IREX

An Evaluation-based Program for the Development of Compact

Interoperable ISO/IEC 19794-6 Standardized Iris Images

Iris Exchange (IREX) Evaluation 2008

Concept, Evaluation Plan and API

Patrick Grother

NIST

September 10, 2008

23.	Image length	4B	$M > 0$	Length of image data. For uncompressed 8 bit greyscale data this is W x H x 1.
24.	Image number	2 B	1	This field is an index starting at 1
25.	Quality	1 B	$0 \le Q \le 100$, 255	This value shall be computed and stored here by the SDK
26.	Quality algorithm vendor ID	2B		This value is optional in IREX. IBIA code may be used - NIST would build register.
27.	Quality algorithm ID	2B		The value is optional in IREX
28.	Rotation angle of eye	2 B	0 ... 0xFFFE 0xFFFF =ROT_ANGLE_UNDEF	= (unsigned short) round(65536*angle/360) modulo 65536, where angle is measured counter-clockwise in degrees
29.	Rotation uncertainty	2 B	0 ... 0xFFFE 0xFFFF =ROT_UNCERTAIN_UNDEF	= (unsigned short) round(65536*uncertainty/180) where $0 \le$ uncertainty < 180 is measured in degrees and is the absolute value of maximum error
	Fields supporting ROI-masked images			
30.	Mask value for upper eyelid	1B		These three fields are populated if Kind = 7. Otherwise all set to zero. See definitions and rules in section 6.4. Beyond IREX 08 the depth may need to be >= 8 bits.
31.	Mask value for lower eyelid	1B		
32.	Mask value for sclera	1B		
	Fields supporting UNSEG polar images			
33.	Number of samples radially	2B	$0 < NR$	These six fields shall be populated if Kind = 16 or Kind = 48 (see line 3). Otherwise all set to 0. NR is the number of samples along a spoke NC is the number of spokes around the iris The polar image data on the last line of this table has height NR, width NC. Coordinate system is zero oriented with (0,0) at the top left corner. The inner and outer circle centers are concentric.
34.	Number of samples circumferentially	2B	$0 < NC$	
35.	X coordinate of inner + outer circle centers	2 B	$0 \le x < W$	
36.	Y coordinate of inner + outer circle centers	2 B	$0 \le y < H$	
37.	Inner circle radius	2 B	$0 < r$	
38.	Outer circle radius	2 B	$0 < r$	
	Fields supporting all images			
39.	X coord of the center of the ellipse approximating the pupil boundary	2 B		Population of these six fields is allowed but not required for all "Kinds". If an implementation elects not to compute the ellipse then it shall assign X = 0xFFFF on line 39. Consumers of this data shall ignore these fields if X = 0xFFFF i.e. value on line 39 is out of bounds with respect to the width on line 16.
40.	Y coord of the center of the ellipse approximating the pupil boundary	2 B		
41.	X coord of the intersection pt. of the semi-major axis with the ellipse approximating the pupil	2 B		
42.	Y coord of the intersection pt. of the semi-major axis with the ellipse approximating the pupil	2 B		
43.	X coord of the intersection pt. of the semi-minor axis with the ellipse approximating the pupil	2 B		
44.	Y coord of the intersection pt. of the semi-minor axis with the ellipse approximating the pupil	2 B		
45.	X coord of the center of the ellipse approximating the iris boundary	2 B		Population of these six fields is allowed but not required for all "Kinds". If an implementation elects not to compute the ellipse then it shall assign X = 0xFFFF on line 45. Consumers of this data shall ignore these fields if X = 0xFFFF i.e. value on line 45 is out of bounds with respect to the width on line 16. Continued next page ...
46.	Y coord of the center of the ellipse approximating the iris boundary	2 B		
47.	X coord of the intersection pt. of the semi-major axis with the ellipse approximating the iris	2 B		

