FaceVACS-SDK C++ Reference Manual

Version 8.9.5

Copyright by Cognitec GmbH

Generated by Doxygen 1.8.4

Contents

0.1 Main Page

0.1.1 Manual Main Page

Please read the License Notice before you use the FaceVACS-SDK software.

The use of the software is only permitted if you agree with this license agreement. Other important information can be found in the Readme.txt file which is located in the root of the installation cdrom.

If you are already using FaceVACS-SDK find out what is new in this version.

For information about how to enable your development license see FaceVACS-SDK License Activation Procedure.

For information about supported platforms and available FaceVACS-SDK features see Supported Platforms and available functionality

For information about 3rd party licenses or copyrights used in this product see Licences and Copyrights.

The FaceVACS-SDK Manual consists of the following sections :

- FaceVACS-SDK Userguide
  - describes installation, biometric concepts and redistribution,
- FaceVACS-SDK Tutorial
  - explains step by step the use cases covered by FaceVACS-SDK.
- FaceVACS-SDK Reference
  - contains a detailed description of the interfaces and abstractions of FaceVACS-SDK,
- FaceVACS-SDK FAQ
  - contains answers to frequently asked questions related to FaceVACS-SDK.

0.2 License Agreement


```
FaceVACS SOFTWARE TECHNOLOGY
End User License Agreement (EULA)


COGNITEC'S TECHNOLOGY IDENTIFIED ABOVE IS A PROPRIETARY TECHNOLOGY OF
SOFTWARE APPLICATIONS WHICH INCLUDE THE SOFTWARE FaceVACS-ACQUISITION,
FaceVACS-DBSCAN, FaceVACS-ALERT AND FaceVACS-SDK AND MAY INCLUDE
ASSOCIATED MEDIA, PRINTED MATERIALS, AND ONLINE OR ELECTRONIC
DOCUMENTATION ("PRODUCTS") . ANY USE OF COGNITEC PRODUCTS IS SUBJECT
TO THE TERMS AND CONDITIONS AS OUTLINED IN THE EULA, A LEGAL AGREEMENT
BETWEEN YOU, THE END USER (EITHER AN INDIVIDUAL OR A SINGLE ENTITY),
AND COGNITEC. PLEASE READ THIS LICENSE AGREEMENT CAREFULLY BEFORE
USING OR INSTALLING THIS SOFTWARE. COGNITEC IS WILLING TO LICENSE THE
SOFTWARE TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE
TERMS CONTAINED IN THIS LICENSE AGREEMENT. IF YOU DO NOT AGREE TO THE
TERMS AND CONDITIONS OF THIS AGREEMENT, DO NOT INSTALL OR USE THE
SOFTWARE.

1. Definitions.

The terms used in this EULA beginning with a capital letter shall have
the meanings as defined below or as defined elsewhere in the text
marked with the term put in quotation marks: "Access" means to use or
benefit from using the functionality of the Software in accordance
with the documentation; "COGNITEC" means Cognitec Systems GmbH,
Germany, Grossenhainer Str.101, D-01127 Dresden, a company organized
under the laws of the Federal Republic of Germany; "Demonstration
Software" means a version of the COGNITEC FaceVACS -Software Product
licensed by COGNITEC for use in a technical environment designed to
conduct demonstration to customers and not for production use;
```

```
"Evaluation Software" means a version of the COGNITEC FaceVACS
-Software Product licensed by COGNITEC for use in a technical
environment designed to conduct evaluations of the Software and
testing and not for production use; "FaceVACS-Software" means a
Software Product provided by Cognitec under the brand of FaceVACS,
that is licensed to the user, including but not limited to
FaceVACS-Alert, FaceVACS-DBScan, FaceVACS-Acquisition, FaceVACS-SDK;
"FaceVACS-Component" means Integrators' Kit as integral component of
one of Cognitec's FaceVACS-Software Products to be used solely by the
Integrator for the sole purpose to generate Integrator applications
for third parties redistributable solely in combination with
Integrated Products; "Install" means to place a copy of the Software
onto a hard disk or other storage medium through any means (including
but not limited to use of an installation utility application
accompanying the Software) for the purpose of permitting Access to the
Software; "Integrator" means user of FaceVACS-Software developing
Integrated Products; "Integrated Product" means Software combined with
hardware devices (e.g. computers, cameras, door terminals) and/or
software products of COGNITEC or of a third party, forming a resulting
combined system that can perform functions that its independent
components could not perform separately; "Internal Computer Network"
shall mean your private, proprietary computer network resource
accessible only by employees and/or contractors of your specific
corporate enterprise or similar organization ("Authorized Users") and
solely controlled by you; "Internal Computer Network" specifically
excludes the Internet (as such term is commonly defined) or any other
network community that is open to the public, including without
limitation membership or subscription driven groups, associations or
similar organizations; connection by secure links such as VPN or dial
up to your Internal Computer Network for the purpose of allowing
Authorized Users to Access the Software is considered use over an
Internal Computer Network; "Permitted Number" means one (1) unless
otherwise indicated under a valid license granted by COGNITEC; "Sample
Code" means software source code or executable code, in either way
labelled as 'sample code'; "Software" means (a) all of the contents of
the files, disk(s), CD-ROM(s) or other media with which this Agreement
is provided, including but not limited to (i) COGNITEC or third party
computer information or software (including Evaluation Software and
Demonstration Software); (ii) digital images, stock photographs, clip
art, sounds or other artistic works ("Stock Files"); (iii) an
Application Programming Interface "API"; (iv) SQL database schemes and
guidance how to modify database schemes, and (v) related explanatory
written materials or files ("Documentation"); and b) fixes, modified
versions, updates, additions, and copies of the Software, if any,
licensed to you by COGNITEC (collectively, "Updates").

2. Software License.

2.1 General. As long as you comply with the terms of this End User
License Agreement (the "Agreement"), COGNITEC grants you a license to
use the Software as provided in this Section 2. In addition, depending
on which Software you have licensed, Sections 3. (FaceVACS
Software-Products), 4. (Evaluation Software) and/or 5. (Demonstration
Software) may also govern your use of the Software.

2.1 Backup Copy. You may make one (1) additional copy of each validly
licensed copy of the COGNITEC Software and Demonstration Software (but
not the Evaluation Software) in machine readable form for backup
purposes only, provided that you include any and all COGNITEC
copyright notices or other designations that appear or may appear in
or on the Software, without alteration or removal of any such
copyright or other notice on the original copy of the Software. You
may Install and Access backup copies of Demonstration Software only in
the event that your primary copy has failed. With respect to each copy
of COGNITEC Software, you may Install such backup copy on any of your
Computer(s) solely for backup purposes in the event of Computer
failures. You may Access the backup copy of COGNITEC Software for
production purposes only in the event that your primary copy has
failed or is under maintenance and is not in production use. At no
time may the total number of the Computer(s) on which you Access both
the primary copy and backup copy of COGNITEC software for production
purposes exceed the Permitted Number. Except as otherwise described in
this Section 2.3, all terms and conditions of this Agreement shall
```

apply to your Installation and Access of any backup copy of the Software. You may not transfer the rights to a backup copy unless you transfer all rights in the Software as provided in the Transfer Section of this Agreement.

2.2 Use in Compliance with the Law. As between you and COGNITEC, you assume all risk and are solely responsible for any and all liability resulting from Access to the Software in a way that violates (or that produces content that violates) any law or the rights of others including, without limitation, laws concerning copyright infringement.

2.3 You acknowledge that the Software licensed hereunder contains third party components that are licensed pursuant to its own terms and conditions ("Embedded Software"), A copy of the licenses associated with such Embedded Software can also be obtained on written request from Cognitec.

3. FaceVACS-Software

3.1 General. This Section applies only if you have obtained a valid license to COGNITEC Software Products other than Evaluation Software and Demonstration Software versions of such Software Products. In addition to the other terms contained herein, your license to the COGNITEC Software Products is limited as follows.

3.2 Software Installation. COGNITEC grants to you a non-exclusive license to Install COGNITEC Software Products on one or more Computer(s) on your Internal Computer Network for the purposes described in the specification of your license to Cognitec Software, provided that: (a) only the licensed functionality as set forth in the license agreement to COGNITEC Software is used or facilitated; and (b) none of the licensed parameters limiting the licensed functionality (like gallery size, number of calls per time period, number of concurrent users) do not exceed the Permitted Number. Except as expressly permitted herein, you may not Install all or any portion of COGNITEC Software onto any Computer if doing so would cause you to exceed the Permitted Number of licensed parameters.

