IREX

An Evaluation-based Program for the Development of Exchangeable Iris Imagery

and Support for Compact Interoperable ISO/IEC 19794-6 Records

Iris Exchange (IREX) Evaluation 2008

Concept, Evaluation Plan and API

Patrick Grother

NIST

November 19, 2007

6.9. Compression metrics

JPEG implementations accept a quality parameter that controls the quantization of the DCT coefficients specified in ISO/IEC 10918. The quantitative effect of this parameter is not standardized, and may vary between implementations.

ISO/IEC 15444 JPEG 2000 implementations are parameterized by a target value, such as the number of bits per pixel.

Some input images are more compressible than others. For example, a motion-blurred image will compress well, as will images with large areas of smooth skin content.

NIST intends to establish operational guidance on the application of compression. NIST intends to quantify compression damage in terms of, at least, PSNR as the independent variable. However, compression software does not take this an input parameter and thus practical implementation would require iterative schemes to achieve a desired entropy or PSNR value. This may be tenable if computational expense is not prohibitive. NIST intends to measure and report compression times.

6.10. Provision of sensor information to SDKs

NIST will provide the manufacturer and model information to the image processing functions provided in the SDK. This allows the implementation to tailor its algorithms to known properties of the sensor (e.g. spectral properties of the illuminant).

NIST is not currently in possession of detailed sensor specifications, and it is therefore incumbent on participants to acquire such information and to use it as they see fit.

Q	Does this provision favor some (e.g. experienced) vendors, and should therefore all sensors be identified simply as "unknown"?

6.11. Repetition of the ICE 06 evaluation

IREX 2008 may include, as one slice of its scope, a repetition of the ICE 06 trial[4]. This arises because matching of rectilinear ICE 06 images, compressed at an initial ratio of 1:1, is logically an approximate repetition of the ICE 06 trial. This is approximate because the SDK protocol here differs from that used in ICE.

6.12. Verification performance

The test will embed pure 1:1 template comparisons. It will not enroll a population and thus will not support

- verification systems that run in an identification mode,
- cohort normalization techniques[5], and
- 1:N testing, which is not needed to study the format and compression effects targeted by this study.

6.13. Phased testing

In an attempt to support SC37 timelines, NIST will depart from its usual practice (e.g. MINEX II, ELFT I) and conduct this test in just a single phase. This means the results of testing will be published sometime after the implementations are received, without any interim disclosure of results to the supplier, and without any resubmission of SDKs to NIST. We will nevertheless communicate aberrant behavior to suppliers - for example, if generated records are non-conformant to the ISO standard, or if the SDK is not conformant to our API.

Q	Should NIST adopt a two-phase approach? This would support an improvement loop: interim feedback from NIST – supplier development – SDK resubmission.

[4] The ICE homepage is http://iris.nist.gov/ice

[5] Such normalization methods may still be implemented within the matcher, by, for example, storing a supplier-owned internal background set of templates into the SDK.

Q	Should interim results be provided to the supplier? Or, instead, should an report be written with results that does not identify vendors?

6.14. Open-source code for ISO/IEC 19794-6

As part of NIST's Biometric Data Interchange (BIOMDI) software distribution[6], NIST has exposed its open-source "C" code project for the reading, writing and validation of ISO/IEC 19794-6 records. The software is freely distributable. It is under formal version control and subscribers to the server are automatically informed of development activities.

6.15. Standards-based evaluation

The IREX evaluation is an interoperability test. As such, NIST intends to conduct it in conformance to the ISO/IEC FDIS 19795-4 — *Biometric Performance Testing and Reporting — Part 4: Interoperability Performance Testing* standard. This standard establishes requirements for the execution of multi-supplier biometric interoperability tests, primarily those evaluating standardized interchange formats such as ISO/IEC 19794-x.

Q	Should NIST release a reference rectilinear to polar interpolation routine?

