A biometric reference system for iris
OSIRIS version 4.1

Foreword. *This version of OSIRIS_v4.1 was developed by Telecom Sud Paris (previously GET-INT) by Guillaume Sutra, Bernadette Dorizzi et Sonia Garcia-Salicetti. This documentation was written by Nadia Othman.*

Abstract. *The OSIRIS (Open Source for IRIS) reference system is an open source iris recognition system developed in the framework of the BioSecure project [1]. The system is inspired by Daugman works [2]. It is composed of four processing modules (segmentation, normalisation, encoding and matching), which correspond to the four steps of the complete iris recognition procedure. In version v4.1, the segmentation part uses a Viterbi algorithm to find the boundaries of the iris and of the pupil [3]. The normalisation step is based on Daugman's rubber-sheet model and it uses the coarse contours detected by the Viterbi. Note that there is no assumption of circularity of the normalized iris and pupil contours. The encoding and matching parts are based respectively on Gabor phase demodulation and Hamming distance classification.*

Introduction:

This document describes the version 4.1 of the OSIRIS reference system and explains how to use it in practice to build an automatic verification experiment based on iris modality. This document is divided into fives parts. In the first part, we present how to install the system. Then, in the second part, the database used and the protocol assiciated are described. In the third part, the role of each module and how to use it are explained. In the fourth part, the algorithms of each module are detailed and finally, the experimental results obtained in the verification are shown in the last part. This document refers to version 4.1 of the software in which the segmentation and the normalization parts have been greatly improved over the previous versions.

A. Installation

The full system has been tested under Linux (Fedora 15.0). For the compilation step, we have used as compiler g++ 4.6.1.

1. Downloading:

The OSIRIS reference system is currently available on the Telecom SudParis subversion server at the following address :

http://share.int-evry.fr/svnview-eph/

The source files are available in the part *ref_syst/Iris_Osiris_v4.1/download/*. Download the archive Iris_Osiris_4.1.tar.gz in your home directory */home/Iris_Osiris_v4.1/*, and then decompress it as follows :

```
> tar zxvf Iris_Osiris_v4.1.tar.gz
```

OSIRIS_v4.1 uses some function of the OpenCV library. So, you need to install OpenCV before compiling OSIRIS source code. If OpenCV is already installed on your machine, skip this part and go to the next section.

To install OpenCV library, download the lastest version at the following address :

http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/

Decompress it in your home (for example) as follows:

```
> tar zxvf opencv-2.4.5.tar.gz
```

Then do the following steps to build opencv:

```
> cd opencv-2.4.5/
```

```
> mkdir release
```

```
> cd release/
```

```
> cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D BUILD_PYTHON_SUPPORT=ON ..
```

Then compile it by typing:

```
> make -j
```

Note: Make sure that opencv.pc is found by pkg-config. Type the following command:

```
> pkg-config opencv --libs
```

=> It should print something like -lopencv_core -lopencv_imgproc ...

OpenCV is now correctly installed.

2. Compilation:

To compile the source code, perform the following steps :

- Enter in the *src* directory :

```
> cd Iris_Osiris_v4.1/src
```

- Compile the source code :

```
> make all
```

This compilation step creates the executable **osiris** in your directory */home/Iris_Osiris_v4.1/src.*

For more information, see: *http://opencv.willowgarage.com/wiki/InstallGuide*

B. Reference database

The National Institute of Standards and Technology (NIST) has provided researchers the database ICE 2005 [4]. This database consists of 2953 grayscale eye images of 132 people. They are acquired with a dedicated LG2200 camera. Each image captures one eye and has a size of 640x480 pixels. This database has been divided into two sub-databases : one for images of the right iris (1425 iris images from 124 persons) and another one for images of the left iris (1528 iris image of 120 persons). In most cases, images of the right and left irises are acquired at the same time. The images in this database contain reflections, blur and interlacing distorsion. In addition, the iris in the database can be covered by eyelids, eyelashes.

Figure 1: Examples of iris images in the ICE 2005 database.

C. Presentation of the software

1. Brief description of the four processing modules

Osiris is composed of four processing modules, corresponding to the four steps of the complete iris recognition procedure. These steps are briefly described below.

1.1. Segmentation

The segmentation step aims at finding accurate contours of iris, that is inner boundary (pupil/iris) and outer boundary (iris/sclera) in order to classify the pixels into two classes : iris and not-iris. This results in building a binary mask, where on-pixels belong to the iris, and off-pixels do not belong to the iris. Moreover, the segmentation step provides two contours (pupil and iris) that will be used by the normalization step.

1.2. Normalization

The normalization step transforms the iris area into a size-invariant strip following Daugman's rubber-sheet method. This step is also applied to the mask.

1.3. Encoding

The encoding step extracts the iris texture by filtering the normalized image by a bank of Gabor filters resulting in an iris template. The Gabor filters are customizable (orientation

and resolution). See section C.3.5.3. The phases of Gabor filtering are encoded on 2 bytes, and produce the iris code.

1.4. Matching

The matching step compares two iris codes using the Hamming distance between the binary codes corresponding to the application points chosen within the iris templates. The application points are customizable (see section C.3.5.4). The masks provided by the segmentation module (or externally provided) can be used to ignore noisy pixels.

2. Input and Output datas

According to your needs, input and output data of the system are not the same. This section describes each data that may be involved through the four modules described in the previous section. The relations between the data and the steps are sum up in Table 1.

	Input of	**Output of**
Original image	Segmentation, Normalization	-
Segmented image	-	Segmentation
Mask of iris	Normalization (optional)	Segmentation
Contours parameters	Normalization	Segmentation
Normalized image	Encoding	Normalization
Normalized mask	Matching	Normalization
Iris template	Matching	Encoding
Matching scores	-	Matching

Table 1: Inputs and Outputs of each step

Each output data can be saved by indicating a directory (or textfile for the matching scores) in the configuration file, as explained in section C.3.

Each input data, excepted the original image, can be computed by Osiris system or loaded from a directory. This allows to experiment external independent algorithm. For instance, it is possible to load segmentation results (mask and contours parameters) from another segmentation algorithm and then to use Osiris only for normalization, and finally use the Masek matching module to achieve the matching step.

