Distribution Overview

This document describes the files, algorithms and test scripts that complement the MBGC data distribution. The MBGC uses the standard file structures and calling conventions introduced and used in the FRGC, FRVT and ICE programs. These include a standard calling signature for executables, signature sets (sigsets) for specifying target and query image sets, similarity matrices for recording match scores and mask matrices for providing ground truth information. These standard structures have proven to be extremely robust and flexible across several biometric modalities.

Briefly, each of the MBGC challenge problems is defined by

1. One or more target sigsets that list a set of enrolled images;
2. One or more query sigsets that list a set of images to be recognized; and
3. One or more mask matrixes that specify which match pairs are matches, non-matches and don't cares.

In MBGC, experiments consist of multiple targets such that each subject only appear in a given target once as opposed to executing one large $n \times m$ experiment with one target and one query set. This allows for normalized matching. The specific combination of targets, queries and mask matrixes for each experiment is listed at the end of this document in Table E. Further the mask matrixes only consider true imposter non-matches (e.g. non-matches where the query image is not in the target set). Non-matching comparisons between query subjects who are in the target set are ignored (don't cares).

This document also describes a recommended directory structure for storing the various file types distributed with the MBGC. While participants are free to utilize any file structure they choose, the suggested structure is convenient because all of the test scripts and examples assume that files locations adhere to this structure. Finally, we the last page of the document for helpful notes to Unix and MS Windows users.

Directory Structure

While there is no required directory structure in MBGC, the sample applications and run scripts assume the directory structure shown in Figure A. The remainder of this document assumes that participants have created and organized the appropriate directories as illustrated below. All experiments (and thus executables) will be instantiated from the top level (*mbgc/*) directory. Executables to score the similarity matrices are in the *mbgc/bin/* directory. There should be a subfolder for each challenge problem below the *mbgc/* directory. Executables for a particular challenge problem should be installed in the The *mbgc/*{*challengeproblem*}/*bin/* directory in that problems subfolder. The *mbgc/*{*challengeproblem*}/*bin/* directory also contains Unix shell scripts and MS Windows batch files which can be used to execute the experiments. Parameter files needed by the algorithms are in the *mbgc/*{*challengeproblem*}/*param/* directory. Signature sets, which list input images for the experiments, are provided in the *mbgc/*{*challengeproblem*}/*sigsets/* directory. Mask matrices, which provide the ground truth (answer key), are provide in the *mbgc/*{*challengeproblem*}/*maskmatrices/* directory. All output files (similarity matrices, signature sets, quality score files) generated during experiment execution will be written to the *mbgc/*{*challengeproblem*}/*output/* directory by the scripts. Please ensure that you have write permission to the *mbgc/*{*challengeproblem*}/*output/* directory.

The downloaded images can be stored in any the appropriate subdirectory. We will refer to this directory as /*data/* in the remainder of this document for clarity. However, you can give this directory any name you chose -simply change the value of the variable **DATA_DIR** in the run scripts to point to your data directory.

