USIT – University of Salzburg Iris Toolkit

Christian Rathgeb, Andreas Uhl, Peter Wild

Multimedia Signal Processing and Security Lab (WaveLab)
University of Salzburg – Austria
Department of Computer Sciences

Version 1.0.1

November 2012

Contents

1 Introduction

2 Installation

3 List of Algorithms

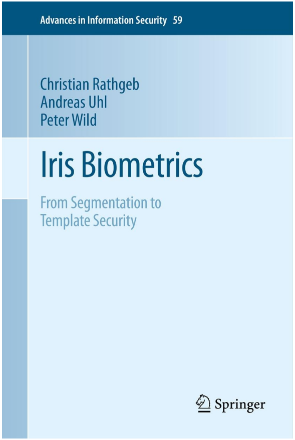
4 Usage

5 Tutorial

Introduction to USIT I

- **USIT - University of Salzburg Iris Toolkit** is a Windows/Linux software package for iris recognition, made publicly available together with the book:

  [1] C. Rathgeb, A. Uhl, and P. Wild. Iris Recognition: From Segmentation to Template Security, *In Advances in Information Security 59*. Springer, New York, 2012. ISBN: 978-1461455707



Advances in Information Security 59

Christian Rathgeb
Andreas Uhl
Peter Wild

Iris Biometrics

From Segmentation to Template Security

Springer

Introduction to USIT II

- **Who may use USIT?**
  Anyone who likes to! We just ask you to:

1. obey the BSD license agreement
   `http://opensource.org/licenses/bsd-license.php`

2. and all the technical reports and papers that report experimental results from this software should provide an acknowledgement and reference to [1]

- **How to acknowledge USIT?**
  The authors suggest the following acknowledgement:

  “USIT – University of Salzburg Iris Toolkit v1.0 [1]”

Introduction to USIT III

- **What is provided with USIT?**
  The software package includes algorithms for iris preprocessing, feature extraction and comparison. Input and output relies on files.

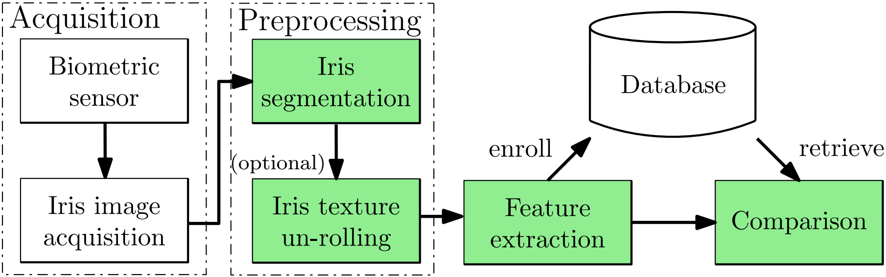


Figure: Typical iris processing chain (provided modules marked green)

Installation

- **What do I need to run USIT?**
  Windows/Linux platform with OpenCV and Boost libs installed

- **Where can I find more information about OpenCV/Boost?**
  `http://opencv.willowgarage.com/`
  `http://www.boost.org/`

- **OpenCV-Download/Install Guide**:
  `http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/`
  `http://opencv.willowgarage.com/wiki/InstallGuide`

- **Boost-Download/Install Guide**:
  `http://www.boost.org/users/download/`
  `http://www.boost.org/doc/libs/release/more/getting_started/index.html`

List of Algorithms I

Iris Segmentation:

1. Contrast-adjusted Hough Transform
2. Weighted Adaptive Hough and Ellipsopolar Transform

Iris Feature Extraction:

1. 1D-LogGabor Feature Extraction
2. Algorithm of Ma *et al.* (re-implementation)
3. Algorithm of Ko *et al.* (re-implementation)
4. Algorithm of Rathgeb and Uhl
5. Context-based Feature Extraction
6. Algorithm of Monro *et al.* (re-implementation)

List of Algorithms II

Iris Biometric Comparators:

1. Hamming Distance-based Comparator
2. Context-based Comparator
3. Comparator of Ko *et al.* (re-implementation)
4. Comparator of Monro *et al.* (re-implementation)

Face/Face-part Detection:

1. Gaussian Face and Face-part Classifier Fusion

Further information on the implemented algorithms can be found in according source files and in [1].