2.1. Original image

The original image is an eye image, considered as a grayscale image by Osiris. It means that a color image will be automatically converted as a grayscale image before any process. The original image can be any size and any resolution. The segmentation parameters

(minimum and maximum diameters for pupil and iris) can be adapted to face a non-common resolution (usually 200 pixels across the iris diameter). The original image is required as input of the segmentation step and of the normalization step.

2.2. Segmented image

Once the segmentation is achieved, Osiris can save the segmented image. This is a color image showing the segmentation results: pupil and iris circles are drawn in green, and non-iris pixels indicated by the mask are colored in red. This image only helps to view the results, but is not used by further processings. Figure 2.c illustrates such a segmented image. The circles are found by least-squares based circle fitting method applied to the coarse contour detected by the Viterbi [3]. Notice that in this version of Osiris, we don't use circles for the normalization but directly the coarse contours detected in the segmentation contrary to what is described in [3].

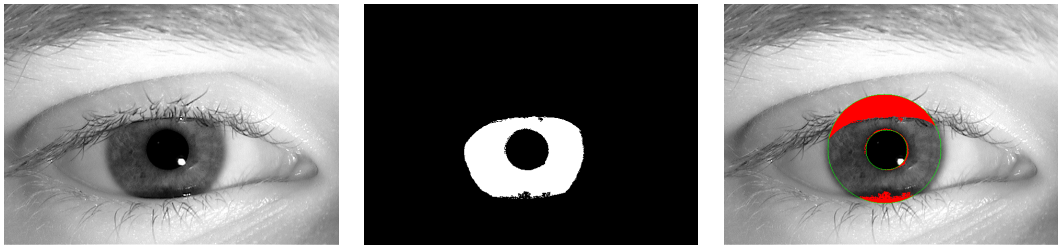

Figure 2: Example of an original image (a), the associated binary mask (b), and the segmented image (c).

2.3. Mask of iris

The segmentation step builds a binary mask of iris, where on-pixels belong to the iris, and off-pixels do not belong to the iris, as shown in Figure 2.b. This binary mask can be used during the matching step in order to ignore noisy pixels, but it must be normalized before the matching step. Thus it is an optional input of the normalization step. If no mask is provided to the normalization step, one blank mask (considering all pixels belong to the iris) will be automatically generated before the normalization step.

The mask can also be externally loaded (for example to test the result of another segmentation algorithm) by indicating an input directory in the configuration file.

2.4. Contours parameters

Two sets of parameters corresponding to the pupil and the iris coarse contours, given by the segmentation step, are used during the normalization step. These parameters are saved in a textfile for each eye, in the following way:

- The first line corresponds to the number of point in the pupil coarse contour;
- The second line corresponds to the number of point in the iris coarse contour;
- The third line corresponds to the coordinates of the pupil coarse contour;
- The last line corresponds to the coordinates of the iris coarse contour.

Each point of the contour has as coordinates: (x_r, y_r, φ) where (x_r, y_r) is the coordinate of the radius relatively to the estimated center and φ the angle between 0 to 2*PI=6.2831 .

For more information see section D.2. An example of the textfile is illustrated in Figure 3.

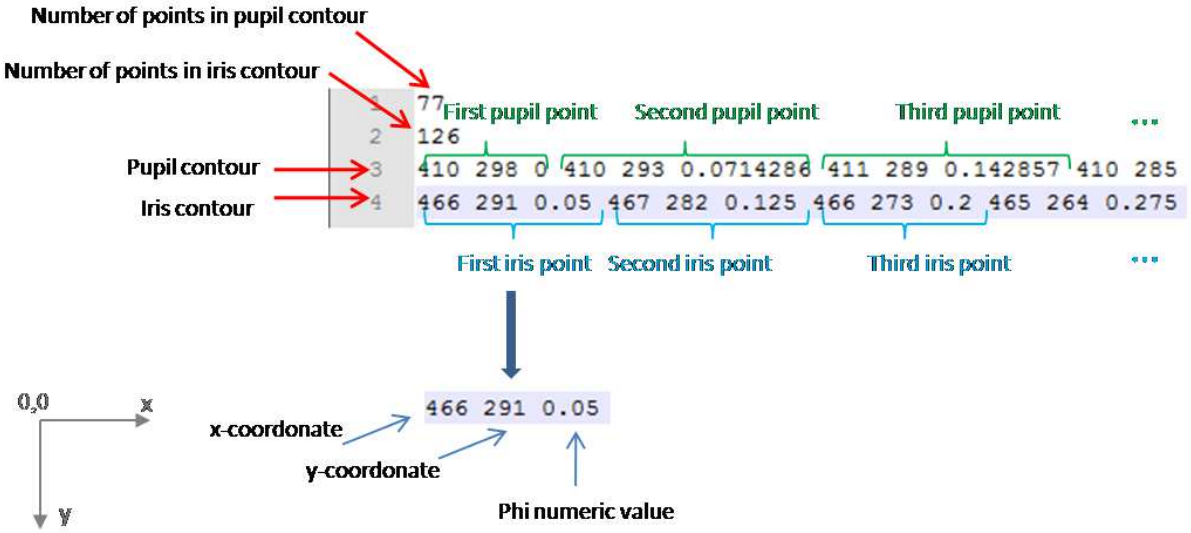

Figure 3 : Example of the content of a textfile for contours parameters.

The contours parameters are a required input for the normalization step. They can be computed by the segmentation step, or externally provided by the user, by indicating an input directory for contours parameters in the configuration file.

2.5. Normalized image

The normalization step transforms the original image into a normalized image, following Daugman's rubber-sheet model (Figure 4.a). The normalized image is required as input of the encoding step. It can be externally loaded by indicating an input directory in the configuration file.

