Face Analysis Technology Evaluation (FATE) MORPH

Performance of Automated Facial Morph Detection and Morph Resistant Face Recognition Algorithms

Concept, Evaluation Plan and API

VERSION 5.0.2

Mei Ngan
Patrick Grother
Kayee Hanaoka
Information Access Division
Information Technology Laboratory

August 16, 2024

NIST

National Institute of Standards and Technology

U.S. Department of Commerce

Revision History

1

Date	Version	Description
July 12, 2019	2.0	Initial document
September 9, 2020	2.0.1	Update link to General Evaluation Specifications document
July 7, 2021	2.1	Add optional `ageDeltaInDays` input argument to function `detectMorphDifferentially` (see Section 5.3.5)
May 19, 2022	3.0	- Remove optional `ageDeltaInDays` input argument to differential morph detection function in Section 5.3.5 - Add new function to support differential morph detection with additional subject metadata in Section 5.3.6
August 18, 2023	3.0.1	Updating project name from FRVT to FATE
February 1, 2024	5.0	Add new functions to perform demorphing (with and without a reference probe photo) in Sections 5.3.8 and 5.3.9. Incrementing version number to 5.0 to align with version of API header file.
April 5, 2024	5.0.1	Updating frequency of submissions to one algorithm submission every four calendar months (see Section 2.3).
August 14, 2024	5.0.2	In alignment with the draft ISO/IEC DIS 20059 standard deprecate APCER and replace with MACER.

2

Table of Contents

List of Tables

1. MORPH

1.1. Scope

Facial morphing (and the ability to detect it) is an area of high interest to a number of photo-credential issuance agencies and those employing face recognition for identity verification. The FATE MORPH test will provide ongoing independent testing of prototype facial morph detection technologies. The evaluation is designed to obtain an assessment on morph detection capability to inform developers and current and prospective end-users. This document establishes a concept of operations and an application programming interface (API) for evaluation of different tasks:

1. Algorithmic capability to detect facial morphing (morphed/blended faces) in still photographs
   a. Single-image morph detection of non-scanned photos, printed-and-scanned photos, and images of unknown photo format/origin
   b. Two-image differential morph detection of non-scanned photos, printed-and-scanned photos, and images of unknown photo format/origin
2. Face recognition algorithm resistance against morphing
3. Demorphing
   a. Single-image demorphing - algorithmic ability to recover images of the original identities from a single morphed face
   b. Two-image differential demorphing – algorithmic ability to recover the image of the "other unknown identity" in a morphed image, given the availability of a reference image belonging to one of the contributing subjects

1.2. General Evaluation Specifications

General and common information shared between all Ongoing FRTE/FATE tracks are documented in the General Evaluation Specifications document - https://pages.nist.gov/frvt/api/FRVT_common.pdf. This includes rules for participation, hardware and operating system environment, software requirements, reporting, and common data structures that support the APIs.

1.3. Reporting

For all algorithms that complete the evaluation, NIST will provide performance results back to the participating organizations. NIST may additionally report and share results with partner government agencies and interested parties, and in workshops, conferences, conference papers, presentations and technical reports.

Important: This is a test in which NIST will identify the algorithm and the developing organization. Algorithm results will be attributed to the developer. Results will be machine generated (i.e. scripted) and will include timing, accuracy and other performance results. These will be provided alongside results from other implementations. Results will be expanded and modified as additional implementations are tested, and as analyses are implemented. Results may be regenerated on-the-fly, usually whenever additional implementations complete testing, or when new analyses are added.

1.4. Accuracy metrics

This test will evaluate algorithmic ability to detect whether an image is a morphed/blended image of two or more faces and/or to correctly reject 1:1 comparisons of morphed images against other images of the subjects used to create the morph (but similarly, correctly authenticate legitimate non-morphed, mated pairs and correctly reject non-

89 morphed, non-mated pairs). Per established metrics[1,2] for assessment of morphing attacks, NIST will compute and
90 report:

91 • Morphing Attack Classification Error Rate (MACER) – the proportion of morph attack samples incorrectly
92 classified as bona fide presentation

93 • Bona Fide Presentation Classification Error Rate (BPCER) – the proportion of bona fide samples incorrectly
94 classified as morphed samples

95 • Mated Morph Presentation Match Rate (MMPMR) - the proportion of comparisons where the morphed
96 image successfully authenticates against all constituents

97 • True Acceptance Rate (TAR) – the proportion of non-morphed, mated comparisons that correctly
98 authenticate

99 • False Match Rate (FMR) – the proportion of non-morphed, non-mated comparisons that incorrectly
100 authenticate

101

102 We will report the above quantities as a function of alpha (the fraction of each subject that contributed to the morph),
103 image compression ratio, image resolution, image size, and others.