Usage

- USIT is based on easy-to-use command line tools.
- Listed algorithms coincide with the list of provided source files, i.e. each algorithm represents a stand-alone application.
- Focusing on feature extraction the following algorithms require a special non-HD-based comparator:

  1. Algorithm of Ko *et al.* → comparator: `koc.cp`
  2. Context-based algorithm → comparator: `cbc.cp`
  3. Algorithm of Monro *et al.* → comparator: `dctc.cp`

Usage - Segmentation I

Contrast-adjusted Hough Transform:

- Source file(s):
  `caht.cpp`
- Segmentation:

  ```
  ./caht -i eye_image -o texture -m mask -s width
  height
  ```
- Examples:

  ```
  ./caht -i file.tiff -o texture.png -s 512 64 -e
  ./caht.exe -i *.tiff -o ?1_texture.png -m
  ?1_mask.png -s 512 64 -e
  ```

Usage - Segmentation II

Weighted Adaptive Hough and Ellipsopolar Transform:

- Source file(s):
  `wahet.cpp`
- Segmentation:
  `./wahet -i eye_image -o texture -m mask -s width height`
- Examples:

  ```
  ./wahet -i file.tiff -o texture.png -s 512 64 -e
  ./wahet.exe -i *.tiff -o ?1_texture.png -m ?1_mask.png -s 512 64 -e
  ```

Usage - Feature Extraction and Comparison I

1D-LogGabor Feature Extraction:

- Source file(s):

  ```
  lg.cpp
  ```

- Feature extraction:

  ```
  ./lg -i iris_texture -o iris_code
  ```

- Comparison:

  ```
  ./hd -i iris_code_1 iris_code_2 -o log_file
  ```

- Examples:

  ```
  ./lg -i texture_1.tiff -o code_1.png
  ./hd -i code_1.png code_2.png -o result.txt
  ```

Usage - Feature Extraction and Comparison II

Hamming Distance-based Comparator:

- Source file(s):

  ```
  hd.cpp
  ```

- Comparison:

  ```
  ./hd -i iris_code_1 iris_code_2 -s shift_min
  shift_max -m mask_file_1 mask_file_2 -a algorithm
  -n form_bit to_bit -o log_file
  ```

- Examples:

  ```
  ./hd -i code_1.png code_2.png -o result.txt
  ./hd -i code_1.png code_2.png -m mask_1.png
  mask_2.png
  ./hd -i code_1.png code_2.png -s -8 8 -n 0 100
  ./hd -i code_1.png code_2.png -s -8 8 -a ssf
  ```

Usage - Feature Extraction and Comparison III

Algorithm of Ma *et al.* (re-implementation):

- Source file(s):

  ```
  qsw.cpp
  ```

- Feature extraction:

  ```
  ./qsw -i iris_texture -o iris_code
  ```

- Comparison:

  ```
  ./hd -i iris_code_1 iris_code_2 -o log_file
  ```

- Examples:

  ```
  ./qsw -i texture_1.tiff -o code_1.png
  ./hd -i code_1.png code_2.png -o result.txt
  ```

Further reading:

- Ma, L., Tan, T., Wang, Y., Zhang, D.: Personal identification based on iris texture analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(12), 1519 - 1533 (2003).

Usage - Feature Extraction and Comparison IV

Algorithm of Ko *et al.* (re-implementation):

- Source file(s):

  `ko.cpp, koc.cpp`
- Feature extraction:

  ```
  ./ko -i iris_texture -o iris_code
  ```
- Comparison:

  ```
  ./koc -i iris_code_1 iris_code_2 -o log_file
  ```
- Examples:

  ```
  ./ko -i texture_1.tiff -o code_1.png
  ./koc -i code_1.png code_2.png -o result.txt
  ```

Further reading:

- Ko, J.G., Gil, Y.H., Yoo, J.H., Chung, K.I.: *A novel and efficient feature extraction method for iris recognition*. ETRI Journal 29(3), 399401 (2007)

Usage - Feature Extraction and Comparison V

Algorithm of Rathgeb and Uhl:

- Source file(s):

  ```
  cr.cpp
  ```

- Feature extraction:

  ```
  ./cr -i iris_texture -o iris_code
  ```

- Comparison:

  ```
  ./hd -i iris_code_1 iris_code_2 -o log_file
  ```

- Examples:

  ```
  ./cr -i texture_1.tiff -o code_1.png
  ./hd -i code_1.png code_2.png -o result.txt
  ```

Usage - Feature Extraction and Comparison VI

Context-based Iris Recognition:

- Source file(s):
  `cb.cpp, cbc.cpp`
- Feature extraction:
  ```
  ./cb -i iris_texture -o iris_code
  ```
- Comparison:
  ```
  ./cbc -i iris_code_1 iris_code_2 -o log_file
  ```
- Examples:
  ```
  ./cb -i texture_1.tiff -o code_1.png
  ./cbc -i code_1.png code_2.png -o result.txt
  ```