48.	Y coord of the intersection pt. of the semi-major axis with the ellipse approximating the iris	2 B		
49.	X coord of the intersection pt. of the semi-minor axis with the ellipse approximating the iris	2 B		
50.	Y coord of the intersection pt. of the semi-minor axis with the ellipse approximating the iris	2 B		
51.	Freeman code length for pupil-iris boundary	2 B	NP bytes	See section 6.7. These two blocks are allowed but not required for all "Kinds". If NP = 0, the sclera-iris FCC length value follows immediately. If NS = 0, the image follows immediately. Aug28 2008: If the number of FCC elements in Table 5 is K the length in bytes will be: N = 6 + 3K/8 if 3K is divisible by 8; N = 6 + 3K/8 + 1 otherwise
52.	Freeman code data for pupil-iris boundary	NP		
53.	Freeman code length for sclera-iris boundary	2 B	NS bytes	
54.	Freeman code data for sclera-iris boundary	NS		
55.	Image data	M		Length M encoded on line 23, in bytes

1

7. PC-based API specification

7.1. Overview

This section describes the IREX API. All SDK's submitted to IREX 08 shall implement the functions below here as required by the classes of participation listed in Table 4.

7.2. Testing interface

7.2.1. Requirement

IREX participants shall submit an SDK which presents the "C" prototyped interface given in the following subsections.

7.2.2. Sensor identifiers

IREX will use images from:

- a large corpus collected using the LG 3000.
- a larger corpus collected using the Securimetrics PIER camera.
- the smaller sequestered ICE 06 corpus of LG 2200 images

NIST is actively seeking to extend this to include other sources - please see NIST's call for images on Page 2. To support interoperable i.e. cross-sensor matching, the SDK will be told the sensor the two byte unsigned integer values in Table 7.

Table 7 – Sensor identifiers

#	Sensor Manufacturer and Model	Identifier
1	LG 2200	0x2A16
2	LG 3000	0x2A1E
3	LG 4000	0x2A26
4	Securimetrics PIER	0x1A03
5	Unknown or unspecified	0x0000

Presence on this table indicates NIST's intention to use images captured by these devices. NIST will revise this table as other data becomes available.

7.2.3. Geometric, photometric or other alterations to images

It is at the discretion of the provider whether to alter the input images during the Table 6 record preparation. If a vendor believed for instance that contrast enhancement would produce a more easy-to-recognize image then the implementation is permitted to do this. A more important example is described in Section 2 of [CAM07,CAM08].

> For those images in which the iris was partly outside of the original image frame, the missing pixels were replaced with black ones. For those in which the algorithms detected that the gaze was directed away from the camera, as gauged by projective deformation of the eye shape, a corrective affine transformation was automatically applied which effectively “rotated” the eye in its socket back into orthographic perspective on-axis with the camera.

Such steps are allowed and are likely to allow downstream feature extractors and matchers to give better performance. NIST takes no position on whether these or other operations should be applied. NIST does however prohibit the application of compression and recommends against any action which would blur the image. Note that vendors might profitably implement local image processing steps (i.e. not requiring the entire original image) in the front-end of the feature extraction routines.

7.2.4. Conversion of raw rectilinear imagery to cropped rectilinear

To assess viability of the proposed standard crop-only format, participating submissions to IREX 08 shall convert a raw raster iris image into a cropped raster iris image, and write the result as a Table 6 instance of Kind = 3. To do this the implementation will need to find the iris center and crop symmetrically around it. The implementation shall not compress the image data.

June 25, 2008. The implementation shall not rotate the iris: Instead if rotation is detected it shall be recorded in the appropriate fields of Table 6, i.e. lines 28-29. Template generators and/or matchers should heed such values.

The provided SDK shall implement the API call specified in Table 8.