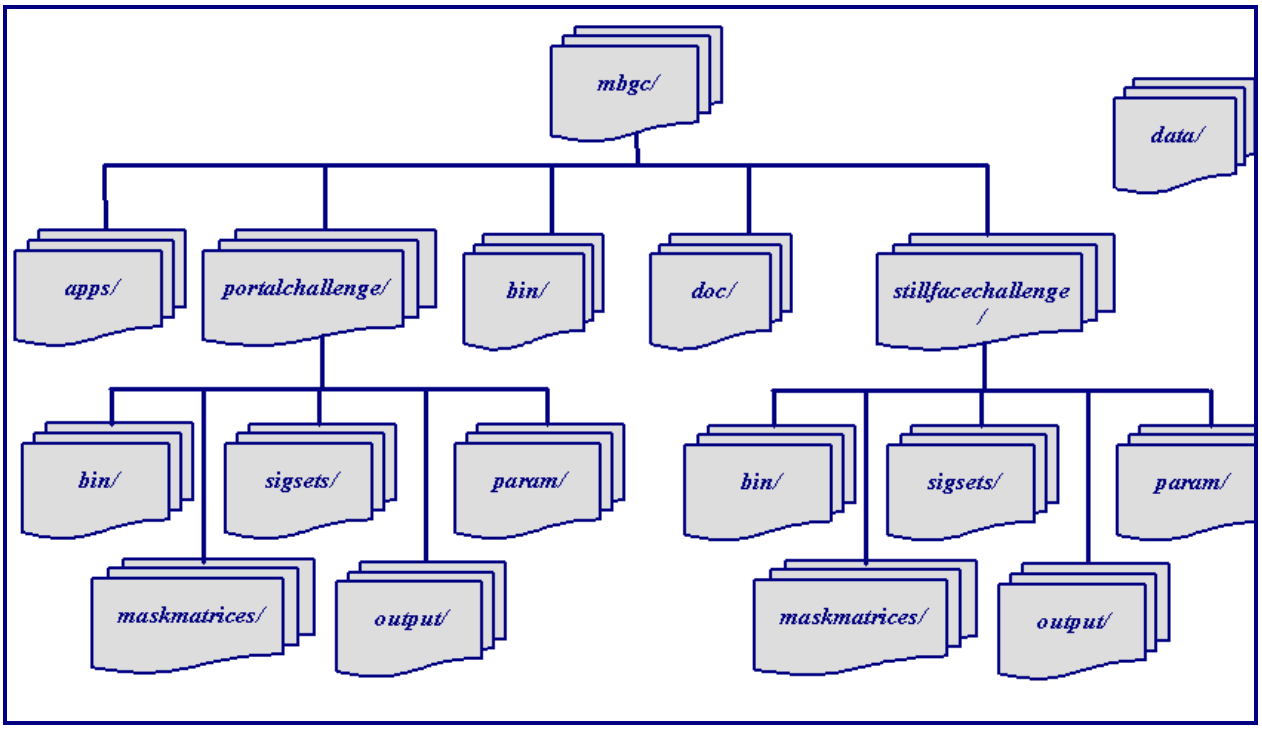

Figure A MBGC directory structure.

Executable Calling Signature

It is assumed that all MBGC executables will have a common calling signature. This signature was used with great success in the FRGC, FRVT and ICE evaluations. It will also be employed if there is an independent government evaluation at the conclusion of the MBGC. Specifically, the executable calling signature is:

executable_name ***parameter_file image_directory target_sigset query_sigset similarity_file quality_target_sigset quality_query_sigset***

Table A describes the arguments used above in the executable calling signatures. It is important to note that all filenames will be relative to the working directory (i.e. *mbgc/*)directory.

Parameter name	Type	Format	Description
parameter_file	Input	XML	An XML document that specifies experiment description information, configuration parameters and the name of metadata files.
image_directory	Input	string	The relative path to the image (data) directory
target_sigset	Input	Sigset	The name of the target signature set. This document will contain a list of the target images.

Parameter name	Type	Format	Description
query_sigset	Input	Sigset	The name of the target signature set. This document will contain a list of the query images.
similarity_file	Input/ Output	Similarity Matrix	The name of the similarity (or distance) file. This is the primary output data structure.
quality_target_sigset	Output (optional)	Sigset	The name of the target sigset to which quality scores should be written. This file should be a copy of the input target sigset with quality scores added. Ignore this parameter if your executable does not compute quality scores.
quality_query_sigset	Output (optional)	Sigset	The name of the query sigset to which quality scores should be written. This file should be a copy of the input query sigset with quality scores added. Ignore this parameter if your executable does not compute quality scores.

Table A Description of the executable arguments

MBGC/Bin Directory

The *mbgc/bin/* directory has two applications. The FRVT2006MatchingExample confirms the setup of your directory structure and illustrates have to parse and write the standard file types. The ROCFromMultiSims creates ROC that depicts the accuracy of your algorithm. Follow the following steps to run these scripts:

1. Compile the matrix, sigset, utilities and xpath libraries in the *mbgc/apps/btools/c++/* directory. There are Unix makefiles and Visual Studio projects for making these libraries.
2. Compile the FRVT2006MatchingExample executable under the *mbgc/apps/btools/* directory. Place the resulting executables in the *mbgc/bin/* directory.
3. Compile the ROCFromMultiSims executable under the *mbgc/apps/btools/* directory. Place the resulting executables in the *mbgc/bin/* directory.
4. Set the value of the variable **DATA_DIR** in the *mbgc/(challengeproblem}/bin/* directory. The variable's value should be the path to your top level MBGC data directory.
5. Run the script from *mbgc/(challengeproblem}/* passing ../bin/FRVT2006MatchingExample as the argument.