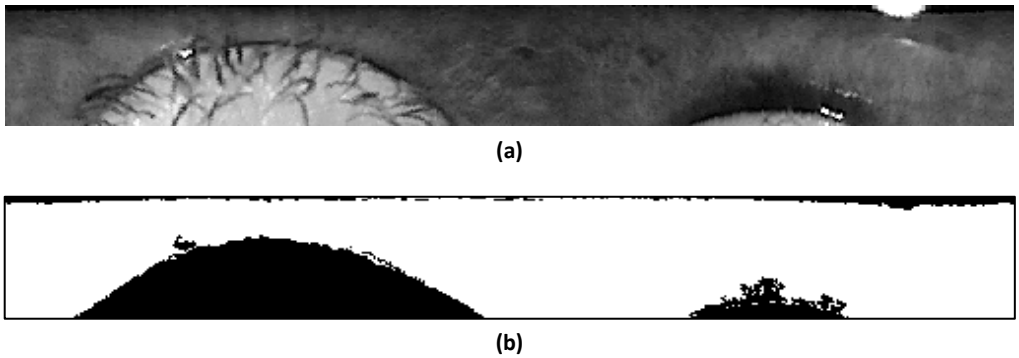

Figure 4 : Example of a normalized image (a) and its associated normalized mask (b)

2.6. Normalized mask

The normalization step also transforms the mask into a normalized mask (Figure 4.b). The normalized mask is not required for the encoding step, but is an optional input of the

matching step, in order to ignore noisy pixels. It can be externally loaded by indicating an input directory in the configuration file.

2.7. Iris template

The encoding step extracts the iris code by filtering the normalized image by Gabor filters. According to Daugman works, only 256 application points within the iris texture are necessary to achieve iris recognition. However, Osiris was developed for tests and experiments, such as changing the number and the position of the application points. That is why all pixels are saved as the iris template, and the application points will be defined only during the matching step. The iris template is saved as a binary image of size W x (n*H), where n is the number of Gabor Filters (actually there are n/2 Gabor filters, with real part and imaginary part, but Osiris consider real part and imaginary part as two independent filters) and W x H is the size of normalized image, as illustrated in figure 4. The iris template is required as input of matching step. It can be externally loaded by indicating an input directory in the configuration file.

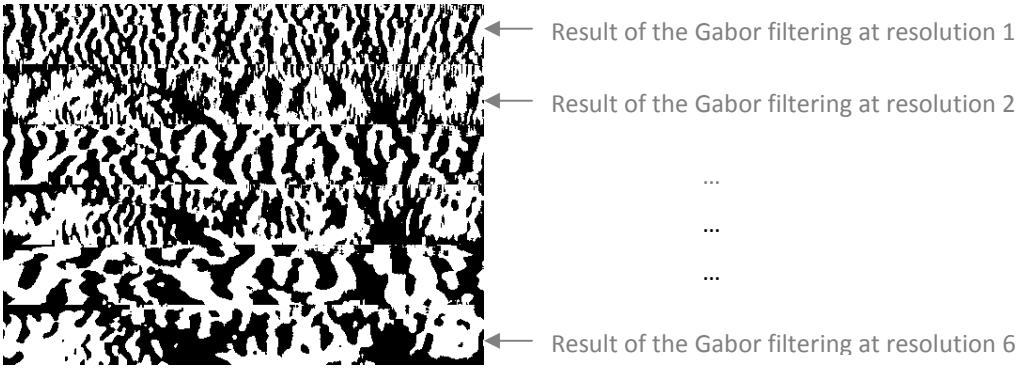

Figure 5 : Example of an iris template computed by Osiris, when 6 Gabor filters are used.

2.8. Matching score

The global disimilarity score between two iris codes is ranged between 0 (completely similar) and 1 (completely different). The matching scores are saved in a textfile following the structure illustrated in figure 5.

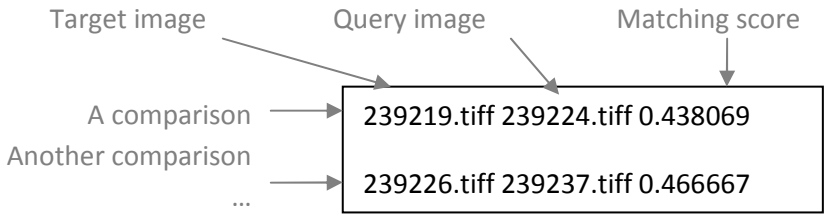

Figure 6 : Example of a textfile containing the matching scores

Figure 7 shows the overall structure of Osiris system.