Table 8 – IREX API for preparation of cropped rectilinear records

<table>
<tr><td>Prototype</td><td colspan="2">INT32 convert_raster_to_cropped_rectilinear(
const BYTE *uncompressed_raster_data,
const UINT16 image_width,
const UINT16 image_height,
const BYTE horz_orientation,
const BYTE vert_orientation,
const BYTE scan_type,
const BYTE which_eye, // August 28, 2008: New parameter
const BYTE image_format,
const BYTE intensity_depth,
const UINT16 nist_encoded_device_id,
INT16 *bbox_topleft_x, // June 25, 08: These two parameters are new and are introduced to allow NIST
INT16 *bbox_topleft_y, // to survey over crop sizes. July 28, 2008: They're are SIGNED (x,y) so can be negative.
const UINT32 allocated _bytes, // August 28, 2008: New parameter added for safety.
BYTE * c_rectilinear_image);</td></tr>
<tr><td>Description</td><td colspan="2">This function takes a raw input image and outputs a corresponding cropped image.
The memory for the output structure is allocated before the call i.e. the implementation shall not allocate memory for the result. The function returns either success (0) or failure (non-zero). Failure indicates a failure to convert the image. The result will nevertheless be a conformant instance with zero irides and zero images (0 on Line 22 of Table 6)</td></tr>
<tr><td rowspan="11">Input Parameters</td><td>uncompressed_raster_data</td><td>The uncompressed raw image used for template creation.</td></tr>
<tr><td>image_width</td><td>The number of pixels indicating the width of the image.</td></tr>
<tr><td>image_height</td><td>The number of pixels indicating the height of the image.</td></tr>
<tr><td>horz_orientation</td><td rowspan="2">NIST anticipates setting these values to ORIENTATION_BASE, per [STD05, 6.5.4].</td></tr>
<tr><td>vert_orientation</td></tr>
<tr><td>scan_type</td><td>Progressive or interlaced. Values per the standard.</td></tr>
<tr><td>which_eye</td><td>EYE_UNDEF = 0 (0x00)
EYE_RIGHT = 1 (0x01)
EYE_LEFT = 2 (0x02) These are the values used in ISO/IEC 19794-6:2005.</td></tr>
<tr><td>image_format</td><td rowspan="2">NIST anticipates using only unprocessed uncompressed 8 bit grayscale data, so the image format will be 0x0002, and the intensity depth will be 8, both per [STD05].</td></tr>
<tr><td>intensity_depth</td></tr>
<tr><td>nist_encoded_device_id</td><td>A two byte unsigned integer value from Table 7</td></tr>
<tr><td>allocated_bytes</td><td>Number of bytes NIST allocated for the output image record</td></tr>
<tr><td rowspan="3">Output Parameters</td><td>c_rectilinear_image</td><td>The output record, per Table 6 with Kind = 3</td></tr>
<tr><td>bbox_topleft_x</td><td>X coordinate in the original image from which the crop was prepared</td></tr>
<tr><td>bbox_topleft_y</td><td>Y coordinate in the original image from which the crop was prepared</td></tr>
<tr><td rowspan="6">Return Values</td><td>0</td><td>Success</td></tr>
<tr><td>2</td><td>Elective refusal to generate this Kind of output record</td></tr>
<tr><td>4</td><td>Involuntary failure to make output record - e.g. could not find iris-sclera boundary.</td></tr>
<tr><td>6</td><td>Elective refusal to produce an output record - e.g. on quality grounds.</td></tr>
<tr><td>8</td><td>Cannot parse input data (i.e. assertion that input record is non-conformant)</td></tr>
<tr><td>Other</td><td>Vendor-defined</td></tr>
</table>

The number of times a non-zero error codes is returned will be counted, reported and appropriately factored into analyses.

7.2.5. Conversion of raw rectilinear imagery to cropped ROI-masked rectilinear

To assess viability of the proposed standard's cropped-and-ROI-masked format, participating submissions to IREX 08 shall convert a raw raster iris image into a cropped raster iris image, replace eyelids and sclera with fixed pixel values, and write the result as a Table 6 instance of Kind = 7. To do this the implementation will need to find the iris center and crop symmetrically around it, and find the eyelids and iris-sclera boundaries. The implementation shall not compress the output image data.

The function shall be implemented with the API call specified in Table 9.

