade-off (DET) plots [1]. These plots present matching accuracy as a trade-off between two error rates as a decision threshold is adjusted. Examples of DET plots[3] can be found in the first IREX IX report. Other accuracy statistics that may be reported are the *false negative identification rate* (FNIR) at a fixed *false positive identification rate* (FPIR)[4] and purely rank-based statistics such as rank R "hit" and "miss" rates.

IREX 10 is conducted entirely offline without any real-time image acquisition component. As such, it is not possible to prompt for a new sample if the given sample is deemed to be of insufficient quality. Failures to extract useful feature information from iris images will effectively inflate the "miss rate" in accuracy statistics.

Computation Time Timing will be assessed as the elapsed real time (as opposed to CPU time) for core biometric operations (e.g. feature extraction, identification). Timing estimates will be computed on an unloaded machine running a single process. The machine's specifications are described in Section 3. NIST may also explore the relationship between search time and the size of the enrollment database, with an eye on whether the relationship is sub-linear.

[1] IREX 10 Hompage: https://www.nist.gov/itl/iad/image-group/ongoing-irex

[2] IREX Homepage: http://www.nist.gov/itl/iad/ig/irex.cfm

[3] See page 18 of the first IREX IX report: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8207.pdf#page=18.

[4] Definitions of FNIR and FPIR are provided on page 11 of the first IREX IX Report: https://nvlpubs.nist.gov/nistpubs/ir/2018/NIST.IR.8207.pdf#page=11.

Accuracy-speed Trade-off NIST may investigate whether a collective accuracy-speed trade-off exists for submitted implementations. Accuracy-speed trade-off refers to the phenomenon where more accurate implementations tend take longer to complete their operations.

Template Sizes NIST will report the size of the proprietary matching templates generated by the implementation.

Runtime Memory Usage NIST may monitor runtime memory usage during identification and/or feature extraction.

2 Participation Requirements

2.1 Who Can Participate?

Participation is open to any commercial organization or academic institution that has the ability to implement an iris matching algorithm. There is no charge and participation is open worldwide.

A high-level description of the process is as follows:

1. Mail the completed Participation Agreement[5] to NIST.
2. Wrap your matching software in a C++ software library that adheres to the API specifications and runtime requirements described in Section 4 of this document.
3. Submit your implementation to NIST, following the cryptographic protection procedures described in Section 2.3.
4. NIST will validate your submission to ensure its correct operation (Section 2.4).
5. NIST will assess the performance of your implementation and post the results to the IREX 10 webpage.

Participants only need to complete the Participation Agreement the first time they submit an implementation (or whenever the point of contact changes). Participants are permitted to submit a new implementation every 3 months.

2.2 Implementation Requirements

2.2.1 Linking

Participants shall submit their implementations as pre-compiled and linkable libraries. Both shared and static libraries are permitted. The library name shall be libIREX10.so if shared and libIREX10.a if static. Additional dynamic or shared library files may be submitted that support this core library. Participants shall *not* provide any source code. Header files should not be necessary, but if provided, should not contain any intellectual property or any material that is otherwise proprietary.

NIST will link the submitted library file(s) to our ISO 2011 C++ language test drivers. Participants are required to provide their libraries in a format that is linkable using g++ version 4.8.5. Thus, libraries must export their functions according to that compilers naming convention. The functions that must be exported are defined in "irex.h" (found in the validation package) and described in Section 4 of this document. The software libraries must be 64-bit. Participants may provide customized command-line linking parameters. A typical link line might be:

```
g++ -I. -Wall -m64 -o irex_main irex_main.cpp -L. -lIREX10
```

[5]The participation agreement can be found at:
https://www.nist.gov/programs-projects/irex-10-identification-track

Participants are strongly advised to verify library-level compatibility with g++ (on an equivalent platform) prior to submitting their software to NIST to avoid linkage problems (e.g. symbol name and calling convention mismatches, incorrect binary file formats, etc.). NIST's test machines will have CentOS 7.6 installed, which can be downloaded from nigos.nist.gov[6]

Intel ICC is not available. Access to GPUs is not permitted. Intel Integrated Performance Primitives (IPP) libraries are permitted if they are delivered as part of the developer-supplied library package. It is the provider's responsibility to establish proper licensing of all libraries.

NIST will ignore requests to alter parameters by hand (e.g. modify specific lines in an XML configuration file). Any such adjustments must be submitted as new implementations.