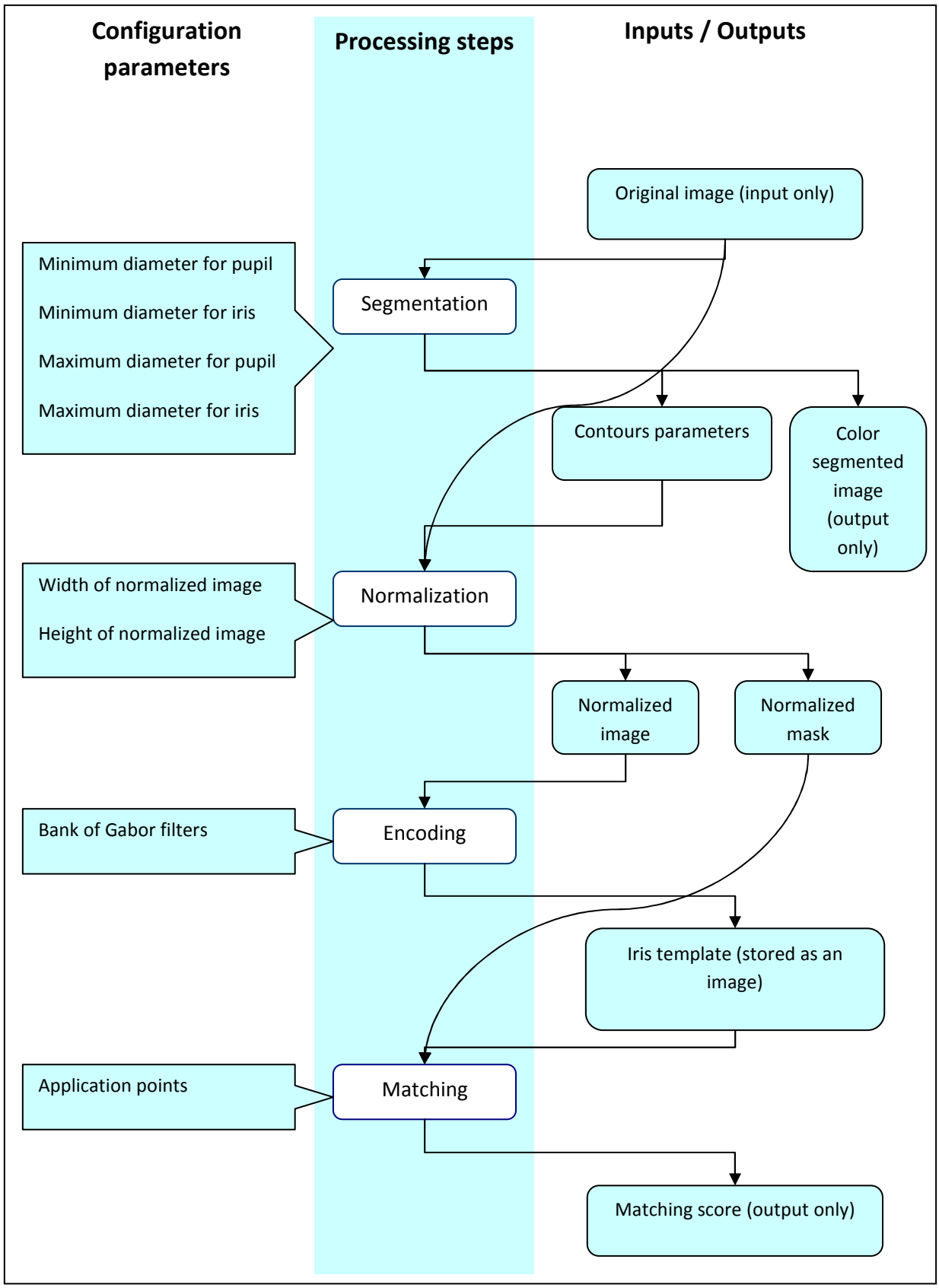

Figure 7 : Overall structure of Osiris_v4.1 system

3. Configuration of Osiris_v4.1

3.1. How to launch Osiris_v4.1

The execution of one or more processings steps on many images is based on the use of the executable *osiris* available in the directory */home/Iris_Osiris_v4.1/src* (see section A) and the configuration script.

Syntax to launch OSIRIS:

```
./osiris <pathConfigurationIniFile>
```

3.2. Overall description of the configuration file

The configuration file allows configuring Osiris in order to achieve one or more processing steps on many images, to select the outputs that must be saved, and to specify some other processing options. The configuration file is structured as an *ini* file, in which a *value* is associated to a *key*. All the keys and possible associated values are sum up in table 2. This configuration file is read by the Osiris system before processing anything.

Key	**Possible values**	**Description**
Process segmentation	Boolean : *yes, true, 1, y, on* *no, false, 0, n, off*	The commands to achieve one or more of the processing steps
Process normalization		
Process encoding		
Process matching		
Use the mask provided by osiris		Usually set to “yes”. Set to “no” only if you use your own masks, or if you want to test with blank masks
List of images	String	Path to the textfile containing the list of images
Load images	String	Input directories Comment the line if you do not want to load the input (in this case it will be computed by Osiris if necessary)
Load parameters		
Load masks		
Load normalized images		
Load normalized masks		
Load iris templates		
Save segmented images	String	Output directories Comment the line if you do not want to save this output
Save contours parameters		
Save masks of iris		
Save normalized images		
Save normalized masks		
Save iris templates		
Save matching scores	String	The name of textfile (with path) in which matching scores will be saved
Minimum diameter for pupil	Integer	> 11 pixels
Maximum diameter for pupil		-
Minimum diameter for iris		> 99 pixels
Maximum diameter for iris		-
Width of normalized image		Common size :

Height of normalized image		512x64
Gabor filters	String	The path to the textfile containing Gabor filters
Application points		The path to the textfile containing the application points
Suffix for segmented images	String	A suffix is added to the eye name to make the difference between all possible datas associated to this eye. *For example: eye1_segm.bmp, eye1_code.bmp, etc…*
Suffix for parameters		
Suffix for masks of iris		
Suffix for normalized images		
Suffix for normalized masks		
Suffix for iris templates		

Table 2 : Keys and possible values of the configuration file.

3.3. How to edit the configuration file

Here is a guideline for quickly editing the configuration file:

1. Choose the processing steps.
2. Indicate the list of images that will be considered. The list of images is a textfile containing the name (without the path) of each eye to be processed. If the matching step is selected, the first eye will be compared to the second one, the third eye will be compared to the fourth one, and so on. You can use blank space, tabulations or end-line character to separate the eye names. It can be more readable to present the list on two columns in case of matching, whereas it is more readable to present the list on one column when matching is not required.
3. Specify the input directories, that means the location of the input data needed by Osiris in order to achieve the processing steps chosen in 1. For example, if the segmentation step is not requested, but the normalization step is, then you must indicate the directory containing the contours parameters **AND** the directory containing the original images (and optionnally the directory containing the masks if you want to normalize the masks as well as the images).
4. Specify the output directories, that means the location of the data produced by Osiris. If you do not want to save data, just comment the corresponding line. In case of matching, the matching scores are saved in a textfile (not a directory).
5. The other parameters are technical settings, and should not be changed as long as you do not experiment something special.

3.4. Example of configuration

Suppose that you have one set of eye images, and you want to use all Osiris steps in order to get matching scores. It could be possible to achieve all this in one runtime, by writing a list of matching couples, and requesting all processes on the whole list. But in that case, the segmentation, normalization and encoding steps will be computed several times for one eye that occurs more than once in the list (which is frequent in a list of matching). It would take a very long and useless time to compute.