The script will output the resulting similarity matrices and roc.txt files in the *mbgc/(challengeproblem}/output/* directory.

FRVT2006MatchingExample

The FRVT2006MatchingExample application is a contrived sample executable that adheres to the executable calling signature described above. It is in the *mbgc/apps/FRVT2006MatchingExample/* directory. While not useful as a recognition algorithm, it can be used to confirm that your directory structure is setup properly. When called, it will parse thru the input target and query signature sets and confirm that all of the required files can be read. It will also output a similarity matrix (and quality XML signature sets) of the appropriate dimension to the specified output filename. The source code for the FRVT2006MatchingExample application is also extremely valuable because it illustrates examples of parsing, creating and writing all of the standard structures discussed in the remainder of this document.

Follow these steps to use the FRVT2006MatchingExample:

1. Compile the source using the Unix makefile or Visual Studio project file in the *mbgc/apps/ FRVT2006MatchingExample/* directory.
2. Place the resulting executables in the *mbgc/bin/* directory.
3. Set the value of the variable **DATA_DIR** in the *mbgc/(challengeproblem}/bin/* directory. The variable's value should be the path to your top level MBGC data directory.
4. Run the desired test script in the *mbgc/(challengeproblem}/bin/* directory providing ../bin/FRVT2006MatchingExample as the argument to the script.

The FRVT2006MatchingExample executable can also be called from the command line using the executable calling signature:

../bin/FRVT2006MatchingExample p*aram/parameter_file.xml …./data/ target_sigset.xml query_sigset.xml similarity_file.mtx quality_target_sigset.xml quality_query_sigset.xml*

ROCFromMultiSims

The ROCFromMultiSims application is an analysis tool that computes FAR and FRR scores that can be graphed to produce a ROC graph. The algorithm takes as input a parameter file and a simple XML file that lists pairs of similarity matrices and mask matrices that define the

experiment. It outputs a text file that lists the FAR and FRR scores. The general calling signature is:

ROCFromMultiSims *parameter.xml roc_filename.txt matrix_list.xml*

For MBGC, a typical call might be:

../bin/ROCFromMultiSims *param/analysis.xml output/test_smrslls_vvnps_roc.txt param/test_smrslls_vvnps_matrices.xml*

Follow these steps to use the FRVT2006MatchingExample:

1. Compile the source using the Unix makefile or Visual Studio project file in the *mbgc/apps/ROCFromMultiSims/* directory.
2. Place the resulting executables in the *mbgc/bin/* directory.
3. Run the desired test script in the *mbgc/bin/* directory providing our algorithm name as the argument to the script.

bTools

bTools is a set of libraries for parsing, managing and writing all MBGC data structures including signature sets, similarity matrices, mask matrices, and XML parameter file. They are easily compiled using either the Unix makefiles or the Visual Studio projects under the individual folder in either the *mbgc/apps/btools/c++/* or the *apps/btools/java/* directories. The use of each individual library is detailed in the remainder of this document.

MBGC/{ChallengeProbem}/Bin Directory

The *mbgc/{challengeproblem}/bin/* directory has both MS Windows batch files and Unix Bash scripts for executing the experiments. Follow the following steps to run these scripts:

1. Compile the matrix, sigset, utilities and xpath libraries in the *mbgc/apps/btools/c++/* directory. There are Unix makefiles and Visual Studio projects for making these libraries.
2. Compile the ROCFromMultiSims executable under the *mbgc/apps/btools/* directory. Place the resulting executables in the *mbgc/bin/* directory.

3. Run the desired script from *mbgc/(challengeproblem}/* passing the name of the executable for your recognition algorithm as the argument.

The script will output the resulting similarity matrices and roc.txt files in the *mbgc/(challengeproblem}/output/* directory.

Data Directory

This document assumes that images are stored in seven directories below a main */data/* directory (see **Figure B**). The full path to this top-level data directory will be passed as an argument to the executable. Your algorithm should then append this string to each of the image file listed in the signature sets. For example, if the parameter *image_directory* is */home/username/data/* and the sigset refers to an image named '*NIR-face-vide/04233v1466.avi*', you should process the image in the file */home/username/data/NIR-face-video/04233v1466.avi*. Note: the value of the *image_directory* argument will be consistent with the underlying operating system. Thus, *d:\home\username\data* and *home/username/data/* would be provided for MS Windows and Linux operating systems respectively.