7. Data structures

7.1. Overview

Much of IREX 08 will involve the generation or use of the data structures defined in the following two subsections. The first is the ISO/IEC 19794-6:2005 record. The second is the proposed unsegmented polar augmentation of that record. In both cases, the values in certain fields have been constrained to specific values. For example, the number of images is always 1, and all image data is in uncompressed form.

7.2. Identifying the records of the standard

The ISO/IEC 19794-6 standard [STD05] required all iris biometric data blocks (BDBs) to be wrapped in CBEFF headers. CBEFF, published as ISO/IEC 19785-1, advances abstract fields and values for encapsulating, signing and encrypting BDBs.

For IREX 08, CBEFF headers shall be absent.

In ISO/IEC 19794-6:2005, the only means of differentiating rectilinear records from polar records is via the CBEFF fromat type, as shown in Table 5. However, CBEFF encapsulation is now optional in the draft revision [N2226] and there is no normative mechanism for determining the encoding. This presents a bug worthy of WG3 attention.

Table 5 – Identifying variants via CBEFF format type

Geometric Encoding	CBEFF_BDB_format_type
Rectilinear	0x0009
Polar	0x000B

For IREX 08, in order to differentiate the various forms, NIST will deviate from the standard and require the Version Number field to be populated and respected as per Table 6.

Table 6 – Identifying variants using Version Number

Geometric Encoding	Version Number	
All encodings in ISO/IEC 19794-6:2005	'nnn'	The 2005 standard was published with the undefined entry 'nnn'. This differs from all other parts 19794-x which stated definitive hex values.

[6] The BIOMDI repository currently includes code for handling standardized iris, face, finger and minutiae records. Instructions for access are described here: http://www.itl.nist.gov/iad/894.03/nigos/biomdi.html

All encodings in ISO/IEC 19794-6:200X, X > 7	0x30333000	'030\0' - the draft revision [N2226] value.
Rectilinear	0x30313000	'010\0'
No-inner-boundary Polar	0x30313100	'011\0'
Full Polar	0x30313200	'012\0'
Proposed unsegmented Polar	0x30313300	'013\0'

7.3. Rectilinear image record structure

Rectilinear image generators shall produce records conformant to Table 7. This is identically the structure of the ISO/IEC 19794-6:2005 standard.

Table 7 – Structure of the ISO/IEC 19794-6:2005 standard

	Section title and/or field name		L	IREX actual or required values	Remarks
	Iris Record Header (Table 2 of ISO/IEC 19794-6:2005)				
1.	Format Identifier		4 B	0x49495200	i.e. ASCII "IIR\0"
2.	Version Number		4 B	0x30313000 0x30313100 0x30313200	See Table 6
3.	Record Length		4 B	$45 \le L \le 2^{32} - 1$	Total length
4.	Capture Device ID		2 B	0	Value will be provided to SDK
5.	Number of eyes imaged		1 B	1	
6.	Record header length		2 B	45	
7.	Iris image properties	Horz. orientation	2 b	ORIENTATION_BASE	
8.		Vert. orientation	2 b	ORIENTATION_BASE	
9.		Scan type	2 b	SCAN_TYPE_PROGRESSIVE, SCAN_TYPE_INTERLACE_FRAME	Rectilinear format only
10.		Occlusions	1 b	0	Polar format only
11.		Occlusion filling	1 b	0	Polar format only
12.		Boundary extraction	1 b	0	Polar format only
13.	Iris diameter (rect)		2 B	> 0	Rectilinear format only
14.	Image format		2 B	IMAGEFORMAT_MONO_RAW	IREX SDKs are neither permitted nor required to compress or decompress image data. Thus this field shall not be IMAGEFORMAT_MONO_JPEG, or IMAGEFORMAT_MONO_JPEG2000.
15.	Raw image width		2 B	> 0	
16.	Raw image height		2 B	> 0	
17.	Intensity depth		1 B	8	
18.	Image transformation (polar only)		1 B	1	TRANS_STD = 1 i.e. linear radial interpolation per [STD05, 6.5.4].
19.	Device unique identifier		16 B	'0000000000000000'	16 character zeroes '0' = 0x30, not '\0' = 0x00..
	Iris Biometric Subtype Header (Table 3 of ISO/IEC 19794-6:2005)				
20.	Eye		1 B	EYE_RIGHT, EYE_LEFT	
21.	Number of iris images of this eye		2 B	1	
	Iris Image Header (Table 4 of ISO/IEC 19794-6:2005)				
22.	Image number		2 B	1	This field is an index starting at 1
23.	Quality		1 B		
24.	Rotation angle of eye		2 B	ROT_ANGLE_UNDEF > 0	For rectilinear For polar unless next line is set non-zero
25.	Rotation uncertainty		2 B	ROT_UNCERTAIN_UNDEF > 0	For rectilinear and polar when rotation is not estimated. For polar when rotation is estimated.