Usage - Feature Extraction and Comparison VII

Algorithm of Monro *et al.* (re-implementation):

- Source file(s):
  ```
  dct.cpp, dctc.cpp
  ```
- Feature extraction:
  ```
  ./dct -i iris_texture -o iris_code
  ```
- Comparison:
  ```
  ./dctc -i iris_code_1 iris_code_2 -o log_file
  ```
- Examples:
  ```
  ./dct -i texture_1.tiff -o code_1.png
  ./dctc -i code_1.png code_2.png -o result.txt
  ```

Further reading:

- Monro D. M., Rakshit S., Zhang D.: DCT-based Iris Recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence.* 29(4):586-595 (2007)

Usage - Face/Face-part Detection

Gaussian Face and Face-part Classifier Fusion:

- Source file(s):
  `gfcf.cpp`
- Detection:
  ```
  ./gfcf -i picture -o face left_eye right_eye
  ```
- Examples:
  ```
  ./gfcf -i *.tiff -o ?1_face.png ?1_eyeleft.png
  ?1_eyeright.png -q -t
  ```
- **Attention**: This program needs file access (within the program directory) to `haarcascade_frontalface_default.xml`, `haarcascade_eye_tree_eyeglasses.xml`, `haarcascade_mcs_eyepair_big.xml`, `haarcascade_mcs_nose.xml` shipped with OpenCV.

Usage - Hamming Distance-based Evaluation

Performance of Hamming Distance-based verification of iris codes:

- Source file(s):

  ```
  hdverify.cpp
  ```

- Evaluation:

  ```
  ./hdverify -i files class -s min max -r rocfile
  ```

- Examples:

  ```
  ./hdverify -i files/class*/*.tiff ?1 -t -s -7 7
  -r roc.dat
  ```

Usage - Binary mask comparison

Performance of segmentation algorithms wrt. binary noise masks:

- Source file(s):

  ```
  maskcmp.cpp
  ```

- Evaluation:

  ```
  ./maskcmp -i file1 file2
  ```

- Examples:

  ```
  ./maskcmp -i *_mask1.tiff ?1_mask2.tiff -o
  output.dat
  ```

Usage - Wildcards

- For batch processing and usage of wildcards, in Linux shells be sure to turn off globbing (`set -o noglob`).
- In filenames, you can use `*` as a wildcard character to match any character sequence.
- If `*` is used as a wildcard (one or multiple times), `?1, ?2, ...` in other related attributes refer to the contents of the n-th `*`.
- **Example**: `./hdverify -i files/class*/*.tiff ?1 -t -s -7 7 -r roc.dat` processes each of the directories `files/class1, files/class2, ... files/class999` and each file ending in `.tiff` in each of these directories. As class label, the `?1` refers to the contents of the matched character sequence, i.e. `1` for `files/class1`, `2` for `files/class2`, etc.

Usage - Quiet Mode and Time Progress

Quiet Mode:

- Executions can be run in quiet mode with option `-q`
- Example:

```
./lg -i texture_1.tiff -o code_1.png -q
```

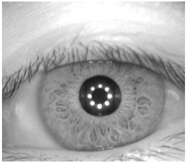
Time Progress:

- The time progress of executions can be displayed with option `-t`
- Example:

```
./hd -i ⋆.png ⋆.png -s -7 7 -o compare.txt -q -t
```

Tutorial I

- The iris recognition processing chain starts with the sample input of an eye image (on te right)

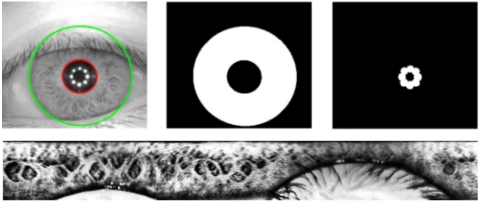


- In order to segment the image use one of the provided segmentation algorithms, e.g. WAHET

- Example:

```
./wahet -i eye.jpg -sr seg.jpg -bm bmask.jpg -o
tex.jpg -e -lt 2
```

- Output images:



Tutorial II

- Once textures are generated feature vectors (iris-codes) can be generated, e.g. by applying the 1D-Log Gabor feature extraction.

- Example:

```
./lg -i tex_1.jpg -o code_1.png
```

- Finally generated feature vectors can be compared, either by using the `hd`-comparator or algorithm specific comparators.

- Example:

```
./hd -i code_1.png code_2.png -s -8 8
```

- Sample output:

```
hd(code_1.png,code_2.png) = 0.3856217
```