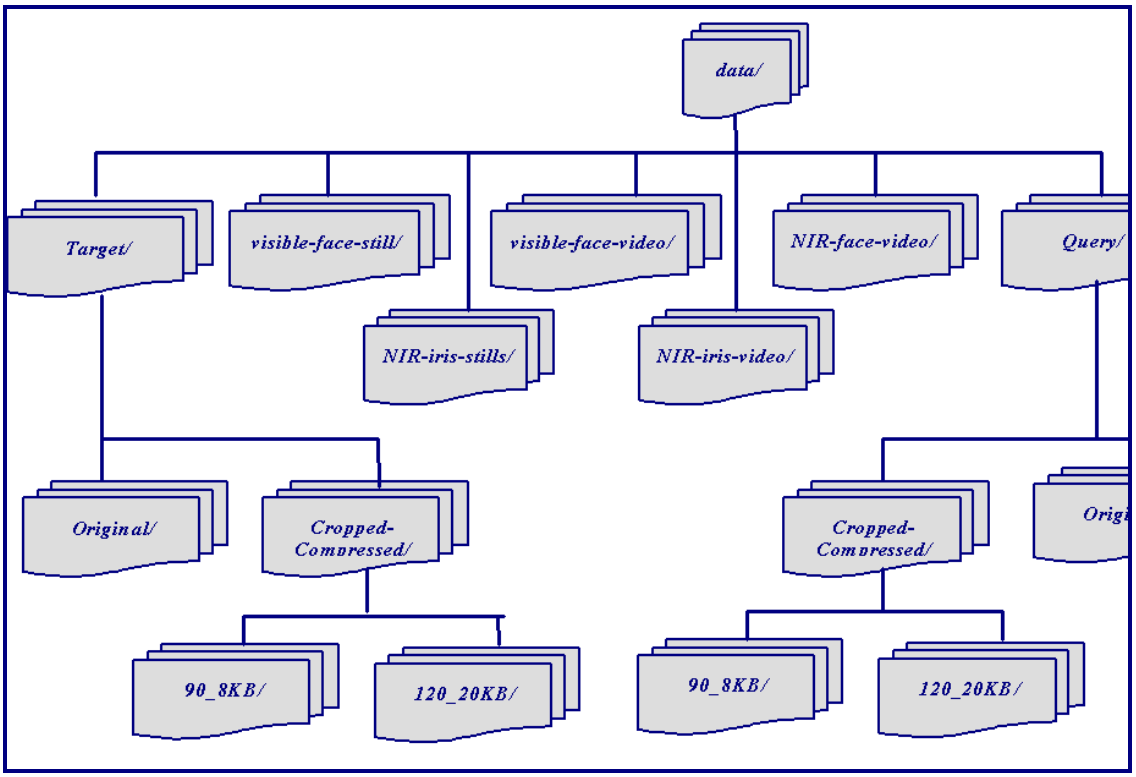

Figure B MBGC data directory structure

Parameter Files

Overview

Parameter files are XML documents that provide experiment description information and configuration values for executables. To simplify their processing, configuration values will always be specified via the values of attributes in the parameter file. We suggest that MBGC participants adhere to passing information to their algorithms via XML parameter files.

Structure

While there is not a set structure for parameter files, the structure will be similar to the example shown below in Figure C. In this example, elements are depicted by black text (capitalized), attributes are shown with blue text (lowercase) and attribute values have red text (uppercase and in quotes). The *Experiment* element is the outer element. It has three attributes named *name* with value "*StillIris_versus_NIRIris*", *type* with value "*Normalized*" and *feature_extraction_mode* with value "*Full*". The *Experiment* element has three child elements *Target*, *Query* and *Log_File*. Both the *Target* and *Query* elements have four attributes: *mode*, *capture*, *min_recordings* and *max_recordings*. The *LogFile* element has one attribute named *name* that specifies the name of the experiment log file.

```
<?xml version="1.0"?>
<Experiment name="StillIris_versus_NIRIris" type="Normalized" feature_extraction_mode="Full" >
  <Target mode="StillIris" capture="LG2200" min_recordings="6" max_recordings="6" />
  <Query mode="NIRIris" capture="Portal" min_recordings="2" max_recordings="2" />
  <LogFile name="StillIris_versus_NIRIris_log.txt" />
</Experiment>
```