Instead, it is recommended to proceed in two runtimes. The first one for segmentation, normalization, and encoding steps, applied on a list containing only one occurrence of each

eye, and saving the iris templates and the normalized masks as outputs. The second one for the matching step, applied on a “matching list”, and taking as input the iris templates and normalized masks from the first runtime. Illustrations of the lists and configuration files are shown in figure 8: 8.a and 8.b correspond to the first runtime (segmentation, normalization, encoding) ; 8.c and 8.d correspond to the second runtime (matching). Note that the matching list in figure 8.c could have been a one-column list. In that case, the first line would be matched with the second line, the third line with the fourth line, and so on.

```
1  239219.tiff
2  239224.tiff
3  239226.tiff
4  239229.tiff
5  239230.tiff
6  239234.tiff
7  239237.tiff
8  239246.tiff
9  239251.tiff
10 239254.tiff
11 239256.tiff
12 239258.tiff
13 239259.tiff
14 239261.tiff
15 239262.tiff
16 239263.tiff
17 239264.tiff
18 239267.tiff
19 239271.tiff
20 239273.tiff
21 239275.tiff
22 239276.tiff
23 239279.tiff
24 239280.tiff
25 239281.tiff
26 239292.tiff
27 239293.tiff
28 239294.tiff
29 239302.tiff
30 239308.tiff
31 239313.tiff
32 ...
```

(a) eyes_list.txt

```
1  # By default, all other process are set to "no"
2  Process segmentation = yes
3  Process normalization = yes
4  Process encoding = yes
5  Process matching = no
6  Use the mask provided by osiris = yes
7
8  # The list of eyes (only one occurrence of each eye)
9  List of images = C:/Database/eyes_list.txt
10
11 # The folder containing the iris codes
12 Load original images = C:/Database/Images/
13
14 # Save the iris codes (stored as images)
15 Save iris codes = C:/Database/Iris_Codes/
16
17 # The normalized masks will be used during matching
18 Save normalized masks = C:/Database/Normalized_Masks/
19
20 # A priori knowledge about the pupil and iris sizes
21 Minimum diameter for pupil = 50
22 Maximum diameter for pupil = 160
23 Minimum diameter for iris = 160
24 Maximum diameter for iris = 280
25
26 # Size for normalization
27 Width of normalized image = 512
28 Height of normalized image = 64
29
30 # For encoding step
31 Gabor filters = ./filters.txt
32
```

(b) configuration.ini

```
239663.tiff 246917.tiff
239663.tiff 241920.tiff
239663.tiff 245029.tiff
239663.tiff 243318.tiff
239663.tiff 242484.tiff
239663.tiff 242988.tiff
239663.tiff 244657.tiff
239663.tiff 245705.tiff
239663.tiff 243585.tiff
239663.tiff 246323.tiff
239663.tiff 239915.tiff
239663.tiff 246856.tiff
239663.tiff 244693.tiff
239663.tiff 243740.tiff
243585.tiff 246156.tiff
243585.tiff 246618.tiff
243585.tiff 247155.tiff
...
```

(c) matching_list.txt

```
# By default, all other process are set to "no"
Process matching = yes

# The list of matching couples (target/query)
List of images = C:/Database/matching_list.txt

# The folder containing the iris codes
Load iris codes = C:/Database/Iris_Codes/

# The folder containing the normalized masks
Load normalized masks = C:/Database/Normalized_Masks/

# Save the matching scores in a textfile
Save matching scores = C:/Database/result_matching.txt

# Only the application points are needed for matching
Application points = ./points.txt
```

(d) configuration.ini

Figure 8 : Lists of eyes and configuration files associated to the example in 3.4.

Two examples of *configuration.ini* files are available in the directories */Iris_Osiris_v4.1/scripts/Template/* and */Iris_Osiris_v4.1/scripts/Matching/*. The first example, execute the segmentation, normalization and encoding steps, and the second one, do the matching step.

3.5. Technical parameters

In addition to commands and inputs/outputs information, the configuration contains some technical parameters which are explained below.

3.5.1. Diameters

The maximum and minimum diameters are used to improve the segmentation step. It constraints the search of pupil and iris to a known range of diameters. The first effect is to reduce the errors of segmentation. The second effect is to compute faster. Thus it is recommended to roughly survey pupil and iris sizes before segmenting, and get an a priori knowledge of object sizes for one set of images. For example, we set on ICE-2005 database the following ranges :

	minimum diameter	**maximum diameter**
pupil	50 pixels	160 pixels
iris	160 pixels	300 pixels

Table 3: Example of object sizes set for ICE-2005 database

3.5.2. Size of normalized image

The width and height of the normalized image can be changed if necessary. Default values are 512x64 pixels.

3.5.3. Gabor filters

The location of Gabor filters is indicated in the configuration files. The gabor filters are stored in a textfile this way:

- the filter coefficients are stored as real numbers
- for complex filters (such as Gabor), the real part and imaginary part are considered as 2 independent filters
- the first line indicates the number of filters (if there are 3 complex filters, the first line indicates “6”, because there are 3 real-part filters and 3 imaginary-part filters)
- for each filter, the filter size (rows, columns) preceed the coefficients.

The figure 9 shows the beginning of such a textfile. There are 6 filters. The first and second filters are 9-by-15 matrices (they correspond to the real part and imaginary part of a 9-by-15 Gabor filter). The third filter is a 9-by-27 matrix (its coefficients are not all visible). The other filters are not visible in the screenshot.

```
6

9   15

-0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
-0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
-0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
-0.50   -0.50   -0.50   -0.50   -0.50   1.00   1.00   1.00   1.00   1.00   -0.50   -0.50   -0.50   -0.50   -0.50
-0.50   -0.50   -0.50   -0.50   -0.50   1.00   1.00   1.00   1.00   1.00   -0.50   -0.50   -0.50   -0.50   -0.50
-0.50   -0.50   -0.50   -0.50   -0.50   1.00   1.00   1.00   1.00   1.00   -0.50   -0.50   -0.50   -0.50   -0.50
-0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
-0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
-0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25

9   15

0.25   0.25   0.25   0.25   0.25   -0.50   -0.50   0.00   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
0.25   0.25   0.25   0.25   0.25   -0.50   -0.50   0.00   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
0.25   0.25   0.25   0.25   0.25   -0.50   -0.50   0.00   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
0.50   0.50   0.50   0.50   0.50   -1.00   -1.00   0.00   1.00   1.00   -0.50   -0.50   -0.50   -0.50   -0.50
0.50   0.50   0.50   0.50   0.50   -1.00   -1.00   0.00   1.00   1.00   -0.50   -0.50   -0.50   -0.50   -0.50
0.50   0.50   0.50   0.50   0.50   -1.00   -1.00   0.00   1.00   1.00   -0.50   -0.50   -0.50   -0.50   -0.50
0.25   0.25   0.25   0.25   0.25   -0.50   -0.50   0.00   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
0.25   0.25   0.25   0.25   0.25   -0.50   -0.50   0.00   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25
0.25   0.25   0.25   0.25   0.25   -0.50   -0.50   0.00   0.50   0.50   -0.25   -0.25   -0.25   -0.25   -0.25

9   27

-0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   -0.2
-0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   -0.2
-0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   -0.2
-0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   -0.5
-0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   -0.5
-0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   -0.50   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   1.00   -0.5
-0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   -0.25   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   0.50   -0.2
```

Figure 9: Screeshot of the textfile containing the Gabor filters.