104 We will also report error tradeoff plots (BPCER vs. MACER, MMPMR vs. FMR, parametric on threshold).

105 2. Rules for participation

106 2.1.   Implementation Requirements

107 Developers are not required to implement all functions specified in this API. Developers may choose to implement
108 one or more functions of this API – please refer to Section 5.3.1 for detailed information regarding implementation
109 requirements.

110 2.2.   Participation agreement

111 A participant must properly follow, complete, and submit the FRTE/FATE MORPH Participation Agreement. This must
112 be done once, either prior or in conjunction with the very first algorithm submission. It is not necessary to do this for
113 each submitted implementation thereafter.

114 2.3.   Number and Schedule of Submissions

115 Participants may send one submission as often as every four calendar months from the last submission for evaluation.
116 NIST reserves the right to amend this section with submission volume and frequency limits. NIST will evaluate
117 implementations on a first-come-first-served basis and provide results back to the participants as soon as possible.

118 2.4.   Validation

119 All participants must run their software through the provided FATE MORPH validation package prior to submission.
120 The validation package will be made available at https://github.com/usnistgov/frvt. The purpose of validation is to
121 ensure consistent algorithm output between the participant's execution and NIST's execution. Our validation set is
122 not intended to provide training or test data.

---

[1] International Organization for Standardization: Information Technology – Biometric presentation attack detection – Part 3: Testing and reporting. ISO/IEC FDIS 30107-3:2017, JTC 1/SC 37, Geneva, Switzerland, 2017

[2] U. Scherhag, A. Nautsch, C. Rathgeb, M. Gomez-Barrero, R. Veldhuis, L. Spreeuwers, M. Schils, D. Maltoni, P. Grother, S. Marcel, R. Breithaupt, R. Raghavendra, C. Busch: "Biometric Systems under Morphing Attacks: Assessment of Morphing Techniques and Vulnerability Reporting", in Proceedings of the IEEE 16th International Conference of the Biometrics Special Interest Group (BIOSIG), Darmstadt, September 20-22, (2017)

123 ## 3. Data structures supporting the API

124 The data structures supporting this API are documented in this section and in the General Evaluation Specifications
125 document available at – https://pages.nist.gov/frvt/api/FRVT_common.pdf with corresponding header file named
126 *frvt_structs.h* published at https://github.com/usnistgov/frvt.

127 ### 3.1. Subject Metadata

128 Data structure representing information about a subject.

129 **Table 1 – Structure for a single image**

C++ code fragment	Remarks
```typedef struct SubjectMetadata```	
```{```	
``` Sex sex;```	Sex of the subject
``` int16_t ageInMonths;```	Age of subject (in months) in probe image; -1 indicates an unassigned value
``` int16_t ageDeltaInMonths;```	Age/time difference (in months) between probe and reference image; -1 indicates an unassigned value
```} SubjectMetadata;```	