3.3 Internal Computer Network Use. You may permit Authorized Users to Access COGNITEC Software through your Internal Computer Network as defined in your license agreement. In addition, but only so far as your license agreement allows, you may configure COGNITEC Software so as to automate (such as through the use of scripts and/or batch processing) software features (such as image resizing) provided that such automated process is initiated by Authorized Users from within your Internal Computer Network. You may Access COGNITEC Software to deliver content to any Computer connected to your Internal Computer Network so far as the license agreement allows.

3.4 Other Network Use. Except as otherwise provided in this Section 3.4 you may not (a) permit Access to COGNITEC Software by any users other than Authorized Users; or (b) Access COGNITEC Software (or permit others to do so) to deliver content either directly or through commands, data or instructions from or to a Computer not part of your Internal Computer Network unless this is allowed by your license agreement.

3.4.1 Authorized Users. Authorized Users may Access COGNITEC software (either directly or through an automated process) to deliver content to a Computer that is outside your Internal Computer Network so far as the license agreement allows;.

3.4.2 Other Users. So far as your license agreement allows you may provide Access to COGNITEC software or to specific features of COGNITEC Software to users outside your Internal Computer Network for the sole purpose of initiating a process (including an automated process) that results in COGNITEC Software delivering content (including content that has been modified based on user-specified preferences or information in accordance with this Agreement) to a Computer outside your Internal Computer Network.

3.5 Integrated Products: This subsection 3.5 only applies if you have obtained a valid license to Software as part of a device (e.g. camera, data-terminal) that delivers content to a computer or an Internal Computer Network. You are granted the nonexclusive right to use one (1) copy of this Software with each computer or Internal Computer Network connected to the device with which the Software was acquired.

3.6 Integrator’s Use Rights. As Integrator you may use the Software only on the basis of a valid FaceVACS Integrator license, install the FaceVACS Components on one or more computer(s) that are part of your Internal Computer Network, Access the API Information subject to Section 3.7 below, and use and modify the Sample Code and merge all or any portion of the Sample Code into your own code solely for the purpose of facilitating your Access to the Software to build applications that work in conjunction with the Product in accordance with the FaceVACS license agreement and with this Agreement. You may reproduce and redistribute any such Sample Code along with any modification (that means enhancements to the functionality of the Sample Code) you make thereto and the other files that may be listed and identified in the Documentation expressly as "redistributable files" provided that you agree: (a) to distribute such code only in object code form; (b) to merge the redistributable code into your own code only in combination with the Integrated Products; (c) to redistribute such code to third parties only on the basis of purchased end-user-licenses according to the FaceVACS license agreement between you and COGNITEC; (d) include COGNITEC’S copyright notices on your programs that include portions of Sample Code or the other redistributable code , except for those programs in which you include a copyright notice reflecting your own copyright ownership in such programs; (e) to indemnify, hold harmless and defend COGNITEC and its affiliates from and against any claims or lawsuits, including attorney’s fees, that arise or result from the use or distribution of the Integrated Products; and (f) to otherwise comply with the terms of this EULA.

3.7 Confidentiality. You agree that you will treat any received proprietary information from Cognitec including but not limited to API Information, know-how, ideas, concepts with the same degree of care to prevent unauthorized disclosure to anyone other than Authorized Users as you accord to your own confidential information, but in no event less than reasonable care. Your obligations under this Section with respect to such information shall terminate when you can document that such information was in the public domain at or subsequent to the time it was communicated to you by COGNITEC through no fault of yours. You may also disclose such information in response to a valid order by a court or other governmental body, when otherwise required by law, or when necessary to establish the rights of either party under this Agreement, provided you give COGNITEC advance written notice thereof.

4. Evaluation Software.

4.1 This Section 4 applies only if you have obtained a valid license to Evaluation Software. In addition to the other terms contained herein, your license to the Evaluation Software is limited to use strictly for your own internal evaluation purposes and not for production purposes, and is further limited to a period not to exceed sixty (30) days from the date you acquire the Evaluation Software unless otherwise agreed. You may Install the Evaluation Software on a total number of Computers not to exceed the Permitted Number, and permit Authorized Users to Access the Evaluation Software through your Internal Computer Network to deliver content within your Internal Computer Network. No other network use is permitted for Evaluation Software. Your rights with respect to Evaluation Software are further limited as described in subsection 8.2.

4.2 COGNITEC reserves the right to terminate your license to Evaluation Software at any time in its sole discretion. You agree to return or destroy your copy of the Evaluation Software upon termination of this Agreement for any reason. To the extent that any provision in this Section is in conflict with any other term or condition in this Agreement, this Section shall supersede such other term(s) and condition(s) with respect to the Evaluation Software, but only to the extent necessary to resolve the conflict

5. Demonstration Software.

5.1 This Section 5 applies only if you have obtained a valid license to Demonstration Software. In addition to the other terms contained herein, your license to the Demonstration Software is limited to use in your technical environment strictly for testing and demonstration purposes for customers and not for production purposes. You may Install the Demonstration Software on a total number of Computers not to exceed the Permitted Number, and permit Authorized Users to Access the Software through your Internal Computer Network to deliver content within your Internal Computer Network. No other network use is permitted for Demonstration Software. If the Product you have received with this license is Demonstration Software, then your rights under this Agreement will terminate immediately upon the earlier of (a) the expiration of agreed demonstration period, or (b) the expiration of your appointment as distributor.

5.2 COGNITEC reserves the right to terminate your license to Demonstration Software at any time in its sole discretion. You agree to return or destroy your copy of the Demonstration Software upon termination of this Agreement for any reason. To the extent that any provision in this Section is in conflict with any other term or condition in this Agreement, this Section shall supersede such other term(s) and condition(s) with respect to the Demonstration Software, but only to the extent necessary to resolve the conflict

6. Intellectual Property Rights.

6.1 The Software and any copies that you are authorized by COGNITEC to make are the intellectual property of and are owned by COGNITEC Systems GmbH and its suppliers. The structure, organization and code of the Software are the valuable trade secrets and confidential information of COGNITEC Systems GmbH and its suppliers.

6.2 The Software is protected by copyright, including without limitation by German Copyright Law, U.S. Copyright Law international treaty provisions and applicable laws in the country in which it is being used. You may not copy the Software, except as set forth in Section 2, (Software License) and 3 (FaceVACS Software limited as described under Section 3). Any copies that you are permitted to make pursuant to this Agreement must contain the same copyright and other proprietary notices that appear on or in the Software. Except as expressly stated herein, you agree not to modify, adapt or translate the Software. You also agree not to reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software except to the extent you may be expressly permitted to decompile under applicable law, it is essential to do so in order to achieve operability of the Software with another software program, and you have first requested COGNITEC to provide the information necessary to achieve such operability and COGNITEC has not made such information available. COGNITEC has the right to impose reasonable conditions and to request a reasonable fee before providing such information. Any information supplied by COGNITEC or obtained by you, as permitted hereunder, may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software. Requests for information should be directed to the COGNITEC Customer Support Department. Trademarks shall be used in accordance with accepted trademark practice, including identification of trademarks owners’names.

6.3 Trademarks can only be used to identify printed output produced by the Software and such use of any trademark does not give you any rights of ownership in that trademark.

6.4 Except as expressly stated herein, this Agreement does not grant you any intellectual property rights in the Software and all rights not expressly granted are reserved by COGNITEC.

7. Transfer.

You may not rent, lease, sell, sublicense, un-bundle and/or repackage for distribution or resale, or authorize all or any portion of the Software to be copied onto another user’s Computer or Installed or

Accessed on another user’s Computer except as may be expressly permitted herein or in a valid license agreement. You may not transfer or assign this Agreement or any license to use the Software without the prior written consent of COGNITEC, which shall not be unreasonably withheld. The parties agree that COGNITEC is hereby entitled to assign and/or transfer all or part of its rights and obligations under this Agreement to any third party. Notwithstanding the foregoing, any successor, representative or assignee which shall succeed by purchase, merger, or consolidation to the properties, substantially in your entirety as a legal entity shall be entitled to the rights and shall be subject to the obligations of its predecessor in interest under this Agreement, provided that such entity executes a prior acknowledgement confirming such entity’s acceptance and agreement to be bound by and comply with the terms of this Agreement and COGNITEC receives such written acknowledgement prior to the transfer. Notwithstanding anything to the contrary in this Agreement (with the exception of Section 8 thereof), you may not transfer FaceVACS- Evaluation Software and Demonstration Software.