Dependencies on external dynamic/shared libraries such as compiler-specific development environment libraries are discouraged. If absolutely necessary, external libraries must be provided to NIST after receiving prior approval from the test liaison. Image processing libraries such as libpng and NetPbm should not be required since NIST will handle image reading and decompression.

2.2.2 Installation

Installation Must be Simple Installation shall require the simple copying and/or decompression of files followed by a linking operation. There shall be no need for interaction with the participant provided everything goes smoothly. It shall not require an installation program.

No License Requirements or Usage Restrictions The implementation shall allow itself to be executed on any number of machines without the need for machine-specific license control procedures or activation. The implementation shall neither implement nor enforce any usage controls or restrictions based on licenses, number of executions, presence of temporary files, etc. No activation dongles or other hardware shall be required.

Sufficient Documentation Must be Provided Documentation should be provided for all (non-zero) participant-defined error or warning return codes.

Disk-Space Limitations The implementation may use supporting libraries and configuration files. The total size of a submission shall not exceed a gigabyte.

2.2.3 Runtime

Single-threaded Requirement Implementations must run in single-threaded mode.

No writing to Standard Error or Standard Output The implementation will be tested in a non-interactive "batch" mode without terminal support. Thus, the submitted library shall run quietly (i.e. it should not write messages to "standard error" or "standard output". An implementation may write debugging messages to a log file. This log file must be declared in the documentation.

Exception Handling Should be Supported The implementation should support error/exception handling so that, in the case of an unexpected error, a return code is still provided to the calling application. The NIST test harness will gracefully terminate itself if it receives an unexpected return code, as it usually indicates improper operation of the implementation.

No External Communication Implementations running on NIST hosts shall not side-effect the runtime environment in any manner except through the allocation and release of memory. Implementations shall not write any data to an external resource (e.g. a server, connection, or other process). Implementations shall not attempt to read any resource other than those explicitely allowed in this document. If detected, NIST reserves the right to cease evaluation of the software, notify the participant, and document the activity in published reports.

Components Must be Stateless All implementation components shall be "stateless" except as noted elsewhere in this document. This applies to iris detection, feature extraction and matching.

[6] https://nigos.nist.gov/evaluations/CentOS-7-x86_64-Everything-1810.iso

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 reserves the right to cease evaluation of the software, notify the participant, and document the activity in published reports.

Minimum Speed Requirements The implementations shall perform operations within the time constraints specified by Table 1. These time limits apply to the function call invocations defined in Section 4 on a Dell M910 system described in Section 3. Since NIST cannot regulate the maximum runtime per operation, limitations are specified as 90th percentiles (i.e. 90% of all calls to the function shall complete in less time than the specified duration). The limitations assume each template is generated from a single iris image.

Table 1: Time limitations for specific operations.

Operation	Timing Restriction
Creation of a verification template from a single 640x480 pixel image	1000 ms
Creation of an enrollment template from a single 640x480 pixel image	1000 ms
Search duration on a database of one million templates	20 000 ms

Template Size Templates should not require more than 100 KB of persistent storage per iris image. Participants should inform NIST if their implementations require more than 100 KB of persistent storage.

2.3 Submission Procedure

All software, data, and configuration files submitted to NIST must be signed and encrypted. Signing is performed to ensure authenticity of the submission (i.e. that it actually belongs to the participant). Encryption is performed to ensure privacy. Implementations shall be submitted to NIST as encrypted gpg files. Encryption shall follow the procedures described at http://www.nist.gov/itl/iad/ig/encrypt.cfm.

If the encrypted implementation is below 20MB, it can be emailed directly to the IREX 10 Liaison (irex@nist.gov). If the encrypted implementation is above 20MB, it can be provided to NIST as a download from a webserver. NIST shall not be required to register or enroll in any kind of membership before downloading the implementation. Providing NIST with a Google Drive link is recommended. For security reasons, DropBox links are not currently accepted.

NOTE: NIST will not accept any implementations that are not signed and encrypted. NIST accepts no responsibility for anything that occurs as a result of receiving files that are not encrypted with the NIST public key.

All submissions shall adhere to the naming convention described in Table 2.

Table 2: Naming convention for a submitted implementation.

Name Format:	irex10_1N_<participantName>_<increment>.<ext>.gpg			
	irex10_1N	Participant Name	Submission Incrementor	Extension
Description:	Fixed for all submissions.	A short name for the participant (e.g. ”thebes”).	A two-digit zero-indexed identifier that increments anytime a new submission is provided to NIST.	compression type (e.g. ”zip”, ”tar”)
Example:	irex10_1N_thebes_00.tar.gz.gpg			

2.4 Software Validation

Upon receipt, NIST will validate the implementation to ensure its correct operation. The validation process involves running the implementation over a small sample of publicly available data. This test data will be provided to the participant, who must run the implementation in-house and provide NIST with the comparison results. NIST will then verify that the participant's in-house results are consistent with the output produced in the NIST BRL. The validation data along with instructions can be downloaded from the IREX 10 webpage (https://www.nist.gov/itl/iad/image-group/ongoing-irex).