Table 9 – IREX API for preparation of cropped-and-ROI-masked records

Prototype	INT32 convert_raster_to_cropped_and_masked_rectilinear(const BYTE *uncompressed_raster_data, const UINT16 image_width, const UINT16 image_height, const BYTE horz_orientation, const BYTE vert_orientation, const BYTE scan_type, const BYTE which_eye, // New August 28, 2008. const BYTE image_format, const BYTE intensity_depth, const UINT16 nist_encoded_device_id, const UINT32 allocated _bytes, // August 28, 2008: New parameter added for safety. BYTE * cm_rectilinear_image);	
Description	This function takes a raw input image and outputs the corresponding cropped and ROI-masked image. The memory for the output structure is allocated before the call i.e. the implementation shall not allocate memory for the result. The function returns either success (0) or failure (non-zero). Failure indicates a failure to convert the image. The result will nevertheless be a conformant instance with zero irides and zero images (0 on Line 22 of Table 6)	
Input Parameters	uncompressed_raster_data	The uncompressed raw image used for template creation.
	image_width	The number of pixels indicating the width of the image.
	image_height	The number of pixels indicating the height of the image.
	horz_orientation	NIST anticipates setting these values to ORIENTATION_BASE, per [STD05, 6.5.4].
	vert_orientation	
	scan_type	Progressive or interlaced. Values per the standard.
	which_eye	EYE_UNDEF = 0 (0x00) EYE_RIGHT = 1 (0x01) EYE_LEFT = 2 (0x02) These are the values used in ISO/IEC 19794-6:2005.
	image_format	NIST anticipates using only unprocessed uncompressed 8 bit grayscale data, so the image format will be 0x0002, and the intensity depth will be 8, both per [STD05].
	intensity_depth	
	nist_encoded_device_id	A two byte unsigned integer value from Table 7
	allocated_bytes	Number of bytes NIST allocated for the output image record
Output Parameters	cm_rectilinear_image	The output cropped and ROI-masked image, per Table 6 with Kind = 7
Return Values	0	Success
	2	Elective refusal to generate this Kind of output record
	4	Involuntary failure to make output record - e.g. could not find iris-sclera boundary.
	6	Elective refusal to produce an output record - e.g. on quality grounds.
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined

The number of times a non-zero error code is returned will be counted, reported and appropriately factored into analyses.

7.2.6. Conversion of raw rectilinear imagery to unsegmented polar

To assess viability of the proposed standard's unsegmented polar format, participating submissions to IREX 08 shall convert a raw raster iris image into an unsegmented polar image and write the result as a Table 6 instance of Kind = 16. To do this the implementation will need to find concentric circles in the pupil and outside the iris, and to execute the forward polar transformation (using NIST will provide reference polar transformation code). The implementation shall not compress the output image data.

The function shall be implemented with the API call specified in Table 10.

Table 10 – IREX API for creation of unsegmented polar records

Prototype	INT32 convert_raster_to_unsegmented_polar(const BYTE *uncompressed_raster_data, const UINT16 image_width, const UINT16 image_height, const BYTE horz_orientation, const BYTE vert_orientation, const BYTE scan_type, const BYTE which_eye, // New August 28, 2008. const BYTE image_format, const BYTE intensity_depth, const UINT16 nist_encoded_device_id, const UNIT16 num_samples_radially, // Aug 6 2008: If either of these input parameters are 0, the implem- const UINT16 num_samples_circumferentially, // entation should decide suitable values or use published defaults. const UINT32 allocated _bytes, // August 28, 2008: New parameter added for safety. BYTE * unseg_polar_image);	
Description	This function takes raw input image and outputs the corresponding unsegmented polar image. The coordinates of the pupil and iris centers are returned also. The memory for the template is allocated before the call i.e. the implementation shall not allocate memory for the result. The function returns either success (0) or failure (non-zero). Failure indicates a failure to convert the image. The result will nevertheless be a conformant instance with zero irides and zero images (0 on Line 22 of Table 6)	
Input Parameters	uncompressed_raster_data	The uncompressed raw image used for template creation.
	image_width	The number of pixels indicating the width of the input image.
	image_height	The number of pixels indicating the height of the input image.
	horz_orientation	NIST anticipates setting these values to ORIENTATION_BASE, per [STD05, 6.5.4].
	vert_orientation	
	scan_type	Progressive or interlaced. Values per the standard.
	which_eye	EYE_UNDEF = 0 (0x00) EYE_RIGHT = 1 (0x01) EYE_LEFT = 2 (0x02) These are the values used in ISO/IEC 19794-6:2005.
	image_format	NIST anticipates using only unprocessed uncompressed 8 bit grayscale data, so the image format will be 0x0002, and the intensity depth will be 8, both per [STD05].
	intensity_depth	
	nist_encoded_device_id	A two byte unsigned integer value from Table 7
	num_samples_radially	The number of sample along a spoke. The output polar data shall have this height.
	num_samples_circumferentially	The number of "spokes" around the iris. The output polar data shall have this width.
	allocated_bytes	Number of bytes NIST allocated for the output image record
Output Parameters	unseg_polar_image	The output record, per Table 6, with the values on lines 33 - 38 set correctly. The Kind on line 3 and the image data on line 55 shall be set as follows. — EITHER: Set Kind = 16 and insert proper polar format image data. You can use the NIST code available for the interpolation needed to execute the forward polar transform. — OR : Set Kind = 1, copy the input raster to the output record.
Return Value	0	Success
	2	Elective refusal to generate this Kind of output record