8. Warranty.

8.1 Except as may be otherwise provided in the Sections 8.2, 8.3 and 8.4 below, COGNITEC warrants to the person or entity that first purchases a license for the Software from Cognitec for use pursuant to the terms of this license, that the Software will perform substantially in accordance with its specification for the ninety (90) day period following receipt of the Software when used on the recommended operating system and hardware configuration. Non-substantial variations of performance from the Documentation does not establish a warranty right. THIS LIMITED WARRANTY DOES NOT APPLY TO EVALUATION and DEMONSTRATION SOFTWARE, PRE-RELEASE (BETA), TRYOUT, PRODUCT SAMPLER, OR NOT FOR RESALE COPIES OF SOFTWARE (See Section 8.2). To make a warranty claim, you must return the Software to the location where you obtained it along with proof of purchase within such ninety (90) day period. If the Software does not perform substantially in accordance with the Documentation, the entire liability of COGNITEC and your exclusive remedy shall be limited to either, at COGNITEC’S option, the replacement of the Software or the refund of the license fee you paid for the Software. THE LIMITED WARRANTY SET FORTH IN THIS SECTION GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE ADDITIONAL RIGHTS, WHICH VARY FROM JURISDICTION TO JURISDICTION. For further warranty information, please contact COGNITEC..

8.2 NOTWITHSTANDING ANYTHING ELSE TO THE CONTRARY IN THIS AGREEMENT, EMBEDDED SOFTWARE, PRE-RELEASE (BETA)-, TRYOUT-SOFTWARE, PRODUCT SAMPLER, OR NOT FOR RESALE COPIES OF SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL COGNITEC OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8.3 Evaluation and Demonstration Software. YOU ACKNOWLEDGE THAT THE EVALUATION SOFTWARE AND THE DEMONSTRATION SOFTWARE MAY AUTOMATICALLY PLACE VISIBLE SIGNS ON CONTENT PROCESSED BY SUCH SOFTWARE AND IT COULD CONTINUE TO DO SO UNTIL SUCH TIME THAT YOU PURCHASE A LICENSE TO A FULL VERSION OF THE SOFTWARE. YOU ACKNOWLEDGE THAT THE ABSENCE OF SUCH SIGNS DOES NOT RESTRICT COGNITEC’S RIGHTS IN THE SOFTWARE AND THAT YOU MAY NOT DERIVE FURTHER RIGHTS FROM THIS ABSENCE. COGNITEC IS LICENSING THE EVALUATION AND DEMONSTRATION SOFTWARE ON AN "AS IS" BASIS, SOLELY AS A DEMONSTRATION MODEL. COGNITEC DISCLAIMS ANY WARRANTY OR LIABILITY OBLIGATIONS TO YOU OF ANY KIND, INCLUDING, WHERE LEGALLY LIABILITY CANNOT BE EXCLUDED FOR PRE-RELEASE SOFTWARE, BUT IT MAY BE LIMITED, COGNITEC’S LIABILITY AND THAT OF ITS SUPPLIERS SHALL BE LIMITED TO THE SUM OF FIFTY DOLLARS (U.S. $50) IN TOTAL.

8.4 Limited Warranty under German Law. If the laws of Germany is applicable according to Section 13, then Section 8.1 does not apply, instead, COGNITEC warrants that the Software will perform substantially in accordance with the Documentation for a period of twelve (12) months following receipt of the Software when used on the recommended hardware configuration. A non-substantial variation of performance from the Documentation does not establish a warranty right. To make a warranty claim, you must return the Software, at our expense, to the location where you obtained it along with proof of purchase within such six (6) month period. If the Software does not perform substantially in accordance with the Documentation, COGNITEC is entitled to repair or replace the Software according to its choice. If this fails, you are entitled to a reduction of the purchase price (reduction), or a recession of the purchase agreement (recession). For further warranty information, please contact COGNITEC.

9. Disclaimer

THE FOREGOING LIMITED WARRANTY STATES THE SOLE AND EXCLUSIVE REMEDIES FOR COGNITEC'S OR ITS SUPPLIER'S BREACH OF WARRANTY. COGNITEC AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE. EXCEPT FOR THE FOREGOING LIMITED WARRANTY, AND FOR ANY WARRANTY, CONDITION, REPRESENTATION OR TERM TO THE EXTENT TO WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, COGNITEC AND ITS SUPPLIERS MAKE NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS, EXPRESS OR IMPLIED, WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE AS TO ANY OTHER MATTERS, INCLUDING BUT NOT LIMITED TO NON-INFRINGEMENT OF THIRD PARTY RIGHTS, INTEGRATION, SATISFACTORY QUALITY OR FITNESS FOR ANY PARTICULAR PURPOSE. THE PROVISIONS OF SECTION 9 AND SECTION 10 SHALL SURVIVE THE TERMINATION OF THIS AGREEMENT, HOWSOEVER CAUSED, BUT THIS SHALL NOT IMPLY OR CREATE ANY CONTINUED RIGHT TO INSTALL OR ACCESS THE SOFTWARE AFTER TERMINATION OF THIS AGREEMENT.

10. Limitation of Liability under German Law.

If the Laws of Germany is applicable according to Section 13, then Section 9 does not apply, instead, COGNITEC may be liable without limitation for damages you have incurred under or in connection with this Agreement only if the damage has been caused by the wilful or grossly negligent act of COGNITEC or its agents. COGNITEC is liable only to the extent of the typically foreseeable damage for such damages, which have been caused by any other negligent breach of a substantial contractual duty by COGNITEC or its agents. Any further liability of COGNITEC is excluded. These aforementioned limitations apply irrespective of their legal basis, in particular with regard to any pre-contractual or auxiliary contractual claims. The limitations shall not apply, however, to any mandatory liability under the applicable German Product Liability Act, nor to any damage which is caused due to the breach of an express warranty to the extent that such express warranty was intended to protect the user against the specific damage incurred.