26.	Image length	4 B	LEN	
27.	Image data	LEN		

7.4. Standard polar image record structure

Polar image generators shall produce records conformant to Table 7. This is identically the structure of the ISO/IEC 19794-6:2005 standard. Note that the standard treats rotation estimates and uncertainties differently for rectilinear and polar instances, and thus appropriate values shall be assigned. This specification does not limit implementations to circular models.

7.5. Unsegmented Polar image record structure

The following structure is advanced as an implementation of the proposed unsegmented polar addition to the ISO standard [UNSEG]. The fields in yellow do not appear in the existing 2005 standard.

Q	The [UNSEG] proposal suggests that non-circular boundary information should also be encoded in the revised standard. Should NIST include fields for the storage of such? If so, how?

Table 8 – Structure for the proposed unsegmented polar format

	Section title and/or field name		L	IREX actual or required values	Remarks
	Iris Record Header (Table 2 of ISO/IEC 19794-6:2005)				
28.	Format Identifier		4 B	0x49495200	i.e. ASCII "IIR\0"
29.	Version Number		4 B	0x30313300	i.e. ASCII "013\0". This value is non-standard, and instituted by NIST to differentiate it from the properly standardized record in Table 7.
30.	Record Length		4 B	$57 \le L \le 2^{32} - 1$	Total length
31.	Capture Device ID		2 B	0	Value will be provided to SDK
32.	Number of eyes imaged		1 B	1	
33.	Record header length		2 B	45	
34.	Iris image properties	Horz. Orientation	2 b	ORIENTATION_BASE	
35.		Vert. orientation	2 b	ORIENTATION_BASE	
36.		Scan type	2 b	SCAN_TYPE_PROGRESSIVE, SCAN_TYPE_INTERLACE_FRAME	Rectilinear format only
37.		Occlusions	1 b		Polar format only
38.		Occlusion filling	1 b		Polar format only
39.		Boundary extraction	1 b		Polar format only
40.	Iris diameter (rect)		2 B		Rectilinear format only
41.	Image format		2 B	IMAGEFORMAT_MONO_RAW	IREX SDKs are neither permitted nor required to compress or decompress image data. Thus this field shall not be IMAGEFORMAT_MONO_JPEG, or IMAGEFORMAT_MONO_JPEG2000.
42.	Raw image width		2 B	> 0	
43.	Raw image height		2 B	> 0	
44.	Intensity depth		1 B	8	
45.	Image transformation (polar only)		1 B	1	TRANS_STD = 1 i.e. linear radial interpolation per [STD05, 6.5.4].
46.	Device unique identifier		16 B	'0000000000000000'	16 character zeroes '0' = 0x30, not '\0' = 0x00..
	Iris Biometric Subtype Header (Table 3 of ISO/IEC 19794-6:2005)				
47.	Eye		1 B	EYE_RIGHT, EYE_LEFT	
48.	Number of iris images of this eye		2 B	1	
	Iris Image Header (Table 4 of ISO/IEC 19794-6:2005)				
49.	Image number		2 B	1	This field is an index starting at 1
50.	Quality		1 B		
51.	Rotation angle of eye		2 B	ROT_ANGLE_UNDEF	Values will not be provided to SDK