	4	Involuntary failure to make output record - e.g. could not find iris-sclera boundary.
	6	Elective refusal to produce an output record - e.g. on quality grounds.
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure.

The number of times a non-zero error code is returned will be counted, reported and appropriately factored into analyses.

7.2.7. Template creation

These functions convert a Table 6 record into an opaque proprietary template. The functions will need to look at the header of the input record to determine the content, particularly the "Kind" value on Line 3. The functions should return the defined error code if it does not support certain Kinds. Sep 7, 2008: Two options are provided - one to convert an image into a generic enrollment or verification template and another to allow two functions one for enrollment and another for verification. This "template role" aspect will be respected in Table 12. It supports matching algorithms that are asymmetric. Your choice of Option 1 or 2 must be indicated in a ".h" header file that you supply.

Table 11 – IREX API template creation

Prototype OPTION #1	INT32 convert_image _to_template(const BYTE *input_record, UINT16 *template_size, BYTE *proprietary_template);	
Prototype OPTION #2 Added Sep 7 2008 Two functions here - both functions must be implemented	INT32 convert_image _to_enrollment_template(const BYTE *input_record, UINT16 *template_size, BYTE *proprietary_template); INT32 convert_image _to_verification_template(const BYTE *input_record, UINT16 *template_size, BYTE *proprietary_template);	
Description	This function takes either a rectilinear image, or an ROI-masked image, or an unsegmented polar image, and outputs a proprietary template. The implementation should inspect the input header to determine which kind of imagery is being provided, per the version number values given in section 6.2. The memory for the output template is allocated 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_record	An instance of Table 6. Implementations must alter their behavior according to the Kind of image. The implementation shall support these values: Kind = 1 Kind = 3 Kind = 7 Kind = 48 The SDK does **not** have to support Kind = 16 because NIST will execute any needed reverse polar transforms using NIST code to make Kind = 48 instances.
Output Parameters	Template_size	The size, in bytes, of the output template
	proprietary_template	The output template. The format is entirely unregulated. June 25,2008: NIST will allocated 8KB before the function is called - if 8KB is not enough email us.
Return Value	0	Success
	2	Elective refusal to process this Kind of input record
	4	Involuntary failure to extract features (e.g. could not find iris in the input-image)
	6	Elective refusal to produce a template (e.g. insufficient iris area)
	8	Cannot parse input data (i.e. assertion that input record is non-conformant)
	Other	Vendor-defined failure

The number of times a non-zero error code is returned will be counted, reported and appropriately factored into analyses. When the error code is "2" this will be noted in the IREX report.

7.2.8. Template comparison

This function compares two proprietary templates and returns a real-valued distance score.