Figure C Example of a parameter file

Attribute	Type	Allowed Values	Comments
name (Experiment)	String		The text of this string provides a name for the experiment. This attribute is not needed by users.
Type	enumerated	Normalized 1-many 1-1	The enumerated values of this attribute specify whether the target sets are normalized (only have one image of each subject, support 1-many matching or 1-1 matching.
feature_extraction_mode	enumerated	Partial Full	The enumerated values of this attribute specify whether ground truth data is provided to aid in feature extraction. "Full" indicates

Attribute	Type	Allowed Values	Comments
			that ground truth data is not provided. "Partial" indicates that ground truth is provided in the file denoted by the attribute *ground_truth*. The MBGC is only considering fully automatic algorithms.
Mode	enumerated	2D 3D	The enumerated values of this attribute specify whether the images are 2D or 3D face images.
capture	enumerated	Controlled Uncontrolled	The enumerated values of this attribute specify whether the images were captured under controlled or uncontrolled conditions.
min_recordings	integer	>0	This integer specifies the minimum number of images that will be associated with a signature.
max_recordings	integer	>0	This integer specifies the maximum number of images that will be associated with a signature.
Name (LogFile)	string	Filename	All logging information generated by the program should be written to this file in the output directory.

Table B Description of elements in the similarity header.

Parsing

Due to their simple structure, parameter files are readily parsed with any XML or XPath parser. Source implementation of C++ and Java classes for parsing parameter files are provided in the BEETools (Biometric Experimentation Environment) distribution in the *mbgc//apps/btools/c++/xpath/* and the *mbgc/apps/btools/java/xpath/* directories. These classes, which use the XPath parser, are available for modification by MBGC participants. Parsers will not be made available for other languages (e.g. Matlab). However, users should be able to easily create parsers in other languages using the C++ classes as a guide.

Writing

MBGC users will not be required to write (output) parameter files.

Signature Sets (Sigsets)

Overview

Signature sets (sigsets) are the primary input structure for MBGC. They are used to list the files in the target and/or query sets. They will also be the standard format used to output quality information.

Structure

The signature set document will provide a list of images. XML will be used because its hierarchical structure facilitates a flexible representation of the relationships between subjects, sessions, sensors and files. Specifically, the signature set will consist of a list of *Signature* (subjects) elements. Each *Signature* element will contain one or more *Presentation* child elements that correspond to capture sessions. Each *Presentation* element will contain one or more *Component* elements that correspond to a sensor. Lastly, each *Component* will have one or more *Data* elements that correspond to a file.

Figure D illustrates the general signature set structure with elements depicted by black text, attributes depicted by blue text (lowercase) and attribute values depicted by red text (uppercase and in quotes). This example has two **complex-biometric-signature** elements and thus represents two subjects. (Note: While **complex-presentations** grouped under a single **complex-biometric-signature** are guaranteed to correspond to the same subject, separate signatures may correspond to the same subject.) The signatures are named "*nd1S05288*" and "*nd1S05156*". Both signatures have a single **complex-presentation** (capture session) in the face modality labeled "*nd1S05288*" and "nd1S05156" respectively. The presentations each have two **presentation-component** elements that correspond to the two sensors in the portal. One component corresponds to NIR face video sensor and produces a video stream stored as a **data** (file) in the *avi* format. The other component corresponds to the HD camera which produces a visible face video stored in the *MPEG2-TS* format.

```
<?xml version="1.0" encoding="UTF-8"?>
<biometric-signature-set  xmlns="http://www.bee-biometrics.org/schemas/sigset/0.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.bee-biometrics.org/schemas/sigset/0.1
    http://www.bee-biometrics.org/schemas/sigset/0.1/general.xsd">

<complex-biometric-signature name="nd1S05288" >
    <complex-presentation name="nd1S05288" modality="Face" >
      <presentation-component name="NIR_Face" >
        <data file-name="NIR-face-video/05288v4.avi" file-format="avi" />
      </presentation-component>
      <presentation-component name="Visible_Face" >
        <data file-name="visible-face-video/05288v3.ts" file-format="MPEG2-TS" />
      </presentation-component>
    </complex-presentation>
  </complex-biometric-signature>
  <complex-biometric-signature name="nd1S05156" >
    <complex-presentation name="nd1S05156" modality="Face" >
      <presentation-component name="NIR_Face" >
        <data file-name="NIR-face-video/05156v333.avi" file-format="avi" />
      </presentation-component>
      <presentation-component name="Visible_Face" >
        <data file-name="visible-face-video/05156v331.ts" file-format="MPEG2-TS" />
      </presentation-component>
    </complex-presentation>
  </complex-biometric-signature>
</biometric-signature-set>
```

Figure D Example of a signature set.