130

131 **Table 2 - Labels for subject sex**

Label as C++ enumeration	Meaning
```enum class Sex {```	
``` Unknown=0,```	Either the label is unknown or unassigned
``` Female,```	
``` Male,```	
```};```	

132

133 ### 3.2. Requirement

134 FATE MORPH participants should implement the relevant C++ prototyped interfaces of section 5. C++ was chosen in
135 order to make use of some object-oriented features. Any functions that are not implemented should return
136 ```ReturnCode::NotImplemented.```

137 ## 4. Implementation Library Filename

138 The core library shall be named as libfrvt_morph_*<**provider**>_<**sequence**>*.so, with
139 - provider: single word, non-infringing name of the main provider. Example: acme
140 - sequence: a three digit decimal identifier to start at 000 and incremented by 1 every time a library is sent to
141   NIST. Example: 007
142
143 Example core library names: *libfrvt_morph_acme_000.so, libfrvt_morph_mycompany_006.so.*
144 Important: Public results will be attributed with the provider name and the 3-digit sequence number in the submitted
145 library name.

146 ### 4.1. File formats and data structures

147 #### 4.1.1. ImageLabel describing the format of an image

148 **Table 3 – Enumeration of image label**

Return code as C++ enumeration	Meaning
enum class ImageLabel {	
Unknown=0,	Image origin is unknown or unassigned
NonScanned=1	Non-scanned photo
Scanned=2,	Printed-and-scanned photo
};	

149

5. API specification

151 Please note that included with the FATE MORPH validation package (available at https://github.com/usnistgov/frvt) is
152 a "null" implementation of this API. The null implementation has no real functionality but demonstrates mechanically
153 how one could go about implementing this API.

5.1.   Header File

155 The prototypes from this document will be written to a file named **frvt_morph.h** and will be available to implementers
156 at https://github.com/usnistgov/frvt.

5.2.   Namespace

158 All supporting data structures will be declared in the FRVT namespace. All API interfaces/function calls for this track
159 will be declared in the FRVT_MORPH namespace.

5.3.   API

5.3.1.   Implementation Requirements

162 Developers are <u>not</u> required to implement all functions specified in this API. Developers may choose to implement
163 one or more functions of Table 4, but at a minimum, developers must submit a library that implements

164   1.   Interface of Section 5.3.2,

165   2.   initialize() of Section 5.3.3, and

166   3.   <u>AT LEAST</u> one of the functions from Table 4. For any other function that is not implemented, the function
167   shall return ReturnCode::NotImplemented.

168   **Table 4 – API Functions**

Function	Section

detectMorph() – single image morph detection of
    • Non-scanned photo
    • Printed-and-scanned photo
    • Image of unknown format
	5.3.4

detectMorphDifferentially() – two image differential morph detection of
    • Non-scanned photo
    • Printed-and-scanned photo
    • Image of unknown format
	5.3.5

compareImages() – 1:1 comparison	5.3.6

169

5.3.2.   Interface

171 The software under test **must** implement the interface Interface by subclassing this class and implementing AT
172 LEAST ONE of the methods specified therein.

	C++ code fragment	Remarks
1.	`Class MorphInterface`	
2.	`{` `public:`	
3.	`    static std::shared_ptr<Interface> getImplementation();`	Factory method to return a managed pointer to the `Interface` object. This function is implemented by the submitted library and must return a managed pointer to the `Interface` object.
4.	`    // Other functions to implement`	
5.	`};`	

173   There is one class (static) method declared in `Interface.getImplementation()` which must also be
174   implemented. This method returns a shared pointer to the object of the interface type, an instantiation of the
175   implementation class. A typical implementation of this method is also shown below as an example.

C++ code fragment	Remarks