3.5.4. Application points

The location of the application points is indicated in the configuration file. The application points are stored in a textfile this way:

- the first row indicates the number of application points
- the coordinates of each point follow (row first, column second)

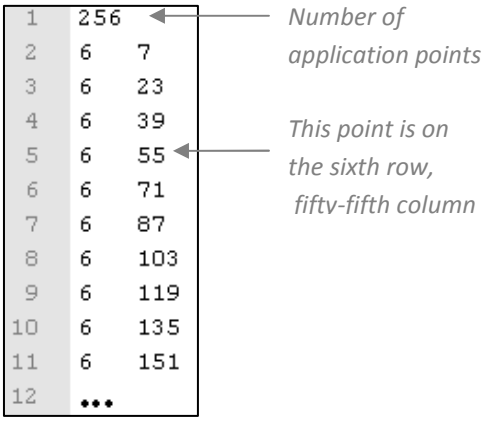

Figure 10: Screenshot of the textfile containing the application points.

D. Description of the algorithms

1. Segmentation

The pupil localization is the first step of the segmentation step. In Osiris v4.1, two criteria are used to find the pupil:

- The sum of pixel values inside the disk-shaped pupil should be near 0.
- The gradient $\overrightarrow{G_p}$ located at point P on the pupil contour is aligned with the radius $\overrightarrow{OP}$, O being the pupil center. Thus the sum of angles θ_p between $\overrightarrow{G_p}$ and $\overrightarrow{OP}$ should be near 0.

In order to speed up the search, the image is downsampled and both criteria are rewritten in the form of filters. Suppose that the pupil radius *r* is known, and we are searching for the center. The first criterion corresponds to finding the minimum in the image filtered by a disk-shaped kernel of size *r*-by-*r*, as illustrated in figure 10. Such filtering is equivalent to summing all the pixels in a disk-shaped neighbourhood. A normalization is applied in order to range the resulting values between 0 and 1, and the result is inverted so that 0 corresponds to a good pupil candidate, and 1 to a bad pupil candidate.

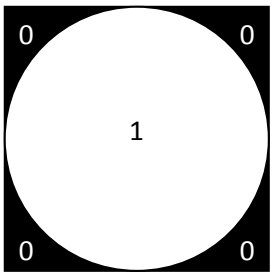

Figure 11: Disk-shaped kernel used in criterion 1.

The second criterion can be rewritten as :

$$\frac{1}{N}\sum_{p=1}^{N}\cos\ \theta_p = 1 \qquad \text{, since } \theta_p = 0$$

$$\frac{1}{N}\sum_{p=1}^{N}\frac{\overrightarrow{G_p}\bullet\overrightarrow{OP}}{\left\|\overrightarrow{G_p}\right\|.\left\|\overrightarrow{OP}\right\|} = 1 \qquad \text{, using the scalar product}$$

$$\frac{1}{N}\sum_{p=1}^{N}\left(X_{G_p}X_{OP} + Y_{G_p}Y_{OP}\right) = 1 \qquad \text{, assuming unit vectors}$$

Where N is the number of points in the ring-shaped neighborhood (arbitrary width). The last line can be seen as the sum of two filtering operations (horizontal and vertical). As $\overrightarrow{OP}$ is a unit vector, X_{OP} and Y_{OP} are easily deduced:

$$X_{op} = \frac{x}{\sqrt{x^2+y^2}} \qquad Y_{op} = \frac{y}{\sqrt{x^2+y^2}}$$

where x and y are the coordinates of point P relatively to center O

In practice, the second criterion is achieved by:

1. Computing horizontal and vertical gradient maps X_{G_p} and Y_{G_p} of the image, using the Sobel operator ;
2. Building two ring-shaped kernels of size r-by-r: X_{OP} and Y_{OP}.
3. Filtering X_{G_p} by X_{OP} and filtering Y_{G_p} by Y_{OP}.
4. Finding the location of the maximum values in the sum of both filterings.

This second criterion leads to values between 0 (gradients on contour are not radial => bad pupil candidate) and 1 (good pupil candidate), the same way as for the first criterion. Thus both criterion are simply added, and the maximum indicates the position of the pupil center. As the pupil radius r is not known *a priori*, all this procedure is repeated on varying radius values r, and the overall maximum corresponds to the complete pupil center (O, r).

In order to retrieve iris contours, the Viterbi algorithm is applied on the gradient map of images processed by anisotropic smoothing. The Viterbi algorithm is exploited at two resolutions: at a high resolution, it allows finding precise contours, while at a low resolution, coarse contours that improve the accuracy of the normalization circles are retrieved. This method is described in the paper: "The Viterbi algorithm at different resolution for enhanced iris segmentation" [3].

Notice that in this version of Osiris (version 4.1), we don't use circles for the normalization as in [3] but we use the coarse contours as explained in the next section. And the accurate contours are used for generating the masks.

2. Normalization

The normalization is based on Daugman's rubber-sheet model. It transforms the area of the iris delimited by the two iris and pupil contours detected in the segmentation into a scale-invariant and pupil-dilation-invariant band.

The coarse contours detected by the Viterbi algorithm are used for the normalisation. This coarse contour select the least number of noisy gradients points as possible and the sampling φ is not uniform. In the present version, we use for the normalisation a non-parametric shape instead of circles as in [3] (see Figure 14).

Let W and H be respectively the width and height of the desired normalized image. Following the Daugman rubber sheet approach, we compute a regular sampling of angles θ_k where k ranges from 0 to W, so that $\theta_0 = 0$ and $\theta_W = 2\pi$:

$$\theta_k = \frac{k}{W} 2\pi \quad , \quad k \in [0, W]$$

Let (x_p, y_p, φ_p) and (x_i, y_i, φ_i) be the coordinates of a point of pupil coarse and iris coarse contours respectively where (x, y) are the x-coordinate and y-coordinate of the radius relatively to the estimated center and φ the angle of the non-regular sampling.