11. Limitation of Liability.

IN NO EVENT WILL COGNITEC OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY DAMAGES, CLAIMS OR COSTS WHATSOEVER OR ANY CONSEQUENTIAL, INDIRECT, INCIDENTAL DAMAGES, OR ANY LOST PROFITS OR LOST SAVINGS, EVEN IF A COGNITEC REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS, DAMAGES, CLAIMS OR COSTS OR FOR ANY CLAIM BY ANY THIRD PARTY. THE FOREGOING LIMITATIONS AND EXCLUSIONS APPLY TO THE EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION. COGNITEC'S AGGREGATE LIABILITY AND THAT OF ITS SUPPLIERS UNDER OR IN CONNECTION WITH THIS AGREEMENT SHALL BE LIMITED TO THE AMOUNT PAID FOR THE SOFTWARE, IF ANY. IN CASE THE LAWS OF GERMANY APPLY, Nothing contained in this Agreement limits COGNITEC's liability to you in the event of death or personal injury resulting from COGNITEC's negligence or for the tort of deceit (fraud). COGNITEC IS ACTING ON BEHALF OF ITS SUPPLIERS FOR THE PURPOSE OF DISCLAIMING, EXCLUDING AND/OR LIMITING OBLIGATIONS, WARRANTIES AND LIABILITY AS PROVIDED IN THIS AGREEMENT, BUT IN NO

```
OTHER RESPECTS AND FOR NO OTHER PURPOSE. FOR FURTHER INFORMATION,
PLEASE CONTACT COGNITEC.

12. Export Rules.

In general, this Software is subject to national or trans-national
export restriction regulations, including but not limited to export
restriction regulations for goods that can be used for permitted
purposes and for prohibited purposes (i.e. dual use goods). You agree
that the Software will not be shipped, transferred or exported into
any country to be used or used in any manner or for purposes
prohibited by the German and European Export Trade Restrictions and
the United States Export Administration Act or any other export laws,
restrictions or regulations (collectively the "Export Laws"). In
addition, if the Software is identified as export controlled items
under the Export Laws, you represent and warrant that you are not a
citizen, or otherwise located within, an embargoed nation (including
without limitation Iran, Iraq, Lebanon, Myanmar, Zimbabwe, Uzbekistan,
Belarus and North Korea) and that you are not otherwise prohibited
under the Export Laws from receiving the Software. All rights to
Install, Access and use the Software are granted on condition that
such rights are forfeited if you fail to comply with the terms of this
Agreement.

13. Governing Law.

This Agreement, each transaction entered into hereunder, and all
matters arising from or related to this Agreement (including its
validity and interpretation), will be governed and enforced by the
substantive laws in force in: (a) the State of Florida, if a license
to the Software is purchased by you having your registered seat of
entity in the United States, Canada, or Mexico; or (b) Japan, if a
license to the Software is purchased by you having your registered
seat of entity in Japan, China, Korea, or other Southeast Asian
country where all official languages are written in either an
ideographic script (e.g., hanzi, kanji, or hanja), and/or other script
based upon or similar in structure to an ideographic script, such as
hangul or kana; or (c) The Federal Republic of Germany, if a license
to the Software is purchased by you having your registered seat of
entity in any other jurisdiction not described above. The respective
courts of Miami, Florida when Florida law applies, Tokyo District
Court in Japan, when Japanese law applies, and the competent courts of
Germany at COGNITEC'S registered seat, when the law of Germany
applies, shall each have non-exclusive jurisdiction over all disputes
relating to this Agreement. This Agreement will not be governed by the
conflict of law rules of any jurisdiction or the United Nations
Convention on Contracts for the International Sale of Goods, the
application of which is expressly excluded.

14. Notice to U.S. Government End Users.

To the extent any licenses granted herein are deemed to constitute the
acquisition of software on behalf of any agency or unit of the United
States Government, it is agreed and understood (i) that the Software
is classified as "Commercial Computer Software" and the Government is
acquiring only "restricted rights" in the software product, as
defined in clause 48 C.F.R. 227-7205-5(c) ; or (ii) the Government's
rights in the Software shall be as defined in clause 48
C.F.R. 52.227-19 of the FAR.

15. Third-Party Beneficiary.

You acknowledge and agree that COGNITEC'S licensors and/or COGNITEC if
you obtained the Software from Cognitec Systems Corporation, a
Delaware corporation, USA, 13800 Coppermine Road, Herndon, VA 20171,
USA an affiliate of COGNITEC or any party other than COGNITEC are
third party beneficiaries of this Agreement, with the right to enforce
the obligations set forth herein with respect to the respective
technology of such licensors and/or COGNITEC.

16. Miscellaneous.

If any part of this Agreement is found void and unenforceable, it will
```

```
not affect the validity of the balance of the Agreement, which shall
remain valid and enforceable according to its terms. This Agreement
shall not prejudice the statutory rights of any party dealing as a
consumer. Updates may be licensed to you by COGNITEC with additional
or different terms. In case of a conflict between this Agreement and
your valid license the valid license agreement shall prevail.

---

If you have any questions regarding this Agreement or if you wish to
request any information from COGNITEC please use the address and
contact information included with this Product to contact COGNITEC.


Copyright COGNITEC Systems GmbH. All rights reserved.
```

0.3 Readme

```
FaceVACS-SDK 8.9.5
------------------

Cognitec's FaceVACS-SDK is a market-leading toolbox for developing
applications for facial recognition.

The contents of this software package are protected by copyright.
Your acceptance of the End-User License Agreement (EULA) is required
before installation.  The EULA can be viewed in file Eula.txt.

Please read the manual concerning installation and usage. You will
also find answers to frequently asked questions there:
doc\refman.pdf (PDF) or doc\html\index.html (HTML).

A detailed description of algorithmic issues can be found in the
document doc\FaceVACSalgorithms.pdf.

A detailed description of sample evaluation issues can be found in the
document doc\FaceVACSsamplevaluation.pdf.

Licensing information
---------------------

The functionality of the software is controlled by a license key. With
the purchase of the software license, you are able to run programs
linked with the FaceVACS-SDK libraries. To enable the purchased license
please read the FaceVACS-SDK Manual section.

System requirements
-------------------

* Intel x586/686-compatible processor, minimum 800 MHz
  (Pentium III or higher, Celeron, Duron, Athlon etc.)
* minimum 384 MB RAM
* Windows XP, Windows Vista, Windows 7, Linux (all PLatforms with support
  for 32 and 64 bit)

For using the Windows Capture Device implementation:

* Video drivers with DirectShow support

* USB cameras like the Philips ToUCam Pro II are recommended

Installation
------------

* Windows:
  -  Run the win32\setup.exe program

* Linux
  - extract the tar file from the installation package and unpack the
    archive into a directory of your choice
```

```
After this is done please read the instructions for setting the build
environment in the FaceVACS-SDK Manual.

Uninstall
---------

* Windows:
  - To un-install the software run Start->Programs->FVSDK_8_9_5->Uninstall FaceVACS-SDK

* Linux
  - Just remove the directory where the software has been installed to (e.g. FVSDK_8_9_5).

Contact
-------

More information can be found on

www.cognitec.com

Questions concerning the software can be directed to

support@cognitec.com

Content of the software package
------------------------------

You will find the following files and
directories:

Readme.txt
   this file
Eula.txt
   End-User License Agreement

win32\setup.exe
    Setup program for Windows
linux\FVSDK_8_9_5.tar.gz
    tarball for linux
android\FVSDK_8_9_5.tar.gz
    tarball for android
doc\specification.pdf
    the technical specification of the product (PDF)
doc\refman.pdf
    the English manual (PDF)
doc\refman.net.pdf
    the English reference manual of the FaceVACS .NET SDK (PDF)
doc\refman.net.pdf
    the English reference manual of the FaceVACS Java SDK (PDF)
doc\refman.android.pdf
    the English reference manual of the FaceVACS Android SDK (PDF)
doc\html\index.html
    the English manual (HTML) without API reference
```

0.4 Copyright

```
3rd Party Copyrights
--------------------

This software is based in part on the work of the Independent
JPEG Group (Jpeg Compression Library)


For reading JPEG 2000 images, the JasPer library v. 1.701 is used:

Copyright (c) 1999-2000 Image Power, Inc.
Copyright (c) 1999-2000 The University of British Columbia
Copyright (c) 2001-2003 Michael David Adams
```

```
All rights reserved.
```


```
For reading and writing png images, libpng is used:

libpng version 1.0.15 - October 3, 2002
Copyright (c) 1998-2002 Glenn Randers-Pehrson
(Version 0.96 Copyright (c) 1996, 1997 Andreas Dilger)
(Version 0.88 Copyright (c) 1995, 1996 Guy Eric Schalnat, Group 42, Inc.)

This product includes software developed by the Apache Software
Foundation (http://www.apache.org/)
(Xerces XML Library)

Parts of the product make use of the cryptographic library
Crypto++; Version 5.2.1  by Wei Dai

The XML schemas provided for the xml based evaluation framework are
modified versions of those provided within the Human Evaluation
Framework (HEF) used during Face Recognition Vendor Test 2002
(www.frvt.org).
```

0.5 User guide - Overview

This user guide gives an introduction to the FaceVACS-SDK.

It explains the use cases of FaceVACS-SDK and the underlying biometric concepts, the installation and build environment settings as well as the redistribution.

The user guide is divided into the following chapters with sections:

0.6 "User Guide - Introduction"

[Next chapter]

0.6.1 What is FaceVACS-SDK

Cognitec's Face Recognition SDK is the way of bringing Face Recognition to a broad range of applications and allows to use face recognition as a biometric authentication model. FaceVACS-SDK uses modern techniques of programming with consistent, logically clear programming interfaces. It covers the basic functions (use cases) of Enrollment, Verification, Identification and defines abstractions which allow SDK users to create implementations for their own applications. Additionally, the FaceVACS-SDK supports the finding of human faces and eyes in images.

0.6.2 Features of FaceVACS-SDK

The main features of FaceVACS-SDK are:

- State of the art face recognition API for easy integration of biometric technology into a broad range of applications
- Support for biometric operations: enrollment, verification and identifications from image streams (live video) and image collections (image databases)
- C++, .NET and Java interface to support modern paradigms of software development:
  - object oriented design
  - extensibility and modularity
- Abstract interfaces support easy adaption to various video streaming hardware and databases
- Detailed manual including API reference and User Guide
- Fully documented examples illustrating the main use cases and providing hints how to create customized implementations
- Suite of tools to perform biometric evaluations on data residing in SQL databases (generation of identification match lists and score matrix data). At the time being, these tools are available for 32 bit platforms only.

0.6.3 Supported Platforms and available functionality

The following table lists which of the features are available on which platform. An **X** marks a supported feature, while a **-** means that the feature is not supported.