Parsing

Signature sets are difficult to parse due to their rich hierarchical structure. Fortunately, C++ and Java classes for parsing similarity matrices are provided in the BEE (Biometric Experimentation Environment Toolset) Tools distribution in the *mbgc/apps/btools/c++/sigset/* and the *mbgc/apps/btools/java/sigset/* directories. We also provide examples for the use of these parsers. It is important to note that some of the BEE Tools examples assume the simplified sigset structure in which each signature has precisely one Presentation, each Presentation has precisely one Component and each Component has precisely one Data member. Care should be used when using this simplified version of the similarity structure. Parsers are not provided for other languages (e.g. Matlab). However, users should be able to create a Java wrapper that parses the signature set and passes the appropriate data structures to the executables.

Writing

Like most XML documents, signature sets are easier to write than they are to parse. Thus, users can either use the C++ and Java classes supplied in BEE (distribution *mbgc/apps/btools/c++/sigset/* and the *mbgc/apps/btools/java/sigset/)* to create signature sets or output them directly. We recommend that the supplied classes be used because they have been rigorously tested and can easily be made compliant with changes in the schemas for signature sets.

Similarity Matrices

Overview

Similarity matrices are the primary output structures of recognition algorithms in MBGC. They consist of a header that specifies the type and dimension of the contained data and the $n \times m$ scores from the biometric algorithm.

Structure

The similarity matrix is similar to many image files in that it contains a textual header prepended to a binary representation of an $n \times m$ data structure. The structure of the header is depicted in Figure E. Here, we see that the header consists of four lines. The first line must contain either the character ‘D’ (for distance matrix) or the character ‘S’ (similarity matrix) followed by the character ‘2’. The second and third lines should contain the name of the target and query signature sets respectively. The target and query name should be the same as they were specified in the call to the matching executable. The fourth line should contain the characters ‘MF’, a space, the number of signatures in the query sigset, a space, the number of signatures in the target sigset, a space, and the integer 0x12345678 written in binary format. All four lines in the header should be terminated by an end-of-line character. Table C describes the elements in the similarity matrix header.

D2
sigsets/still_LG2200_leftLsession1.xml
sigsets/NIR_portal_samples_session1and2.xml
MF 139 70 xV4

Figure E Example of the similarity matrix header.

Name	Format	Separator	Comments
Storage Type	Character ‘S’ or ‘D’	none	Specified similarity scores, ‘S’, or distance measures ‘D’.
Version	The character ‘2’	eol	The value ‘2’ corresponds to the version of similarity matrix.
Target name	string	eol	This string should be the same as the name of the target sigset provided to the matching algorithm.
Query name	string	eol	This string should be the same as the name of the query sigset provided to the matching algorithm.
Format	The characters ‘MF’	space	The values correspond to a matrix, ‘M’, containing float, ‘F’, values.
Rows	integer	space	The number of signatures in the query set.
Cols	integer	space	The number of signatures in the target set.
Magic number	0x12345678 (4-bytes binary)	eol	This binary value is used to check for Endian (byte swapping).

Table C Description of elements in the similarity matrix header.

The scores are written to the file immediately following the header. These should be $n \times m$ 4-byte binary floating point values. Here, n is the number of signatures in the query set and m is the number of signatures in the target set. Thus, the first m values correspond to comparing the first query image to each of the target images. There must not be any white space characters separating scores in the body of the similarity matrix.

Parsing

Source code for C++ and Java classes for parsing similarity matrices are provided in the BEE (Biometric Experimentation Environment) Tools distribution *mbgc/apps/btools/c++/matrix/* and the *mbgc/apps/btools/java/matrix/*. Parsers will not be made available for other languages (e.g. Matlab). However, users should be able to easily create parsers in other languages using the C++ classes as guides.