```#include "frvt_morph.h"```
 	
```using namespace FRVT_MORPH;```
 	
```NullImpl:: NullImpl () { }```
 	
```NullImpl::~ NullImpl () { }```
 	
```std::shared_ptr<Interface>```
```Interface::getImplementation()```
```{```
```   return std::make_shared<NullImpl>();```
```}```
```// Other implemented functions```	

176   ### 5.3.3.   Initialization

177   Before any morph detection or matching calls are made, the NIST test harness will call the initialization function of
178   Table 5. This function will be called BEFORE any calls to fork() are made. This function <u>must</u> be implemented.

179   **Table 5 – Initialization**

Prototype	ReturnStatus initialize(	
	const std::string &configDir,	Input
	const std::string& configValue);	Input
Description	This function initializes the implementation under test and sets all needed parameters in preparation for template creation. This function will be called N=1 times by the NIST application, prior to parallelizing M >= 1 calls to any morph detection or matching functions via `fork()`. This function will be called from a single process/thread.	
Input Parameters	configDir	A read-only directory containing any developer-supplied configuration parameters or run-time data files.
	configValue	An optional string value encoding algorithm-specific configuration parameters. Developers may provide documentation for such configuration parameter(s) in their submission to NIST. Otherwise, the default value for this parameter will be an empty string.
Output Parameters	None	
Return Value	See General Evaluation Specifications document for all valid return code values. This function <u>must</u> be implemented.	

180

181 **5.3.4.**   **Single-image Morph Detection**

182 The function of Table 6 evaluates morph detection on non-scanned photos, scanned photos, and photos of unknown
183 formats. A single image along with an associated image label describing the image format/origin is provided to the
184 function for detection of morphing. Both morphed images and non-morphed images will be used, which will support
185 measurement of a morphing attack classification error rate (MACER) with a bona fide presentation classification error
186 rate (BPCER).

187 *Non-scanned photos*

188 Non-scanned photos are digital images known to <u>not</u> have been printed and scanned back in. There are a number of
189 operational use-cases for morph detection on such digital images.

190 *Scanned photos*

191 While there are existing techniques to detect manipulation of a digital image, once the image has been printed and
192 scanned back in, it leaves virtually no traces of the original image ever being manipulated. So the ability to detect
193 whether a printed-and-scanned image contains a morph warrants investigation.

194 *Photos of unknown format*

195 In some cases, the format and/or origin of the image in question is not known, so images with "unknown" labels will
196 also be tested.

197

198 Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on
199 different computers.

200   **Table 6 – Single-image Morph Detection**

Prototypes	ReturnStatus detectMorph(	
	const Image &suspectedMorph,	Input
	const ImageLabel &label,	Input
	bool &isMorph,	Output
	double &score);	Output
Description	This function takes an input image and associated image label describing the image format/origin, and outputs a binary decision on whether the image is a morph and a "morphiness" score on [0, 1] indicating how confident the algorithm thinks the image is a morph, with 0 meaning confidence that the image is not a morph and 1 representing absolute confidence that it is a morph.	
Input Parameters	suspectedMorph	Input Image
	label	ImageLabel (Section 4.1.1) describing the format of the input image • NonScanned = non-scanned digital photo • Scanned = a photo that is printed, then scanned • Unknown = unknown photo format/origin
Output Parameters	isMorph	True if image contains a morph; False otherwise
	score	A score on [0, 1] representing how confident the algorithm is that the image contains a morph. 0 means certainty that image does not contain a morph and 1 represents certainty that image contains a morph.
Return Value	See General Evaluation Specifications document for all valid return code values. If this function is not implemented, the return code should be set to `ReturnCode::NotImplemented`. If this function is not implemented for a certain type of image, for example, the function supports non-scanned photos but not scanned photos, then the function should return `ReturnCode::NotImplemented` when the function is called with the particular unsupported image type.	

201 **5.3.5.** **Two-image Differential Morph Detection**

202 Two face samples are provided to the function of Table 7 as input, the first being a suspected morphed facial image
203 and the second image representing a known, non-morphed face image of one of the subjects contributing to the
204 morph (e.g., live capture image from an eGate). This procedure supports measurement of whether algorithms can
205 detect morphed images when additional information (provided as the second supporting known subject image) is
206 provided.

207 Similar to single-image morph detection, the function of Table 7 will support non-scanned, scanned, and photos of
208 unknown format/origin. The input image type will be specified by the associated ImageLabel input parameter.

209 Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on
210 different computers.

211 <div align="center">**Table 7 – Two-image Differential Morph Detection**</div>

Prototypes	ReturnStatus detectMorphDifferentially(	
	const Image &suspectedMorph,	Input
	const ImageLabel &label,	Input
	const Image &probeFace,	Input
	bool &isMorph,	Output
	double &score);	Output
Description	This function takes two input images - a known unaltered/not morphed image of the subject (`probeFace`) and an image of the same subject that's in question (may or may not be a morph) (`suspectedMorph`) with an associated image label describing the image format/origin. This function outputs a binary decision on whether `suspectedMorph` is a morph (given `probeFace` as a prior) and a "morphiness" score on [0, 1] indicating how confident the algorithm thinks the `suspectedMorph` is a morph, with 0 meaning confidence that the `suspectedMorph` is not a morph and 1 representing absolute confidence that it is a morph.	