All supported platforms are listed in section FaceVACS-SDK Requirements. Windows x86_32 and Windows x86_64 comprise Windows XP SP3 or newer on a 32 bit, respectively 64bit, i686 compatible machine.

The naming for Linux platforms is analogous to the one for Windows.

Feature	related FaceVACS SDK Namespaces, Classes and Binaries	Windows x86_32	Windows x86_64	Linux x86_32	Linux x86_64	MacOS x86_32	MacOS x86_64
Face and Eye Finder	FRsdk::Eyes, FRsdk::Face, (except FRsdk::Face::Tracker)	X	X	X	X	X	X
Comparison (Enrollment, Verification, Identification, Matching)/td>	FRsdk::Enrollment, FRsdk::Identification, FRsdk::Verification, FRsdk::Tools	X	X	X	X	X	X
Portrait Characteristics	FRsdk::Portrait, FRsdk::Portrait::Feature, FRsdk::ISO_19794_5, FRsdk::ISO_19794_5::FullFrontal, FRsdk::ISO_19794_5::TokenFace	X	X	X	X	X	X

Image Formats	FRsdk::Bmp, FRsdk::ImageIO, FRsdk::Jpeg, FRsdk::Jpeg2000,FRsdk::Pgm, FRsdk::Png	X	X	X	X	X	X
Tracking	FRsdk::Face::Tracker	X	X	X	X	X	X
Biometric Evaluation Tools	see Biometric Evaluation Tool Suite	X	X	-	-	X	X
.Net API	see `FaceVACS-SDK .NET`	.Net 2.0	.Net 2.0	-	-	-	-

0.6.4 FaceVACS-SDK requirements

FaceVACS-SDK supports only Intel i686 compatible computer hardware. The supported **development and deployment platforms** are:

- MS Windows XP with SP3 (32 and 64 bit)
- MS Windows Vista (32 and 64 bit)
- MS Windows 7 (32 and 64 bit)
- MS Windows 8 (64 bit)
- MS Windows Server 2003 with SP2
- MS Windows Server 2003 R2
- MS Windows Server 2008 with SP2
- MS Windows Server 2008 R2
- MS Windows Server 2012
- Linux (32 and 64 bit)
- Mac OS X (32 and 64 bit)

On these platforms the following c++ compilers will be supported:

- for MS Windows: MS Visual C++ 12.0, 11.0 and 10.0 (32 and 64 bit)
- for Linux: GNU C++ 4.6.X and 4.3.X (both 32 and 64 bit)
- for Mac OS X: GNU C++ 4.2.X (both 32 and 64 bit)

On MS Windows, Linux and Mac OS X platforms several run-time libraries (DLLs, shared libraries) are required (see Run-Time Environment for FaceVACS-SDK Applications), which are usually part of the compiler software distributions. It is the responsibility of the FaceVACS-SDK application developer to provide these DLLs to the end user of the application.

0.6.5 Installation of FaceVACS-SDK

- **Linux:** The FaceVACS-SDK is distributed as a gzip'ed tar file. Please untar the file into a location of your choice. After installation, it is necessary to create the correct links to the libraries in the share folder using ldconfig. Therefore please execute the following command in a shell: ***/sbin/ldconfig -n <install-path>/<lib/ulib>/x86_<32/64>/share***

remark: Replace <install-path> with the root path of the FaceVACS-SDK installation and choose 32 or 64 according to the architecture of your computer. On Mac OS X, please repeat that step in the 'ubin' subdirectory, which contains universal binaries and libraries for both, 32 and 64 bit code.

To complete the installation, you need to make some adjustments in the configuration file frsdk.cfg. See Installation Settings for detailed instructions.

remark: Before starting cfgedit from the command line, please set the correct library first. See DLL/Shared library path configuration for information on how to set the path.

- **MS Windows:** The FaceVACS-SDK is distributed as a self-extracting archive with a built-in installer. Just follow the installation wizard instructions.

For all operating systems refer to section Run-Time Environment for FaceVACS-SDK Applications to get the delivered examples running. For the system to find the required FaceVACS-SDK libraries the correct search path must be set (see DLL/Shared library path configuration how to set search path for libraries).

Furthermore a valid activation key is always required to run any application using FaceVACS SDK. See Activating the FaceVACS-SDK License for more details.

0.6.6 FaceVACS-SDK License Activation Procedure

The functionality of the software is determined by the activation information in the FaceVACS-SDK configuration file. Licenses are bound to a computer (see below) and may be limited in time. To run programs linked with the FaceVACS-SDK libraries you have to

- purchase software licenses for each computer where programs are supposed to run
- create a "Computer Identification" on any of these computers
- send the key(s) to Cognitec (`license@cognitec.com`)
- receive the activation information for each machine from Cognitec and
- transfer the activation information to the FaceVACS configuration file of each machine.

  Warning: The limitations are stored in protected configuration keys. Modifying these keys corrupts the activation information and leads to license mismatch exceptions of your application.

Computer Identification

There are several types of computer identifications:

- **HostId**
- **System**
- **HaspId**

Identification based on the host or system id uses information about the pc hardware to create a unique key. No additional hardware or driver is necessary. The HaspID, in contrast, is read from a USB dongle, requiring a driver provided with the FaceVACS SDK software package. A dongle may be moved freely between computes, i.e. they are not bound to one pc.

Installation of the HASP Key USB Driver

The driver for Linux and Windows is provided with the installation package of FaceVACS SDK. The driver can be found in the "hasp-driver" directory inside the installation directory.

To install the driver open a shell and switch to the directory. On Windows start the executable **haspdinst.exe** located in the directory and read the displayed messages carefully. The installation of the driver will be performed, when the executable is called with the command line switch **-install**.

To install the driver on Linux you need root privileges. Change into the /hasp-driver directory and either the /x86_32 or /x86_64 subdirectory, depending on your machine architecture. Execute **./dinst .** and follow the instructions.

The following platforms are supported by **Haspld**:

- Windows XP (w/ Service Pack 3) - 32- & 64-bit
- Windows Server 2003 (w/ Service Pack 2) - 32- & 64-bit
- Windows Server 2008 - 32- & 64-bit
- Windows Server 2012
- Windows Vista (w/ Service Pack1) - 32- & 64-bit
- Windows 7 - 32- & 64-bit
- Windows 8
- SLES 10 SP1 - 32- & 64-bit
- RHEL 5 SP1? - 32- & 64-bit
- Ubuntu Desktop 8.04 - 32- & 64-bit

0.6.7 Creating the Computer Identification

```
<p>
  The Computer Identification should be created and stored to a file using the
  FaceVACS-SDK Configuration Editor:
```

- Start the Configuration Editor
- Select the License Management Dialog via the menu: Setup -> License
- Select Save as... to store the Computer Identification in a file

  Alternatively the Computer Identification can be created using the command line tool `hwkey`. Please follow these steps according to your operating system:

  Linux or MacOS:
  1. Open a command shell.
  2. Change to FaceVACS-SDK installation sub-directory `bin/x86_32, x86_64 or ubin(macos only)`
  3. Execute
     `./hwkey -store`

  All supported MS Windows flavours:
  1. Open a command prompt.
  2. Change to FaceVACS-SDK installation sub-directory `bin\x86_32 or x86_64`
  3. Execute
     `hwkey -store`

  The Computer Identification is stored to the file **hwkey.cfg** and the terminal should display a message like this:

```
C:\\FVSDK_2_0\\bin\\x86_32\\ > hwkey -store
 HWKey stored to: hwkey.cfg
```

Diagnosis

If you want to read and check the Computer Identification, call simply the tool `hwkey` in the directory where it is located. The tool will display lines like that:

```
C:\FVSDK_8_9_2\bin\x86_32\> hwkey
FNUseHardwareKey = HostId
FNHardwareKey = aa095d4c2e0fff:baa356da
```

0.6.8 Obtaining the Activation Information from Cognitec

Once you have created the Computer Identification on each machine, send the file(s) labelled with the machine names to: `license@cognitec.com`

The license(s) will be issued by Cognitec and sent to your email account as attached **activation file(s)**.

0.6.9 Transferring the Activation Information to the Configuration File

The transfer is accomplished by the Configuration Editor:

- Start the Configuration Editor
- Select the License Management Dialog via the menu: Setup -> License
- Select Import Activation Key from File

Alternatively the 'liccopy' command line utility which is placed in the "bin" sub-directory of your FaceVACS-SDK installation, can be used to activate your license too.