52.	Rotation uncertainty	2 B	ROT_UNCERTAIN_UNDEF	
53.	Inner circle X coordinate	2 B	0 ≤ x < W	Coordinate system is zero oriented with (0,0) at the top left corner. The inner and outer circle centers are not necessarily the same.
54.	Inner circle Y coordinate	2 B	0 ≤ y < H	
55.	Inner circle radius	2 B	0 ≤ r	
56.	Outer circle X coordinate	2 B	0 ≤ x < W	
57.	Outer circle Y coordinate	2 B	0 ≤ y < H	
58.	Outer circle radius	2 B	0 ≤ r	
59.	Image length	4 B	LEN	See Line 60
60.	Unsegmented polar-transformed image data	LEN		

8. PC-based API specification

8.1. 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.

8.2. Testing interface

8.2.1. Requirement

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

8.2.2. Sensor identifiers

The following sensors will be identified to the SDK using the two byte unsigned integer values in Table 9.

Table 9 – Sensor identifiers

#	Sensor Manufacturer and Model	Identifier
1	LG 2200	0x2A16
2	LG 3000	0x2A1E
3	Unknown	0x0000

Presence on this table indicates NIST's intention to use images captured by these devices. NIST may add to this table in due course. Also, please see NIST's call for images on Page 2.

8.2.3. Conversion of raw imagery to standard rectilinear image

To support the standard all submissions to IREX 08 shall implement a function to execute the packaging operation of Table 10. While this action is merely syntactic, prior interoperability tests conducted by NIST have revealed a not uncommon inability to reliably instantiate conformant records.

The output instances are used as inputs to subsequent functions (polar processing, or feature extraction). NIST will apply compression-decompression to the image data contained in these records.

Table 10 – Preparation of standard rectilinear records

Input	Action	Output
Raw raster, dimensions, and parameters needed to instantiate the standardized record	Package the input into a conformant record.	Conformant ISO/IEC 19794-6 rectilinear record. These records will be checked for conformance by NIST.

The function shall be implemented with the API call specified in Table 11.

Table 11 – IREX API preparation of standard rectilinear record

Prototype	INT32 convert_raster_to_rectilinear(

	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 image_format, const BYTE intensity_depth, const UINT16 nist_encoded_device_id, BYTE *quality BYTE * ISO_19794_6_rectlinear_image);	
Description	This function takes a raw input image and outputs the corresponding ISO/IEC 19794-6 rectilinear record. This function executes only a syntactic repackaging of the input data. It shall not alter the image data.	
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.
	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 9
Output Parameters	Quality	A [0,100] quality value representing
	ISO_19794_6_rectlinear_image	The output rectilinear image, per Table 7
Return Value	This function returns zero on success or a documented non-zero error code otherwise.	

8.2.4. Conversion of rectilinear to standard polar

To assess viability of the standard's polar format, participating submissions to IREX 08 shall execute the conversion operation of Table 12.   In addition to the required polar instance output, the function shall return center coordinates. This supports measurements of the pixel-level displacement between segmentation algorithms.

Table 12 – Preparation of standard polar records

Input	Action	Output
Conformant ISO/IEC 19794-6 rectilinear record, and desired circumferential and radial numbers of samples.	Find suitable inner and outer circles. Execute forward polar transform.	Segmented polar record, per Table 7. Coordinates of the centers of the inner and outer regions.

The function shall be implemented with the API call specified in Table 13.

Table 13 – IREX API for creation of polar records

Prototype	INT32 convert_rectilinear_to_segmented_polar(const BYTE *ISO_19794_6_rectlinear_image, const UNIT16 num_samples_radially, const UINT16 num_samples_circumferentially, UINT16 *x_iris_center, UINT16 *y_iris_center, UINT16 *x_pupil_center, UINT16 *y_pupil_center, BYTE *ISO_19794_6_seg_polar_image);
Description	This function takes a conformant ISO/IEC 19794-6 rectilinear image and outputs the corresponding segmented polar image as a conformant ISO/IEC 19794-6 instance. The coordinates of the pupil and iris centers are returned also.