Then we estimate the point $\left(X_k^p, Y_k^p\right)$ which corresponds to the θ_k (Figure 12). To do this, we interpolate the two nearest points j and $j+1$ to θ_k of the coarse contour as follow:

$$X_k^p = (1-\alpha) * x_p^j + \alpha * x_p^{j+1}$$
$$Y_k^p = (1-\alpha) * y_p^j + \alpha * y_p^{j+1}$$

with:

$$\alpha = \frac{\theta_k - \varphi_p^j}{\varphi_p^{j+1} - \varphi_p^j}$$

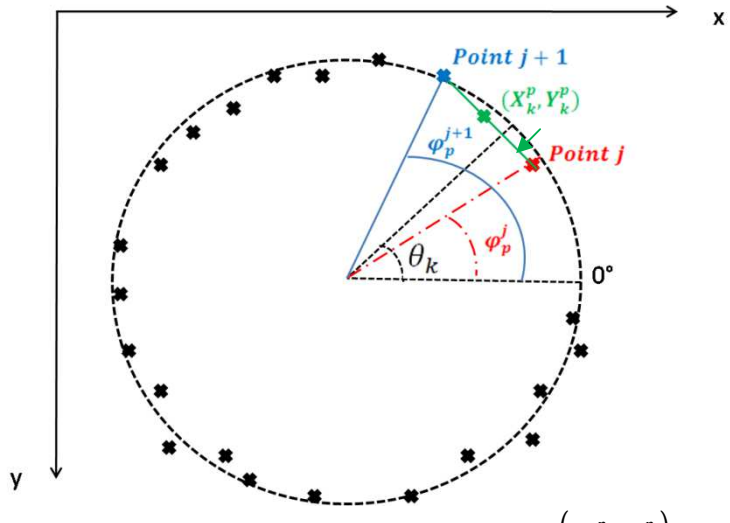

Figure 12: Compute the coordinates of the point $\left(X_k^p, Y_k^p\right)$

In a similar way, we find the point $\left(X_k^i, Y_k^i\right)$ of the iris contour. Let S be the segment formed by $\left(X_k^p, Y_k^p\right)$ and $\left(X_k^i, Y_k^i\right)$. Please note that pupil and iris centers are not necessarily the same points, thus the angle between S and the horizontal line is not necessarily equal to θ_k (Figure13).

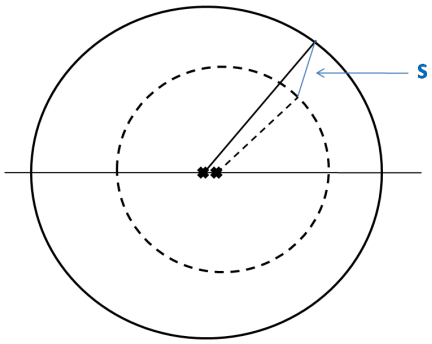

Figure 13: S the segment formed by (X_k^p, Y_k^p) and (X_k^i, Y_k^i). Note: The coarse points are circular in order to simplify the figure.

The segment S is then rescaled so that it fits with the height H of the normalized image. On the normalized image, the pixel on h^{th} row and k^{th} column will take the same value as the pixel located at $\left(x_{k,h}, y_{k,h}\right)$ on the original image :

$$\begin{aligned} x_{k,h} &= \left(1-\frac{h}{H}\right)X_k^p + \frac{h}{H}X_k^i \\ y_{k,h} &= \left(1-\frac{h}{H}\right)Y_k^p + \frac{h}{H}Y_k^i \end{aligned} \quad , \quad h \in [0, H]$$

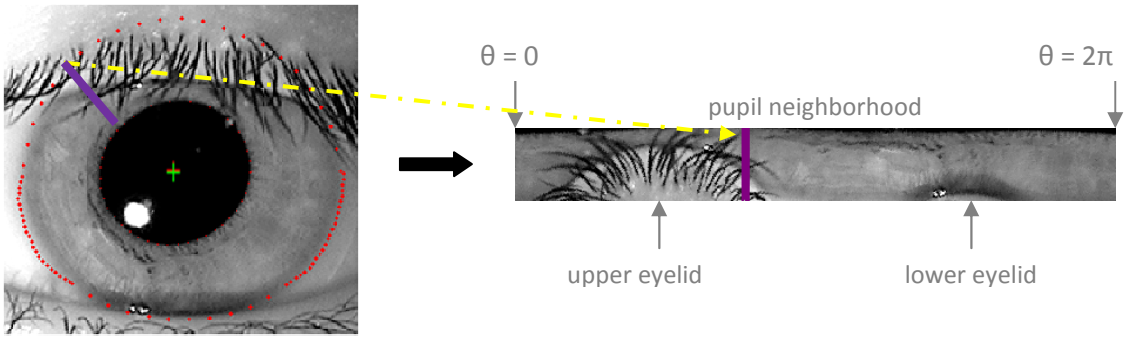

Figure 14: The original image (left) is normalized into a scale-invariant band (right) according to rubber-sheet method.

3. Encoding

The encoding consists in extracting features that characterize the iris texture. For this purpose, Osiris v4.1 uses three complex Gabor filters. The information of texture is carried by the phase of the Gabor filter, which is encoded on two bits as illustrated in figure 15. This is equivalent to know whether the real part and the imaginary part are positive or negative. Thus

Osiris considers each complex Gabor filter as two independent real filters: real part and imaginary part. That is why it is often spoken about 6 filters rather than 3 filters.

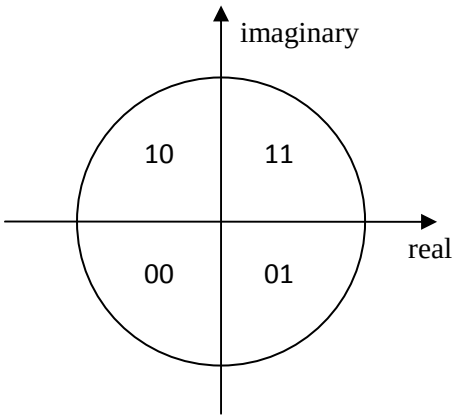

Figure 15: The phase of Gabor is encoded on two bits.