Copy each activation file received in your email to the PC it has been generated for and transfer the activation information to your FaceVACS-SDK configuration file with

liccopy -f <FaceVACS-SDK configuration file> -l <activation file>

For both files you have to provide the complete path name unless they do not reside in the working directory of your command line invocation. Please note, that copying the license using **liccopy** removes all comments (lines starting with #, which are ignored upon parsing) from the configuration file.

Since you might want to have several FaceVACS-SDK configuration files (using different settings for different applications), you will have to transfer the activation information to each of them.

[Next chapter]

0.7 User Guide - Face Recognition Concepts

[Previous chapter] [Next chapter]

0.7.1 Input Data - Samples

Data Samples

The main biometric use cases enrollment, verification and identification explained below require facial data to be processed. Current FaceVACS algorithms can process both intensity image data and shape data. To allow for a uniform interface for facial data input, the FRsdk::Sample data type is provided. This is a compound data type comprising a mandatory intensity image and optional shape data and eye annotations.

Intensity Image Data

Intensity image data can be an approximately frontal gray-scale or color image of a human face in 'photographic view'. More technically this means that the image is created by a central or 'perspective' projection of the face to a plane approximately parallel to the frontal face (see "Pose Requirements" below for admissible deviations from frontal view). Though this seems to be a dispensable requirement since most of the video and images sources usually do provide this kind of images, some do not - for example, 3D sensors providing data in cylindrical coordinates, which provide their texture images in cylindrical projection, too.

Shape and Intensity Data Alignment Requirements

Shape data as delivered by a sensor usually is provided as a set of 3D points or 'vertices'. Though the data might be provided as just a list of vertices, some structuring is useful and provided by most 3D sensors. More specifically, the vertices are often indexed by a 2 dimensional integer grid, i.e. they are arranged like pixels in an intensity image. This is the representation required for the shape image part of FRsdk::Sample.

However, there is one more requirement concerning the relation between the intensity image and the shape image of a sample. Both images should (within some very short time frame) be taken at virtually the same instant and they should be pixel-aligned. This means that each pixel of the intensity image is the projection of the face shape at the location of the corresponding (within the underlying integer grid) vertex.

Shape data resolution and accuracy requirements

- Minimal resolution for shape data is appr. 100 vertices within eye distance
- Accuracy of vertex data 1 mm or better

Shape preprocessing algorithms performed on the shape data can reduce the impact of noise and outliers. However, for shape data with lower accuracy and/or multiple outliers and gaps, biometric performance will degrade.

Facial pose requirements

While a frontal view of the upright face is recommended, some deviations are admissible. A horizontal (yaw) or nose up/down (pitch) rotation of the head of up to 15 degrees is admissible. Also, the face is admitted to deviate from upright view (roll) up to 15 degrees.

Annotations

In addition, eye positions can be provided with a sample. This is useful in cases where for some reason the automatic eye finding algorithms are not appropriate. It also can speed up re-enrollment of existing data after changes of the comparison algorithm, which requires recreation of facial identification records.

How facial data is processed

Which facial data is required and which is used depends on the configuration of the comparison algorithm to be used. By default, only the intensity image is required and used, shape data is silently ignored.

If the configuration is set to use shape data, it becomes mandatory. In this case, samples not containing shape data will be rejected.

0.7.2 Biometric Use Cases

Enrollment

The basic approach with face recognition is similar to that of other biometric technologies like finger print, voice recognition etc. First, an initial *feature set* is constructed from the relevant physical traits of the user. In the case of face recognition, this is done by collecting a set of face samples from a "capturing device". This device can be a real streaming video source like a camera. The samples are analyzed by the algorithms and salient features are extracted. From these features, a so-called *feature set* is constructed. The process of creating this feature set from one or more input images is called enrollment. The feature set generated can then be stored by the application and essentially substitutes a password.

Verification

When a user is to be authenticated (i.e. the user's identity is to be verified), samples will be captured from the device and again a feature set is created. This *feature set* is then compared with the enrollment feature set. If the resulting score value is above a predefined threshold, the user is considered to be authenticated.

Identification

In contrast to the verification use case, with identification the (claimed) identity of the user is not known in advance, but shall be determined based on sample images of the user's face and a set (population) of feature sets with known identities. The identification system takes some samples of the user's face, generates a feature set from them and compares this feature set with each element of this population. The elements yielding the highest comparison values above a certain confidence threshold are candidates for the identity wanted.

0.7.3 The Facial Identification Record (FIR)

The raw biometric samples as produced by the sensing devices (i.e. images in FaceVACS-SDK context) contain noisy, redundant data. Two samples of the same user are unlikely to be identical. In the result of processing the raw samples (images), e.g. during enrollment, feature sets are created. In the context of FaceVACS-SDK we use the term FIR (facial identification record) for these feature sets. The FIR is the digital representation of facial features extracted out of the samples and, as a C++ object, can be serialized (represents a byte stream).

Note

> Please note that the data contained in the FIR strongly depends on the technology, i.e. the algorithms used, and therefore FIR's produced by different versions of the FaceVACS-SDK usually are not compatible. You should take precautions against problems with upgrading to a new FaceVACS-SDK version by storing the enrollment images along with the FIR, which allows for creating an updated FIR without calling on the user again.

0.7.4 FAR, FRR and Score

The result of a comparison between 2 FIR's is a score value in the range [0,1], higher values representing more similar feature sets. Due to inevitable differences between the samples, for example caused by varying facial expressions, poses or lighting conditions, comparing different FIR's of the same person will not yield maximum score values in any case. On the other hand, FIR's created from different persons might yield rather high score values, when they indeed are similar (e.g. there are 0.4 % mono-zygotic twins in the population). Hence, the result of face recognition can only be expressed in terms of probability. To describe the probabilities involved usually the terms **FAR** and **FRR** are used. **FAR (False Acceptance Rate)** is the probability that a sample **falsely matches** the presented FIR, and **FRR (False Rejection Rate)** is the probability that a sample of the right person is **falsely rejected**. The relation between these rates is controlled by the acceptance threshold of the system: if this threshold is set to the level of the highest possible score values, there will be no more false acceptances (FAR = 0), but it will also be impossible to be truly accepted, because even high score values do not match the upper limit, i.e. all attempts to match ones own enrollment will cause false rejections (FRR = 1). Setting the threshold to the lowest comparison value possible will create the inverted case: there are no more false rejections, because all comparison attempts yield "true", so FAR = 1 and FRR = 0 in this case. Plots of both FAR and FRR against a supposed threshold value are a commonly used method for ratings of biometric systems. The value of FAR and FRR at the point where the plots cross is called the **Equal Error Rate (EER)**.

Usually, an analysis of the biometric use case of a given application will result in either an FAR or FRR value to start with. Whereas for an access control system for a company with numerous employees the maximum FRR (with a still reasonable FAR) might be the starting point for the security adjustment in order to prevent long queues in the rush hour, for an application in a high-security area with a small group of authorized persons it might be the FAR. To support these different use cases, the FaceVACS-SDK provides two global functions FRsdk::requestFAR() and FRsdk::requestFRR(). The return value of these functions is a FRsdk::Score, which models the score described above. The returned scores can be used as thresholds for verifications and identifications. In addition, the FaceVA-CS-SDK provides two functions (FRsdk::expectedFAR() and FRsdk::expectedFRR()) to get the expected FAR and FRR from a score value returned by verification and identification. The mappings from FAR/FRR values to scores are designed in a way that they independently of technology changes produce defined score values for defined values of the FAR (see table below):

Score	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
FAR	1.0	$10^{-0.5}$ (= 0.-316)	10^{-1} (= 0.1)	$10^{-1.5}$ (= 0.-0316)	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	0.0