Table 12 – IREX API template matching

Prototype	INT32 match_templates(const BYTE *verification_template, const UINT16 verification_template_size, const BYTE *enrollment_template, const UINT16 enrollment_template_size, double *dissimilarity);	
Description	This function compares two opaque proprietary templates and outputs a non-negative match score. The returned score is a non-negative distance measure. It 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 11), the dissimilarity score shall be -1 and the function return value shall be 2.	
Input Parameters	verification_template	A template from create_template().
	verification_template_size	The size, in bytes, of the input verification template $0 \le N \le 2^{16} - 1$
	enrollment_template	A template from create_template().
	enrollment_template_size	The size, in bytes, of the input enrollment template $0 \le N \le 2^{16} - 1$
Output Parameters	dissimilarity	A dissimilarity score resulting from comparison of the templates, on the range [0,DBL_MAX]. See section 7.2.9.
Return Value	0	Success
	2	Either or both of the input templates were result of failed feature extraction
	Other	Vendor-defined failure

7.2.9. Dissimilarity score

The template comparison function shall return a measure of the dissimilarity between the persons whose iris data is contained in the two templates. So, smaller values indicate more likelihood that the two samples are from the same person. This deviates from many prior NIST tests which have used "larger-is-more-genuine" semantics.

There is no requirement for the scores to be Hamming distances.

There is no requirement for values to obey the metric property.

7.2.10. Implementation identifiers

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

Table 13 – IREX API get_pids function

Prototype	INT32 get_pid(UINT32 *nist_assigned_identifier, char *email_address);	
Description	This function retrieves an identifier that the provider must request from NIST irex@nist.gov, and hardwire into the source code. NIST will assign the identifier that will uniquely identify the supplier and the SDK version number.	
Output Parameters	nist_assigned_identifier	A PID which identifies the SDK under test. The memory for the identifier is allocated by NIST's calling application, and shall not be allocated by the SDK.
	email_address	Point of contact email address as null terminated ASCII string. NIST will allocates at least 64 bytes for this. SDK shall not allocate.
Return Value	0	Success
	Other	Vendor-defined failure

7.3. Software and Documentation

7.3.1. SDK Library and Platform Requirements

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

If dynamic or shared library files are submitted, it is preferred that the API interface specified by this document be implemented in a single "core" library file with the base filename 'libIREX' (for example, 'libIREX.dll' for Windows or 'libIREX.so' for Linux). Additional dynamic or shared library files may be submitted that support this "core" library file (i.e. the "core" library file may have dependencies implemented in these other libraries).

7.3.2. Linking

NIST will link the provided library file(s) to a C language test driver application developed by NIST. The runtime environment shall be either

- RedHat Linux Enterprise 4 or 5 platforms. (PREFERRED)
- The cygwin[10] layer running on a Windows Server 2003 OS.

Both will use GNU's gcc compiler, version 3.3.3. These use libc. The link command might be:

- gcc –o irextest irextest.c -L. –lIREX

Participants are required to provide their library in a format that is linkable using GCC with the NIST test driver, which is compiled with GCC. All compilation and testing will be performed on x86 platforms. Thus, participants are strongly advised to verify library-level compatibility with GCC (on an equivalent platform) prior to submitting their software to NIST to avoid linkage problems later on (e.g. symbol name and calling convention mismatches, incorrect binary file formats, etc.).

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 upon prior approval by the Test Liaison.

7.3.3. Installation and Usage

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

The SDK's usage shall be unlimited. The SDK shall neither implement nor enforce any usage controls or limits based on licenses, execution date/time, number of executions, presence of temporary files, etc.

It is recommended that the SDK be installable using simple file copy methods, and not require the use of a separate installation program. Contact the Test Liaison for prior approval if an installation program is absolutely necessary.

7.3.4. Documentation

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

7.3.5. Modes of operation

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

[10] According to http://www.cygwin.com/ is a Linux-like environment for Windows. It consists of two parts: A DLL (cygwin1.dll) which acts as a Linux API emulation layer providing substantial Linux API functionality; a collection of tools which provide Linux look and feel.

7.3.6. Watermarking of images

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

7.4. Runtime behavior

7.4.1. Speed

The following limits are instituted to constrain NIST's total IREX computational workload. The absolute times are probably less relevant than any relative trends. Deviations above these limits will be allowed but note that timing statistics will be reported.

- The mean template match operation should not exceed 20 milliseconds.
- The mean template creation operation should not exceed 2.5 seconds.
- The mean iris segmentation operation (e.g. polar) should not exceed 2.5 seconds.

The above times assume a vanilla a 2GHz Pentium IV.

7.4.2. Interactive behavior

The SDK 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”.