	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.	
Input Parameters	ISO_19794_6_rectlinear_image	The uncompressed raw image used for template creation.
	num_samples_radially	The requested number from inner to outer boundary
	num_samples_circumferentially	The requested number of samples around the limbus
Output Parameters	x_iris_center	Horizontal and vertical locations of the iris center.
	y_iris_center	
	x_pupil_center	Horizontal and vertical locations of the pupil center.
	y_pupil_center	
	ISO_19794_6_seg_polar_image	The output polar image, per Table 7.
Return Value	This function returns zero on success or a documented non-zero error code otherwise.	

An implementation may elect not to process the input image, or it may fail involuntarily. If this occurs, a documented non-zero error code will be returned and NIST will record this as a failure to segment.

Q	Should NIST recommend or require that the grey level data is not processed? For example, eyelids are not removed, frequency domain filtering is not performed, or histogram equalization is not applied?

Q	Should this function include an input over-segmentation parameter? The values for such a parameter would take on a limited number of values, say 100 for exact detection of the limbus, and 115 for a larger region [LG].

8.2.5. Conversion of rectilinear to no-inner-boundary standard polar

To assess viability of the standard's no-inner-boundary polar format, participating submissions to IREX 08 shall execute the conversion operation of Table 14. In addition to the required polar instance output, the function shall return center coordinates. This supports

- measurement of the pixel-level displacement between segmentation algorithms, and
- generation of a square bounding boxes to be used for generation and testing of cropped rectilinear iris records.

Table 14 – Preparation of standard no-inner-boundary polar records

Input	Action	Output
Conformant ISO/IEC 19794-6 rectilinear record, and desired circumferential and radial numbers of samples.	Find suitable inner and outer circles. Execute forward polar transform.	Standard no-inner-boundary polar record, per Table 7. Coordinates of the centers of the inner and outer regions.

The function shall be implemented with the API call specified in Table 15.

Table 15 – IREX API for creation of no-inner-boundary polar records

Prototype	INT32 convert_rectilinear_to_no_inner_boundary_polar(const BYTE *ISO_19794_6_rectlinear_image, const UNIT16 num_samples_radially, const UINT16 num_samples_circumferentially, UINT16 *x_iris_center, UINT16 *y_iris_center, UINT16 *x_pupil_center, UINT16 *y_pupil_center, BYTE *ISO_19794_6_seg_polar_image);
Description	This function takes a conformant ISO/IEC 19794-6 rectilinear image and outputs the corresponding no-inner-boundary segmented polar image as a conformant ISO/IEC 19794-6 instance. The coordinates of the pupil and iris centers are returned also. 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.	
Input Parameters	ISO_19794_6_rectlinear_image	The uncompressed raw image used for template creation.
	num_samples_radially	The requested number from inner to outer boundary
	num_samples_circumferentially	The requested number of samples around the limbus
Output Parameters	x_iris_center	Horizontal and vertical locations of the iris center.
	y_iris_center	
	x_pupil_center	Horizontal and vertical locations of the pupil center.
	y_pupil_center	
	ISO_19794_6_seg_polar_image	The output no-inner-boundary polar image, per Table 7.
Return Value	This function returns zero on success or a documented non-zero error code otherwise.	

An implementation may elect not to process the input image, or it may fail involuntarily. If this occurs, a documented non-zero error code will be returned and NIST will record this as a failure to segment.

Q	Should NIST recommend or require that the grey level data is not processed? For example, eyelids are not removed, frequency domain filtering is not performed, or histogram equalization is not applied?

Q	Should this function include an input over-segmentation parameter? The values for such a parameter would take on a limited number of values, say 100 for exact detection of the limbus, and 115 for a larger region [LG].