Writing

Source code for C++ and Java classes for writing similarity matrices are provided in the BEE (Biometric Experimentation Environment) Tools distribution *mbgc/apps/btools/c++/matrix/* and the *mbgc/apps/btools/java/matrix/*. Writers will not be made available for other languages (e.g. Matlab). However, users should be able to easily write similarity matrices in other languages using the C++ classes as guides.

Mask Matrices

Overview

Mask matrices along with similarity matrices are an input to the analysis code that computes the performance of recognition algorithms in MBGC. Mask matrices provide the ground truth (answer key) for each MBGC experiment.

Structure

The mask matrix structure is generally the same as the similarity matrix structure in that it consists of a header that specifies the type and dimension of the contained data and the $n \times m$ scores from the biometric algorithm. The only difference is that mask matrices contain $n \times m$ byte (unsigned char) values while similarity matrices have float values. The structure of the mask matrix header is depicted below in Figure F. Here, we see that the header consists of four lines. The first line must contain the character 'M' (for mask matrix) followed by the character '2'. The second and third lines contain the name of the target and query signature sets respectively. The fourth line should contain the characters 'MB', a space, the number of signatures in the query sigset, a space, the number of signatures in the target sigset, a space and the integer 0x12345678 written in binary format. All four lines in the header should be terminated by an end-of-line character. Table C describes the elements in the mask matrix header.

M2
sigsets/still_LG2200_left_session1.xml
sigsets/NIR_portal_samples_session1and2.xml
MB 139 70 1425 xV4

Figure F Example of the mask matrix header.

Name	Format	Separator	Comments
Storage Type	Character ‘M’	none	Specified mask matrix values.
Version	The character ‘2’	eol	The value ‘2’ corresponds to the version of similarity matrix.
Target name	string	eol	The name of the target sigset associated with the experiment.
Query name	string	eol	The name of the query sigset associated with the experiment.
Format	The characters ‘MB’	space	The values correspond to a matrix, ‘M’, containing byte, ‘B’, values.
Rows	integer	space	The number of signatures in the query set.
Cols	integer	space	The number of signatures in the target set.
Magic number	0x12345678 (4-bytes binary)	eol	This binary value is used to check for Endian (byte swapping).

Table D Description of elements in the mask matrix header.

The header is prepended to a binary representation of an $n \times m$ data structure. The ground truth values are written to the file immediately following the header. These should be $n \times m$ 1-byte values. Here, n is the number of signatures in the query set and m is the number of signatures in the target set. Thus, the first m values correspond to comparing the first query image to each of the target images. There must not be any white space characters separating scores in the body of the mask matrix. The hex values in the matrix correspond to matches (0xff), non-matches (0x7f), and don’t care values (0x00). Don’t care values are ignored during accuracy scoring of similarity matrices.

Parsing

Source code for C++ and Java classes for parsing mask matrices are provided in the BEE (Biometric Experimentation Environment) Tools distribution in the *mbgc/apps/btools/c++/matrix/* and the *mbgc/apps/btools/java/matrix/* directories. Parsers will not be made available for other languages (e.g. Matlab). However, users should be able to easily create parsers in other languages using the C++ classes as guides.

Writing

Source code for C++ and Java classes for writing mask matrices are provided in the BEE (Biometric Experimentation Environment) Tools distribution in the distribution *mbgc/apps/btools/c++/matrix/* and the *mbgc/apps/btools/java/matrix/* directories. Writers will

not be made available for other languages (e.g. Matlab). However, users should be able to easily write mask matrices in other languages using the C++ classes as guides.

Log Files

Each executable should produce a log file. The name (and directory) for the log file will be provided in the parameter file via the *name* attribute of the *LogFile* element. The log file should provide detailed information to help troubleshoot algorithms.