The processing is quite simple: filter the normalized image by each Gabor filter, and threshold the result relatively to zero in order to form 6 binary images. In Osiris v4.1 the whole binary images are saved as the "iriscode", even if not all the pixels will be considered during the matching. This is done because the application points, indicating which pixels are to be considered for matching, are not known before the matching step. It allows changing the application points without recomputing the Gabor phases.

4. Matching

Two eyes are compared using the Hamming distance between their iris codes. A matrix of application points is used to indicate which pixels are to be considered during the matching. Moreover, some binary mask can be used to ignore noisy pixels. In Osiris v4.1, it is achieved this way: first a total mask is built: it is the result of logical *and* between the matrix of application points and the masks ; then the a logical *xor* between both iris codes is computed for the application points that are not masked (Figure 16).

A. Experiments:

To test the performance of the new version of the OSIRIS reference system, we used the rights iris in the database ICE 2005. There were totally 7704 intra-class comparisons. For the inter-class, we selected randomly 219231 comparisons. These scripts files can be found in the directory Iris_Osiris_v4.1/scripts/Matching/. Table 4 shows the comparison between the performance of the version 2.0 and 4.2 of Osiris.

	Osiris_v2.0	Osiris_v4.1
EER	5.14%	1.10%
FRR@FAR=0.1%	16.03%	2.01%

Table 4: Performance of OSIRIS_V2.0 and OSIRIS_V4.1

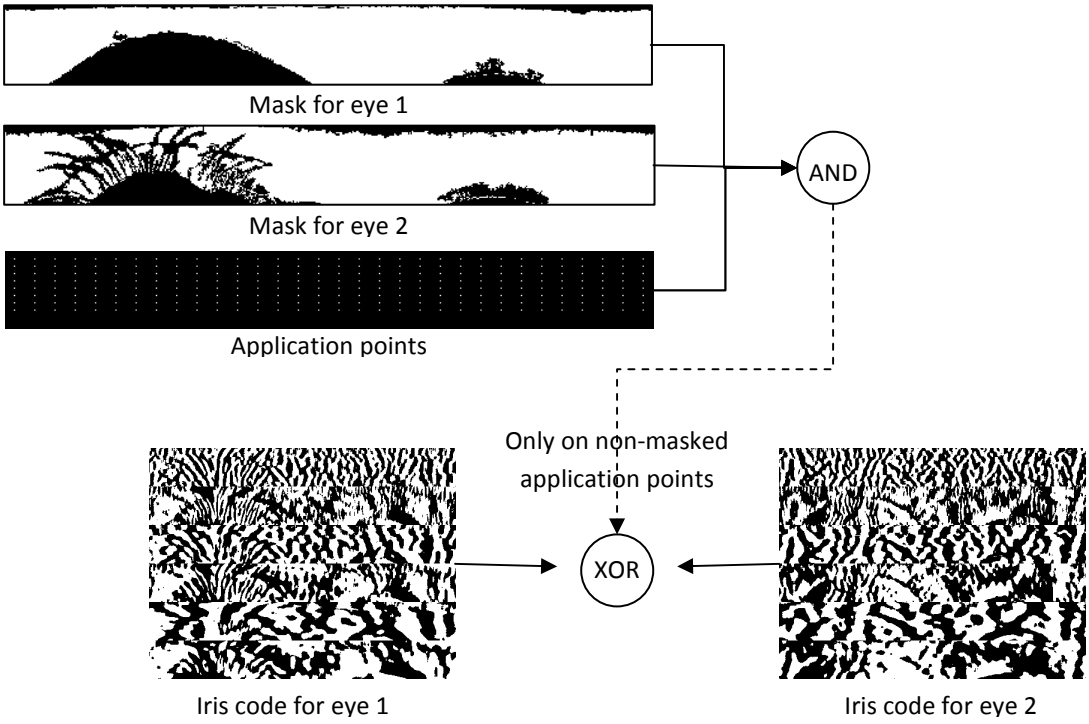

Figure 16: The XOR operator is used to compare two iris codes. Only the non-masekd application points are considered during the XOR operation

We get an EER equal to 1.10% and the FRR at FAR of 0.1% is equal to 2.01%. This new version allows a big amelioration compared to version 2.0 which gave an EER of 5.14% and the FRR at FAR of 0.1 % of 16.03% on the same database with the same protocol. The DET curves are plotted in Figure 17.

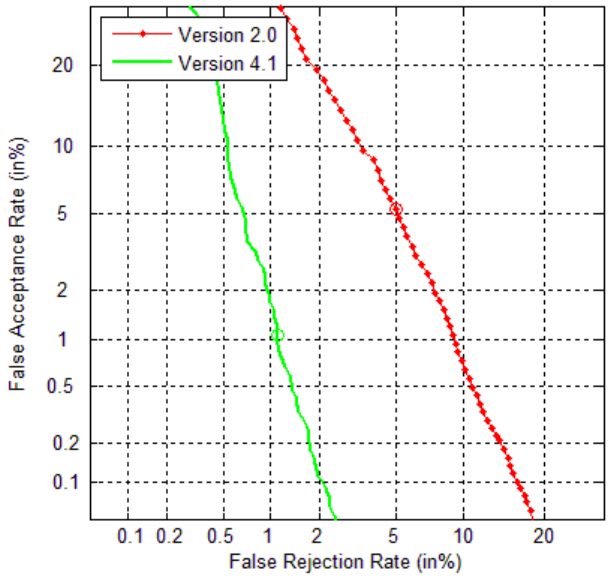

Figure 17: Det curves of reference systems: comparison between the OSIRIS version 2.0 and 4.1.

Reference

[1] Biosecure project: http://biosecure.it-sudparis.eu/AB/

[2] J.G. Daugman, High con_dence visual recognition of persons by a test of statistical independence, IEEE Trans. Patt. Anal. Mach. Intell., 15(11) :1148-1161, 1993.

[3] G. Sutra, S. Garcia-Salicetti, and B. Dorizzi, "The Viterbi algorithm at different resolution for enhanced iris segmentation". 5th IAPR International Conference on Biometrics, p. 310-316, 2012.

[4] ICE2005 Database: http ://iris.nist.gov/ice/