8.2.6. Conversion of rectilinear to unsegmented polar

To examine the viability of the proposed unsegmented polar format, participating submissions to IREX 08 shall execute the conversion of operation of Table 16. To allow surveys over the radial and circumferential sampling rates the function takes as input number-of-samples arguments.

Table 16 – Preparation of unsegmented polar records

Input	Action	Output
Conformant ISO/IEC 19794-6 rectilinear record, and desired circumferential and radial numbers of samples.	Find suitable inner and outer circles. Execute forward polar transform.	Proposed unsegmented polar record, per Table 8.

The function shall be implemented with the API call specified in Table 17.

Table 17 – IREX API for creation of unsegmented polar records

Prototype	INT32 convert_rectilinear_to_unsegmented_polar(const BYTE *ISO_19794_6_rectlinear_image, const UNIT16 num_samples_radially, const UINT16 num_samples_circumferentially, BYTE *unseg_polar_image);	
Description	This function takes a conformant ISO/IEC 19794-6 rectilinear image and outputs the corresponding unsegmented polar image. 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.	
Input Parameters	ISO_19794_6_rectlinear_image	The uncompressed raw image as prepared by the function in Table 11.
	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.
Output Parameters	ISO_19794_6_unseg_polar_image	The output template, per Table 8.
Return Value	This function returns zero on success or a documented non-zero error code otherwise.	

Q	Should NIST recommend or require that the grey level data is not processed? For example, eyelids are not masked, frequency domain filtering is not performed, or histogram equalization is not applied?

Q	Should this function include an input over-segmentation parameter? The values for such a parameter would take on a limited number of values, say 100 for exact detection of the limbus, and 115 for a larger region [LG].

An implementation may elect not to process the input image, or it may fail involuntarily. If this occurs, a documented non-zero error code will be returned and NIST will record this as a failure to segment.

8.2.7. Template creation

This function converts any of the input data structures to an opaque proprietary template. The function will need to look at the header of the input record to determine the content. This routine will need to heed the hexadecimal "Version Number" values in the Table 7 and Table 8 records.

Table 18 – IREX API template creation

Prototype	INT32 convert_image _to_template(const BYTE *input_record, BYTE *proprietary_template);	
Description	This function takes either — a conformant ISO/IEC 19794-6 rectilinear image, or — a conformant ISO/IEC 19794-6 polar image, or — a proposed standard unsegmented polar image, and outputs a proprietary template. The implementation should inspect the input header to determine which kind of imagery is being provided. The memory for the output 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 template record that may be passed to the match_templates function without error.	
Input Parameters	input_record	An input image presented either as instance of Table 7 or Table 8..
Output Parameters	proprietary_template	The output template. The format is entirely unregulated, but it shall include its own length as this will not be provided to the subsequent matcher function.
Return Value	This function returns zero on success or a documented non-zero error code otherwise.	

An implementation may elect not to make a template, or it may fail involuntarily. If this occurs, a documented non-zero error code will be returned, size then NIST will record this as a failure to acquire.

8.2.8. Template comparison

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

Table 19 – 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 *score);	
Description	This function compares two opaque proprietary templates and outputs a non-negative match score. The returned score is a distance measure. It need not satisfy the metric properties. NIST will allocate memory for this parameter before the call. When the function is called with either or both templates containing zero data (see **Error! Reference source not found.**) the function shall assign the value -1 to the score.	
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	score	A similarity score resulting from comparison of the templates, on the range [0,DBL_MAX].
Return Value	This function returns zero on success (i.e. a valid score is produced) or a documented non-zero error code on failure.	

Q	Should NIST add a polarity flag to indicate smaller-is-more-genuine vs. larger-is-more-genuine, or are all suppliers content with smaller-is-more-genuine distance scores?

Q	Will templates be larger than 65535 bytes in size?

8.2.9. Implementation identifiers

The implementation shall support the self-identification function of Table 20. 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 20 – IREX API get_pids function

Prototype	INT32 get_pid(UINT32 *nist_assigned_identifier);	
Description	This function retrieves an identifier that the provider must request from NIST, and compile 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.
Return Value	This function returns zero on success or a documented non-zero error code on failure.	