Experiment	Target(s)	Query(s)	Mask Matrix(s)	Target Size	Query Size
StillFace_versus_HDVideoFace	still_medium_res_session1.xml	visible_video_portal_session1and2.xml	smrs1_vvps12_mask.mtx	139	70
	still_medium_res_session2.xml	visible_video_portal_session1and2.xml	smrs2_vvps12_mask.mtx	139	69
VideoIris_versus_NIR	video_LG2200_left_session1.xml	NIR_portal_samples_session1and2.xml	vlls1_npss12_mask.mtx	139	70
	video_LG2200_left_session2.xml	NIR_portal_samples_session1and2.xml	vlls2_npss12_mask.mtx	139	69
	video_LG2200_right_session1.xml	NIR_portal_samples_session1and2.xml	vlrs1_npss12_mask.mtx	139	70
	video_LG2200_right_session2.xm	NIR_portal_samples_session1and2.xml	vlrs2_npss12_mask.mtx	139	69
StillIris_versus_NIR	still_LG2200_left_session1.xml	NIR_portal_samples_session1and2.xml	slls1_npss12_mask.mtx	139	70
	still_LG2200_left_session2.xml	NIR_portal_samples_session1and2.xml	slls2_npss12_mask.mtx	139	69
	still_LG2200_right_session1.xml	NIR_portal_samples_session1and2.xml	slrs1_npss12_mask.mtx	139	70
	still_LG2200_right_session2.xml	NIR_portal_samples_session1and2.xml	slrs2_npss12_mask.mtx	139	69
MultipleBiometrics StillFace + VideoIrs versus HDVideoFace + NIR	still_medium_res_video_LG2200_left_session1.xml	visible_video_NIR_portal_session1and2.xml	smrvlls1_vvnps12_mask.mtx	139	70
	still_medium_res_video_LG2200_left_session2.xml	visible_video_NIR_portal_session1and2.xml	smrvlls2_vvnps12_mask.mtx	139	69
	still_medium_res_video_LG2200_right_session1.xml	visible_video_NIR_portal_session1and2.xml	smrvlrs1_vvnps12_mask.mtx	139	70
	still_medium_res_video_LG2200_right_session2.xml	visible_video_NIR_portal_session1and2.xml	smrvlrs2_vvnps12_mask.mtx	139	69
MultipleBiometrics StillFace + StillIrs versus HDVideoFace + NIR	still_medium_res_still_LG2200_left_session1.xml	visible_video_NIR_portal_session1and2.xml	smrslls1_vvnps12_mask.mtx	139	70
	still_medium_res_still_LG2200_left_session2.xml	visible_video_NIR_portal_session1and2.xml	smrslls2_vvnps12_mask.mtx	139	69
	still_medium_res_still_LG2200_right_session1.xm	visible_video_NIR_portal_session1and2.xml	smrslrs1_vvnps12_mask.mtx	139	70
	still_medium_res_still_LG2200_right_session2.xml	visible_video_NIR_portal_session1and2.xml	smrslrs2_vvnps12_mask.mtx	139	69

Table E MBGC experiments and data files

Notes

Unix users:

- use **dos2unix *** on all files in the *mbgc/(challengeproblem}/sigsets/* and the *mbgc/(challengeproblem}/param*/ directories.
- use **chmod a+r *** on all files in the *mbgc/(challengeproblem}/sigsets/*, the *mbgc/(challengeproblem}/maskmatrixes/*, and the *mbgc/(challengeproblem}/param/* directories to that they can be read.
- use **chmod a+x** * on all files in the *mbgc/bin/* and the *mbgc/(challengeproblem}/bin/* directories so that they can be executed.
- use **chmod a+w** * on the *mbgc/(challengeproblem}/output/* directory so that it is available for output.

- read mbgc/doc/MBGC_file_overview.pdf
- Set the value of the variable **DATA_DIR** in all of the Bash scripts (*.sh) the mbgc/(challengeproblem}/bin/ directory. The **DATA_DIR** variable's value should be the path to your top level MBGC data directory. For example, the **DATA_DIR** variable for the Portal Challenge should be the path to the directory containing the *NIR-face-video/*, *NIR-iris-video/*, *NIR-iris-stills*/, *visible-face-still*/, and *visible-face-video*/ directories. It should be the path to the directory containing the *Target*/ and *Query/* directories for the Still Challenge.

Windows users:

- read mbgc\doc\MBGC_file_overview.pdf
- Set the value of the variable **DATA_DIR** in all of the Bash scripts (*.bat) the *mbgc\(challengeproblem}\bin* directory. The **DATA_DIR** variable's value should be the path to your top level MBGC data directory. For example, the **DATA_DIR** variable for the Portal Challenge should be the path to the directory containing the *NIR-face-video*, *NIR-iris-video*, *NIR-iris-stills*, *visible-face-still*, and *visible-face-video* directories. It should be the path to the directory containing the *Target* and *Query* directories for the Still Challenge.