Input Parameters	suspectedMorph	Input Image
	label	ImageLabel (Section 4.1.1) describing the format of the suspected morph image • NonScanned = non-scanned digital photo • Scanned = a photo that is printed, then scanned • Unknown = unknown photo format/origin
	probeFace	An image of the subject known not to be a morph (e.g., live capture image)
Output Parameters	isMorph	True if image contains a morph; False otherwise
	score	A score on [0, 1] representing how confident the algorithm is that the image contains a morph. 0 means certainty that image does not contain a morph and 1 represents certainty that image contains a morph.
Return Value	See General Evaluation Specifications document for all valid return code values. If this function is not implemented, the return code should be set to `ReturnCode::NotImplemented`. If this function is not implemented for a certain type of image, for example, the function supports non-scanned photos but not scanned photos, then the function should return `ReturnCode::NotImplemented` when the function is called with the particular unsupported image type.	

212 **5.3.6.** **Two-image Differential Morph Detection with Subject Metadata**

213 Two face samples are provided to the function of Table 8 as input, the first being a suspected morphed facial image
214 and the second image representing a known, non-morphed face image of one of the subjects contributing to the
215 morph (e.g., live capture image from an eGate). **In addition**, subject metadata is provided as input to the algorithm,
216 which includes sex, age of the subject (in months) at the time the probe image is taken, and the age/time difference
217 (in months) between the suspected morph and the live probe image. Operationally, this information might be derived
218 from data read from the machine readable zone of a passport for example. This procedure supports measurement of
219 whether algorithms can detect morphed images when additional subject metadata is provided.

220

221 Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on
222 different computers.

223 **Table 8 – Two-image Differential Morph Detection with Subject Metadata**

Prototypes	ReturnStatus detectMorphDifferentially(	
	const Image &suspectedMorph,	Input
	const ImageLabel &label,	Input
	const Image &probeFace,	Input
	const SubjectMetadata &subjectMetadata,	Input
	bool &isMorph,	Output
	double &score);	Output
Description	This function takes two input images - a known unaltered/not morphed image of the subject (`probeFace`) and an image of the same subject that's in question (may or may not be a morph) (`suspectedMorph`) with an associated image label describing the image format/origin. Additionally, subject metadata is provided as input to the algorithm, which include sex, age of the subject (in months) at the time the probe image is taken, and the age/time difference (in months) between the suspected morph and the live probe image. This function outputs a binary decision on whether `suspectedMorph` is a morph (given `probeFace` as a prior) and a "morphiness" score on [0, 1] indicating how confident the algorithm thinks the `suspectedMorph` is a morph, with 0 meaning confidence that the `suspectedMorph` is not a morph and 1 representing absolute confidence that it is a morph.	
Input Parameters	suspectedMorph	Input Image
	label	ImageLabel (Section 4.1.1) describing the format of the suspected morph image • NonScanned = non-scanned digital photo • Scanned = a photo that is printed, then scanned • Unknown = unknown photo format/origin
	probeFace	An image of the subject known not to be a morph (e.g., live capture image)
	subjectMetadata	SubjectMetadata (Section 3.1) with information about the subject
Output Parameters	isMorph	True if image contains a morph; False otherwise
	score	A score on [0, 1] representing how confident the algorithm is that the image contains a morph. 0 means certainty that image does not contain a morph and 1 represents certainty that image contains a morph.
Return Value	See General Evaluation Specifications document for all valid return code values. If this function is not implemented, the return code should be set to `ReturnCode::NotImplemented`. If this function is not implemented for a certain type of image, for example, the function supports non-scanned photos but not scanned photos, then the function should return `ReturnCode::NotImplemented` when the function is called with the particular unsupported image type.	

224

225 **5.3.7.   1:1 Comparison**

226 Two face samples are provided to the function of Table 9 for one-to-one comparison of whether the two images are of
227 the same subject. The expected behavior from the algorithm is to be able to correctly reject comparisons of morphed
228 images against constituents that contributed to the morph. The goal is to show algorithm robustness against
229 morphing alterations when morphed images are compared against other images of the subjects used for morphing.
230 Comparisons of morphed images against constituents should return a low similarity score, indicating rejection of
231 match. Comparisons of unaltered/non-morphed images of the same subject should return a high similarity score,
232 indicating acceptance of match.

233

234 Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on
235 different computers.

236   **Table 9 – 1:1 Comparison**

Prototypes	ReturnStatus compareImages(	
	const Image &enrollImage,	Input
	const Image &verifImage,	Input
	double &similarity);	Output
Description	This function compares two images and outputs a similarity score. In the event the algorithm cannot perform the comparison operation, the similarity score shall be set to -1.0 and the function return code value shall be set appropriately.	
Input Parameters	enrollImage	The enrollment image
	verifImage	The verification image
Output Parameters	similarity	A similarity score resulting from comparison of the two images, on the range [0,DBL_MAX].
Return Value	See General Evaluation Specifications document for all valid return code values. If this function is not implemented, the return code should be set to `ReturnCode::NotImplemented`.	

237 **5.3.8.   Single-image Demorphing**