3 Test Environment

3.1 Hardware

Testing is performed offline at a NIST facility on a secure ”air-gapped” network. Hardware specifications for some of the test machines in NIST’s Biometrics Research Laboratory (BRL) are:

- Dell M610 - Dual Intel Xeon X5680 3.3 GHz CPUs (6 cores each).
- Dell M910 - Dual Intel Xeon X7560 2.3 GHz CPUs (8 cores each).
- Dual Intel Xeon E5-2695 3.3 GHz CPUs (14 cores each; 56 logical CPUs total).

3.2 Software

The test machines have CentOS 7.6 installed. An ISO image of the distribution can be downloaded from nigos (https://nigos.nist.gov/evaluations/CentOS-7-x86_64-Everything-1810.iso).

NIST’s test harness relies heavily on the open source Biometric Evaluation Framework (http://github.com/usnistgov/libbiomeval [2]).

3.3 Iris Datasets

Currently, IREX 10 only tests over a single iris dataset, but this is likely to change as additional datasets become available. Documentation on this dataset, referred to as ”OPS IV”, can be found at https://www.nist.gov/itl/iad/image-group/irex-datasets. The dataset is sequested

All test datasets are sequestered in the BRL lab and therefore not accessible to participants. The participants are not allowed to view any of the iris samples and are not provided with any representative sets of iris samples.

Ground Truth Integrity: A hazard with collecting operational data is that ground truth identity labels can be incorrectly assigned due to clerical error. NIST may attempt to correct ground truth errors in its test datasets when possible, and only when doing so will not unfairly bias results in favor of specific implementations.

4 API Overview

The design of this API reflects the following testing objectives:

- Support black-box testing.
- Support distributed processing.
- Support graceful and informative failure recovery.
- Support the ability to collect performance statistics.

Submitted libraries must derive from the C++ base class `Interface` declared in the API header file and implement its five declared functions. Table 3 provides a high-level presentation of how NIST’s test harness will call these functions.

The full API documentation is available at
https://usnistgov.github.io/IREX10/API/class_irex_1_1_interface.html.

To assist developers, a minimal working *stub* (a.k.a. null implementation) is available at
https://github.com/usnistgov/IREX10/tree/1N/src/stub.

Table 3: Basic Program Flow

Task	Function
Enrollment	**initializeTemplateCreation(...)** Allows the implementation to perform initialization procedures prior to template creation. *Provides the implementation with advance knowledge of the number of images that will be enrolled as well as read-only access to a participant-supplied configuration directory.* **createTemplate(...)** Creates an enrollment template from one or more iris images. *The implementation must be able to handle multiple calls to this function from multiple instances of the calling application.*
	createDatabase(...) Constructs an enrollment database from a collection of enrollment templates. *This function allows post-enrollment book-keeping, normalization, and other statistical processing of the templates. The implementation should populate the specified directory with everything necessary to perform searches against the database.* *NOTE: The test harness is not guaranteed to call createEnrollmenDatabase() within the same executable as the above functions.*
Pre-search	**initializeTemplateCreation(...)** Allows the implementation to perform initialization procedures prior to template creation. *The implementation is allowed read-only access to the enrollment directory at this stage.* **createTemplate(...)** Creates a search template from one or more iris images. *The implementation must be able to handle multiple calls to this function from multiple instances of the calling application.*
Search	**initializeIdentification(...)** Prepares the implementation for searches against the enrollment database. *The implementation is allowed to read data (e.g. templates) from the enrollment database directory and load them into memory.* **identify(...)** Searches a template against the enrollment database and returns an ordered list of candidates.

5 References

[1] Alvin Martin, George Doddington, Terri Kamm, Mark Ordowski, and Markv Przybocki. The DET curve in assessment of detection task performance. Technical report, DTIC Document, 1997. 1

[2] Gregory Fiumara, Wayne Salamon, and Craig Watson. Towards Repeatable, Reproducible, and Efficient Biometric Technology Evaluations. In *Biometrics Theory, Applications and Systems (BTAS), 2015 IEEE 7th International Conference on*, pages 1–8, Sept 2015. 6