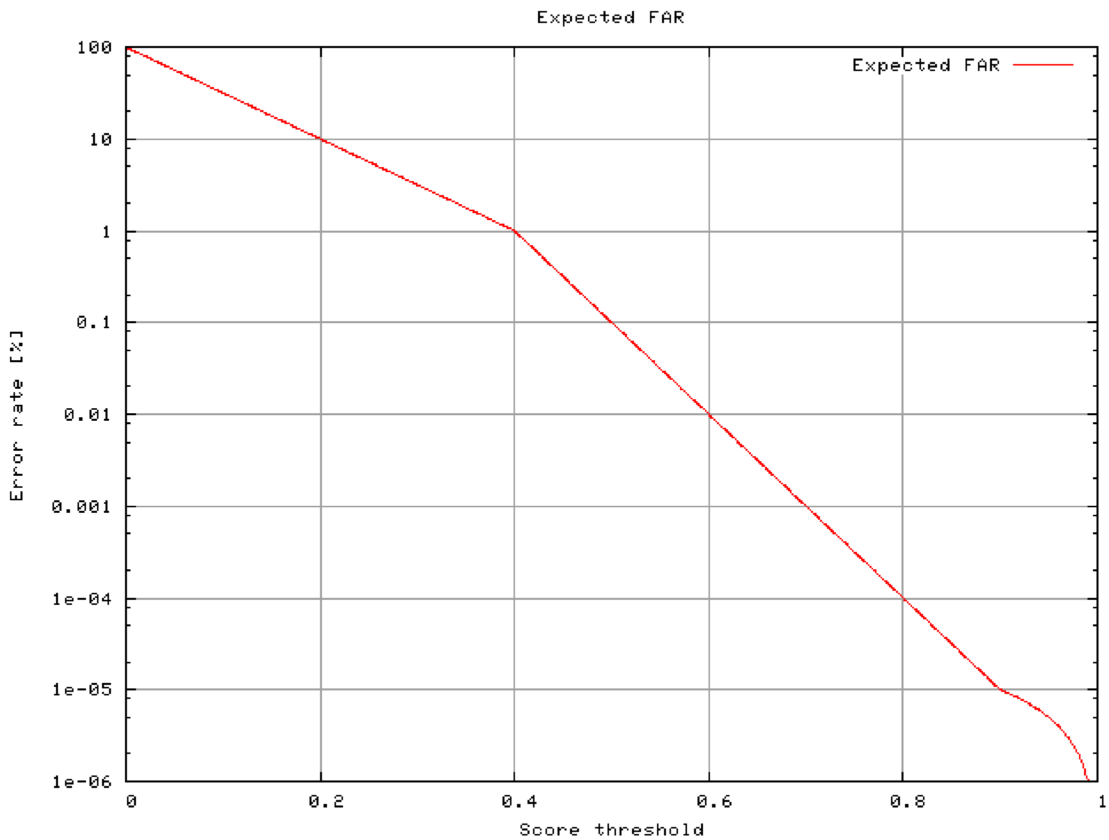


Figure 1: FAR mapping

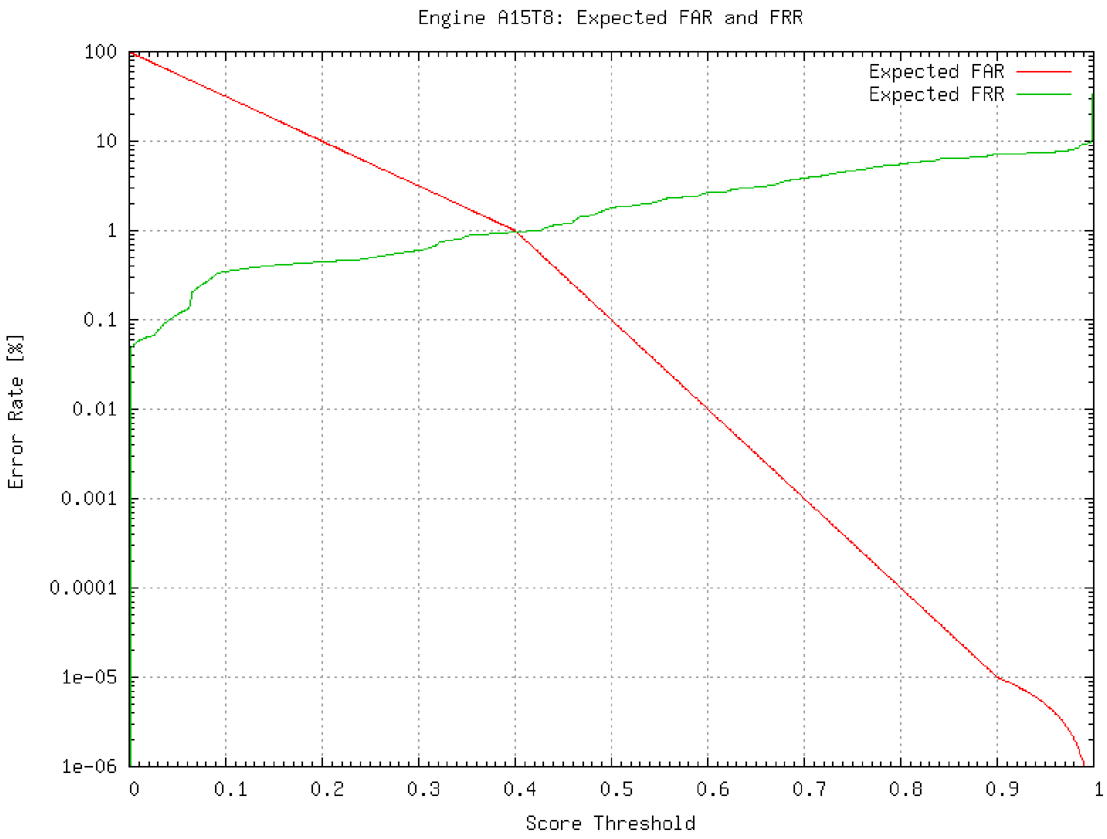


Figure 2: FAR / FRR rates mapping of A15T8

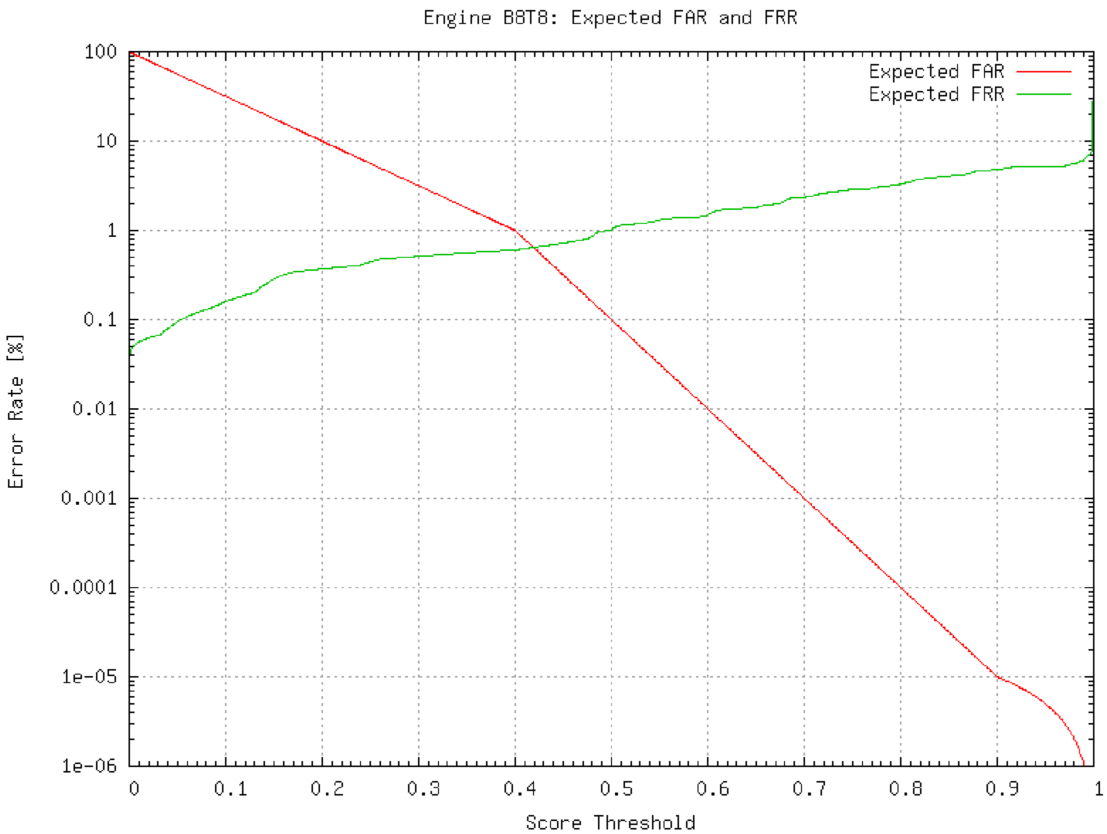


Figure 3: FAR / FRR rates mapping of B8T8

The mappings are derived from results that we achieved on a large test database with high quality passport style photos. For a different database, the relationship between scores and error rates can be different.

0.7.5 How to measure biometric performance

To measure FAR/FRR curves, one starts from a set of N images with known subject identities.