7.4.3. Error codes and status messages

The SDK will be tested in non-interactive “batch” mod, without terminal support. Thus, the submitted library shall run quietly, i.e. it should not write messages to "standard error" and shall not write to “standard output”.

7.4.4. 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.

7.4.5. 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. 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.

7.4.6. Stateful behavior

All components in this test shall be stateless. This applies to segmentation, 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.

8. References

AN27	NIST Special Publication 500-271: American National Standard for Information Systems — *Data Format for the Interchange of Fingerprint, Facial, & Other Biometric Information – Part 1*. (ANSI/NIST ITL 1-2007). Approved April 20, 2007.
BATH06	S. Rakshit and D. M. Monro, Effects of Sampling and Compression on Human Iris Verification, Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing, Vol. 2, No. II, pp. 337-340, Toulouse, May 2006
BATH07	Soumyadip Rakshit and Donald M. Monro, *An Evaluation of Image Sampling and Compression for Human Iris Recognition*, IEEE Transactions On Information Forensics And Security, Vol. 2, No. 3, September 2007

CAM07	JTC001-SC37-N-2125 — *UK Contribution on the effect of severe image compression on iris recognition performance*. May 29, 2007. This document was submitted as a UK contribution to SC 37 Working Group 3. It is available as: John Daugman and Cathryn Downing, *Effect of severe image compression, on iris recognition performance*. Technical Report No. 685, University of Cambridge, Computer Laboratory, UCAM-CL-TR-685, ISSN 1476-2986, May 07. http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-685.pdf http://isotc.iso.org/livelink/livelink/6461927/JTC001-SC37-N-2125.pdf?func=doc.Fetch&nodeid=6461927
CAM08	Also very similar material to [CAM07] can be found in: John Daugman and Cathryn Downing, *Effect of Severe Image Compression on Iris Recognition Performance*, IEEE Trans on Information Forensics and Security, Vol. 3, No. 1, March 2008.
DHS	J. Mayer-Splain for DHS, *DHS Technical Contribution in Support of Retaining the Iris Polar Image Format.* August 31, 2007. This document is a password protected contribution toward the US position on the revision of 19794-6:2005. http://www.incits.org/tc_home/m1htm/2007xdocs/m1070456.pdf
I379	American National Standard for Information Technology – *iris Image Format for Data Interchange*, ANSI/INCITS 379-2004, http://www.incits.org
IRI07	JTC001-SC37-N-2296 US NB Contribution on Compact Iris Format: SC37 Link. Posted 2007-11-03. This document was submitted to M1 for consideration as M1/07-0490. Readers unable to access either link document should contact NIST. http://isotc.iso.org/livelink/livelink/6904418/JTC001-SC37-N-2296.pdf?func=doc.Fetch&nodeid=6904418 http://www.incits.org/tc_home/m1htm/2007xdocs/m1070490.pdf
LG	S. Shah, *Comment on Iris Boundary Determination (Polar Representation) and Polar Format*, September 7, 2007. This document is a password protected US contribution to the January 2008, SC37/WG3 meeting in Tel Aviv. http://www.incits.org/tc_home/m1htm/2007xdocs/m1070486.pdf
MINEX	P. Grother et al., *Performance and Interoperability of the INCITS 378 Template*, NIST IR 7296 http://fingerprint.nist.gov/minex04/minex_report.pdf
MOC	P. Grother and W. Salamon*, MINEX II - An Assessment of ISO/IEC 7816 Card-Based Match-on-Card Capabilities* http://fingerprint.nist.gov/minex/minexII/NIST_MOC_ISO_CC_interop_test_plan_1102.pdf
N2059	JTC001-SC37-N-2059 — *German National Body contribution on the revision project on Iris Image Data standard ISO/IEC 19794-6*. April 20, 2007
N2124	JTC001-SC37-N-2124 — *UK Contribution on a defect in ISO/IEC 19794-6 polar iris image format*. May 29, 2007
N2226	JTC001-SC37-N-2226 — *Base document for revision of ISO/IEC 19794-6, Information technology: Biometric data interchange formats – Part 6: Iris image data*
NIST	P. Grother, *NIST comments toward US position on N2226, revision of ISO/IEC 19794-6*, September 7, 2007 http://www.incits.org/tc_home/m1htm/2007xdocs/m1070488.pdf
PERFSTD	ISO/IEC FDIS 19795-4 — Biometric Performance Testing and Reporting — Part 4: Interoperability Performance Testing. Posted as document 37N2370.
PROC	Hugo Proença and Luís A. Alexandre, *Iris Recognition: An Analysis of the Aliasing Problem in the Iris Normalization Stage*
STD05	ISO/IEC 19794-6:2005 — *Information technology — Biometric data interchange formats — Part 6: Iris image data* The standard was published in 2005, and can be purchased from ANSI at http://webstore.ansi.org/ or ISO.
UNSEG	D. Kim, *Introducing the Unsegmented Polar Format*, posted as m1070606 and m1070606rev to the M1 document register: http://m1.incits.org/m1htm/2007docs/m1docreg_2007.htm. These documents are password protected US contributions to the January 2008, SC37/WG3 meeting in Tel Aviv. Substantially the same information was presented to the Second NIST Quality Workshop on November 8, 2007 and is linked here: http://www.itl.nist.gov/iad/894.03/quality/workshop07/presentations.html as http://www.itl.nist.gov/iad/894.03/quality/workshop07/proc/Kim_Introducing_Unsegmented_Polar_Data_Format_for_NIST_Presentation_20071106_forPrinting.pdf
USNA	Robert W. Ives, Bradford L. Bonney, and Delores M. Etter, *Effect of Image Compression on Iris Recognition*, IMTC 2005 – Instrumentation and Measurement Technology Conference, Ottawa, Canada, 17-19 May, 2005.