8.3. Software and Documentation

8.3.1. SDK Library and Platform Requirements

Participants shall provide NIST with binary code only (i.e. no source code) – supporting files such as header (".h") files notwithstanding. Such files 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/shared library files are permitted.

If dynamic/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/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).

8.3.2. 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

— The cygwin[7] layer running on a Windows Server 2003 OS.

— RedHat Linux Enterprise 4 or 5 platforms.

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
```

[7] 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.

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.

8.3.3. 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.

8.3.4. 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 error and warning codes (see 8.4.3).

8.3.5. 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.

8.4. Runtime behavior

8.4.1. Speed

The following limits are instituted to constrain NIST's total IREX computational workload.

- The mean template match operation shall not exceed 10 milliseconds.
- The mean template creation operation shall not exceed 1.2 seconds (using a 2GHz Pentium IV).
- The mean iris segmentation operation (e.g. polar) shall not exceed 1.2 seconds (using a 2GHz Pentium IV).

d	Should these times be lowered (for operational relevance) or raised (for improved algorithmic function)?

8.4.2. 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".

8.4.3. Error codes and status messages

The SDK will be tested in non-interactive "batch" mode (i.e. 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". Instead, the SDK shall conform to the error notification procedures of Annex A.

8.4.4. 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 cease evaluation, inform the provider and document evidence of such in published reports.

8.4.5. Stateful behavior

All components in this test shall be stateless. This applies to segmentation, feature extraction and matching. Thus the 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

— cease evaluation of all implementations from the supplier,

— inform the supplier, and

— publicly report the behavior, with attribution and documentary evidence.

9. NIST conversion of Rectilinear to Polar Coordinates

Q	Rather than have suppliers execute the conversion of rectilinear to polar format, should NIST instead require the supplier to return the parameters of the transform with which NIST would then execute it.? NIST's transformation code would include neither boundary detection nor segmentation code.

10. 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.
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
MINEX	P. Grother et al., *Performance and Interoperability of the INCITS 378 Template*, NIST IR 7296 http://fingerprint.nist.gov/IREX04/IREX_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
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.
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

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
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
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.
PROC	Hugo Proença and Luís A. Alexandre, *Iris Recognition: An Analysis of the Aliasing Problem in the Iris Normalization Stage*
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
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.

Annex A
Error Codes and Handling

Each participant shall provide their SDK with text documentation of all (non-zero) error or warning return codes (see section 8.3.4, Documentation).

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.

At minimum the bit masked return codes of Table 21 shall be set and returned.

Table 21 – IREX API return codes

Return code	Explanation
0	Success
1	Image size not supported
2	Failed to find iris
4	Failed to find pupil center
8	Failed to find iris center
16	Failed to match templates
32	Failed to match templates – unable to parse probe template
64	Failed to match templates – unable to parse gallery template
> 128	Supplier defined error codes in here.

All messages which convey errors, warnings or other information shall be suppressed.

B.6 Access to IREX test data

The IREX Test Datasets are protected under the Privacy Act (5 U.S.C. 552a), and will be treated as Sensitive but Unclassified and/or Law Enforcement Sensitive.

IREX Participants shall have no access to IREX Test Data, either before, during or after the test.

B.7 Reporting of results

B.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

— enrollment and verification timing statistics.

NIST may compute and report other aggregate statistics.

NIST intends to release Phase 1 results to the participant only.

NIST intends to publish Phase 2 results in one or more NIST Interagency Reports.

The Phase 2 reports will:

— contain the names of Phase 1 participants,

— not contain the results from Phase 1 participants' implementations,

— contain the names of Phase 2 participants, and

— contain the results of all Phase 2 participants' implementations which will associated with the participants names.

B.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.

B.7.3 Citation of the report

After the release of the Phase II Final Report, 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.

B.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.

B.8 Return of the supplied materials

B.8.1 Returning software to vendors

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