238 The function of Table 10 evaluates single-image "demorphing" – algorithmic ability to recover images of both
239 identities simultaneously from a single morphed face. The goal is to show algorithm ability to accurately restore the
240 identities of the contributing subjects if the image is a morph. All morphs will be generated with two contributing
241 subjects, and both morphed and non-morphed images will be evaluated. If the input image is a morph, the algorithm
242 should deduce/restore the two individual face images/identities that contributed to the morph. If the input is a bona
243 fide image, the algorithm should produce two images/identities that are essentially the same as the input photo. NIST
244 will report performance by analyzing face recognition outcomes between the original and restored imagery.

245

246 Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on
247 different computers.

248   **Table 10 – Single-image Demorphing**

Prototypes	ReturnStatus demorph(	
	const Image &suspectedMorph,	Input
	Image &outputSubject1,	Output
	Image &outputSubject2,	Output
	bool &isMorph,	Output (OPTIONAL)
	double &score);	Output (OPTIONAL)
Description	This function takes an input image and outputs two images. If the input image is a morph, the algorithm should deduce/restore the two individual face images/identities that contributed to the morph. If the input is a bona fide image, the algorithm should produce two images that are essentially the same as the input photo. **Optionally**, the algorithm can also return a binary decision on whether the image is a morph and a "morphiness" score on [0, 1] indicating how confident the algorithm thinks the image is a morph, with 0 meaning confidence that the image is not a morph and 1 representing absolute confidence that it is a morph. A score of -1.0 indicates that the algorithm did not implement morph detection and both "isMorph" and "score" will be ignored.	
Input Parameters	suspectedMorph	Input Image
Output Parameters	outputSubject1 outputSubject2	If the input image is a morph, the algorithm should deduce/restore the two individual face images/identities that contributed to the morph. If the input is a bona fide image,

		the algorithm should produce two images that are essentially the same as the input photo.
	isMorph (optional)	True if image contains a morph; False otherwise
	score (optional)	A score on [0, 1] representing how confident the algorithm is that the image contains a morph. 0 means certainty that image does not contain a morph and 1 represents certainty that image contains a morph. A score of -1.0 indicates that the algorithm did not implement morph detection and both "isMorph" and "score" will be ignored.
Return Value	See General Evaluation Specifications document for all valid return code values. If this function is not implemented, the return code should be set to `ReturnCode::NotImplemented`.	

249 **5.3.9.** **Two-image Differential Demorphing**

250 The function of Table 11 evaluates two-image differential "demorphing" – algorithmic ability to recover the image of
251 the "other unknown identity" in a morphed image, given the availability of a reference image belonging to one of the
252 contributing subjects. The goal is to show algorithm ability to accurately restore the identity of the second subject if
253 the image is a morph. All morphs will be generated with two contributing subjects, and both morphed and non-
254 morphed images will be evaluated. If the input image is a morph, the algorithm should deduce/restore the
255 second/unknown individual face image/identity that contributed to the morph. If the input is a bona fide image, the
256 algorithm should produce an image/identity that is essentially the same as the input photo. NIST will report
257 performance by analyzing face recognition outcomes between the original and restored imagery.

258

259 Multiple instances of the calling application may run simultaneously or sequentially. These may be executing on
260 different computers.

261 <div align="center">**Table 11 – Two-image Differential Demorphing**</div>

Prototypes	ReturnStatus demorphDifferentially(	
	const Image &suspectedMorph,	Input
	const Image &probeFace,	Input
	Image &outputSubject,	Output
	bool &isMorph,	Output (OPTIONAL)
	double &score);	Output (OPTIONAL)
Description	This function takes two input images - a known unaltered/not morphed image of the subject (`probeFace`) and an image of the same subject that's in question (may or may not be a morph) (`suspectedMorph`). If the input image is a morph, the algorithm should deduce/restore the other/unknown individual face image/identity that contributed to the morph. If the input is a bona fide image, the algorithm should produce an image that is essentially the same as the input photo. **Optionally**, the algorithm can also return a binary decision on whether the image is a morph and a "morphiness" score on [0, 1] indicating how confident the algorithm thinks the image is a morph, with 0 meaning confidence that the image is not a morph and 1 representing absolute confidence that it is a morph. A score of -1.0 indicates that the algorithm did not implement morph detection and both "isMorph" and "score" will be ignored.	
Input Parameters	suspectedMorph	Input Image
	probeFace	An image of the subject known not to be a morph (e.g., live capture image)
Output Parameters	outputSubject	If the input image is a morph, the algorithm should deduce/restore the other/unknown individual face image/identity that contributed to the morph. If the input is a bona fide image, the algorithm should produce an image that is essentially the same as the input photo.
	isMorph (optional)	True if image contains a morph; False otherwise
	score (optional)	A score on [0, 1] representing how confident the algorithm is that the image contains a morph. 0 means certainty that image does not contain a morph and 1 represents certainty that image contains a morph. A score of -1.0 indicates that the algorithm did not implement morph detection and both "isMorph" and "score" will be ignored.

Return Value	See General Evaluation Specifications document for all valid return code values.
	If this function is not implemented, the return code should be set to `ReturnCode::NotImplemented.`

262