1

A.7 Reporting of results

A.7.1 Reports

The Government will combine appropriate results into one or more IREX reports. Together these will contain, at a minimum, descriptive information concerning IREX, descriptions of each experiment, and aggregate test results. NIST will include

- DET performance metrics as the primary indicators of one-to-one verification accuracy,
- ISO/IEC 19795-4 interoperability matrices as the primary measures of interoperability, and
- Image generation, template generation, and matching timing statistics.

NIST may compute and report other aggregate statistics.

MOD Jun 20, 2008: NIST intends to publish results in one or more NIST Interagency Reports. The reports will contain

- contain the names of participants,
- contain the results of all participants' implementations with attribution to the participants.

A.7.2 Pre-publication review

Participants will have an opportunity to review and comment on the reports. Participants' comments will be either incorporated into the main body of the report (if it is decided NIST reported in error) or published as an addendum. Comments will be attributed to the participant.

A.7.3 Citation of the report

Subsequent to publication of our reports Participants may decide to use the results for their own purposes. Such results shall be accompanied by the following phrase: "Results shown from the Iris Exchange Test (IREX) do not constitute endorsement of any particular system by the U. S. Government." Such results shall also be accompanied by the URL of the IREX Report on the IREX website, http://iris.nist.gov/irex.

A.7.4 Rights and ownership of the data

Any data generated, deduced, measured or otherwise obtained during IREX (excepting the submitted SDK itself), as well as any documentation required by the Government from the participants, becomes the property of the Government. Participants will not possess a proprietary interest in the data and/or submitted documentation.

A.8 Return of the supplied materials

NIST will not return any supplied software, documentation, or other material to vendors.

A.9 Agreement to participate

With the signing of this form, Participants attest that they will not file any IREX-related claim against IREX Sponsors, Supporters, staff, contractors, or agency of the U.S. Government, or otherwise seek compensation for any equipment, materials, supplies, information, travel, labor and/or other participant provided services.

The Government is not bound or obligated to follow any recommendations that may be submitted by the Participant. The United States Government, or any individual agency, is not bound, nor is it obligated, in any way to give any special consideration to IREX Participants on future contracts, grants or other activities.

With the signing of this form, Participants realize that any test details and/or modifications that are provided in the IREX website supersede the information on this form.

With the signing of this form, Participants realize that they cannot withdraw from the IREX without their participation and withdrawal being documented in the IREX Final Report.