Now each image of the set is compared against each other image, resulting in N(N-1) score values (counting each comparison of 2 images only once). We can divide the set of scores into 2 subsets: a set G resulting from comparisons of images of the same persons (genuine scores) and a set I resulting from comparisons of images of different persons (impostor scores).

Next an appropriate subset T is chosen from the range of possible score values (for FaceVACS, that range is the interval [0,1]).

'Appropriate' here means that the scores in T should be dense enough to allow calculation of biometric curves at some desired granularity.

For example, for the range [0,1] one might choose T to contain the 1000 score values 0.001, 0.002, ... 1.000.

Now to compute the **FAR**, for each score t in T one computes the fraction of scores in G that exceed t, that is the fraction of impostor scores above the threshold t.

To compute **FRR**, for each score t in T one computes the fraction of scores in I that are less than t, that is the fraction of genuine scores which are below the threshold t.

Note that the function which results from subtracting FRR from 1 is called **Verification Rate** VR(t):

VR(t) = 1.0 - FRR(t)

To describe the performance of a biometric system, one often uses another curve which is called **ROC** (Receiver

Operating Characteristics). This results from representing VR as a function of FAR. It is obtained by plotting the points (FAR(t),VR(t)) for all t and connecting adjacent points with straight lines (i.e. applying linear interpolation between neighboring points).

0.7.6 Biometric performance and working point

FAR/FRR data are obtained by statistical measurements. Even for one and the same biometric system you will get different data dependent on the use case and the quality of the sample. A finger print system will have a different (supposedly better) performance when used in an office than on a construction site. Similarly, face recognition performance varies with the number and the quality of the enrollment images and the lighting conditions in both enrollment and verification or identification.

To get the best performance, the working point of a FaceVACS-SDK application has to be adjusted, i.e. a score value has to be defined which is to be used as the threshold.

To get an initial estimate for an appropriate score threshold value or ranges to be used, according to your application scenario choose a desired FAR or FRR value and use the plots form section FAR, FRR and Score to read off the corresponding score threshold. Alternatively, you can use the SDK's face comparision algorithm functions requestFAR()/requestFRR() to compute the score threshold which should provide the desided FAR/FRR for your application.

In many cases, you will still have to fine-tune these values according to the comparison results achieved in your installation. Generally, you will start with FAR/FRR values which are to the right of the EER point, since in most cases the FRR at the EER level is much lower than what might be tolerated by the user, whereas the FAR at the EER level is not yet acceptable.

The biometric performance strongly depends on the quality of the enrollments. Also changing environmental conditions and variations of personal appearance may influence the reliability of recognition.

A detailed description of how the FaceVACS-SDK face recognition algorithms work can be found in the document "FaceVACSalgorithms.pdf" in the "doc" sub-directory of the FaceVACS-SDK installation.

0.7.7 Enrollment procedure

A qualified enrollment can substantially increase the performance of the face recognition. Here are some hints how to improve the quality of the enrollments:

- Use several images of the face, showing slightly different views. This can be accomplished either by asking the user to make slight face movements (10-15 degrees in horizontal and vertical direction) or by using an appropriate arrangement of several cameras. 8-12 different images should do.
- Inform the user that taking enrollment pictures is not the same as making a portrait with the photographer. So they should neither pose nor smile, but show an everyday face expression instead which is likely to be the same which they have upon verification.
- If persons wearing glasses experience difficulties to be recognized, merge samples with and without glasses in the enrollment or ask the person to do both enrollment and recognition without glasses.

0.7.8 Influence and arrangement of lighting conditions

In circumstances where you can control the lighting conditions, you should pay attention to the following requirements:

- Lighting of the face should be either diffuse or directed frontal light to avoid volatile shadows within the face region.
- Avoid lighting producing glare in glasses or on shiny skin
- Face illumination should not vary with the position or height of the user

- Watch for automatic gain control features of the cameras used which might be misleading by bright spots within the image or by bright or dark clothes
- Protect the biometric area against additional lighting, especially sunlight
- Lighting conditions should be quite similar if not identical in enrollment and verification situation.

In many cases, fluorescent tubes located to the left and the right of the camera provide a proper illumination. If they are positioned off-center at a degree of approximately 30-45 degrees, there is no glare in glasses. If there are no variations in the light intensity across time, the automatic gain control feature of the cameras can be switched off, adjusting them manually for optimal reproduction of the face area.

For a more exhaustive description of influences of lighting conditions and how to acquire image for biometric processing read the document "imgguide/imgguide.pdf"

0.7.9 Portrait Characteristics and Compliance Tests

Support for ISO/IEC 19794-5

Even images taken properly according to the recommendations for lighting conditions, pose and expression can fail to produce good scores if the image quality within the face region is bad.

To allow for evaluation of the image quality FaceVACS-SDK provides tools for measuring characteristics of a portrait image and testing general quality parameters in order to support portrait acquisition processes compliant with ISO/-IEC 19794-5.

ISO/IEC 19794-5, Biometric Data Interchange Formats - Part 5: Face Image Data specifies various scene, photographic and digital properties a portrait must have to be compliant with this standard.

FaceVACS-SDK supports both measuring these properties as well as testing their compliance with the standard's constraints. FaceVACS-SDK's namespaces FRsdk::Portrait and FRsdk::ISO_19794_5 contain the corresponding classes:

Use FRsdk::Portrait::Analyzer to analyze a portrait (a face image with annotated eyes positions) and create FRsdk-::Portrait::Characteristics representing quality measures needed for further testing.

Use FRsdk::ISO_19794_5::FullFrontal::Test to test compliance of the portrait characteristics with the ISO standard's requirements for Full Frontal images. FRsdk::ISO_19794_5::FullFrontal::Test::assess returns an object of class F-Rsdk::ISO_19794_5::FullFrontal::Compliance representing the result of the assessment. If FRsdk::ISO_19794_5-::FullFrontal::Compliance::isCompliant returns true the portrait successfully passed the test. In that case all other members of FRsdk::ISO_19794_5::FullFrontal::Compliance return true as well. If isCompliant returns false the other members provide hints about which properties of the portrait failed to meet the standard's requirements.

A portrait passing the Full Frontal test is suitable for being processed by the FaceVACS comparison algorithms.

Not in all scenarios it is really required that all portrait characteristics must pass the compliance check of a full frontal image. I some use cases some characteristics can be omitted or the boundaries of compliance tests can be less restrictive. This has to be decided depending of the project and should be based on some investigations which characteristics can be handled less restrictive.

The class FRsdk::ISO_19794_5::FullFrontal::Boundaries contains access member which limits the compliance test of a special characteristics has to be passed. Each of such limit can be adapted using the Configuration Editor.

The ISO standard's Token Face image type used to store extracted face image information is supported as well. FaceVACS-SDK contains functions for extracting, reading and writing Token Face images in namespace FRsdk::I-SO_19794_5::TokenFace.

In addition to the compliance test FaceVACS-SDK offers detection of other portrait properties not required by the IS-O standard. In current implementation Class FRsdk::Portrait::Feature::Test provides test for wearing glasses only. In future releases other properties might be added. Measured on the well known FERET (frontal faces only) database the glasses detector has the following performance for the pre-configured working point:

- FAR (false glasses detection): nearly zero
- FRR (undetected but available glasses): lower 1%

0.7.10 Orientation of Samples

Orientation of the facial samples means in this context whether the image is mirrored or not. If you view an image of yourself in a plane mirror, you will notice that there is a left-right reversal of the image. That is, if you close your left eye, the mirror person closes his or her right eye, and if you close your right eye, the mirror person closes his or her left eye. Usual digital cameras produce non-mirrored images, some of them, however, as well as many video sources, can be configured to transform images to mirrored ones.

Even though orientation of images is not relevant for core FaceVACS face recognition algorithms it does become relevant in verification and identification use cases. Make always sure that orientation is uniform across all face samples used for verification and identification. This includes also the FIRs generated as gallery items for identification or as reference FIR for verification.

For information about geometric orientation of facial images see the documentation of FRsdk::Position.

[Previous chapter] [Next chapter]

0.8 User Guide - Creating Applications with FaceVACS-SDK

[Previous chapter] [Next chapter]

0.8.1 Build Environment

The directory tree of the installed FaceVACS-SDK distribution looks like this:

```
- <install root>   ... the root directory of the FaceVACS-SDK installation
   |
   +- bin          ... directory containing additional utilities
   |  +- x86_32    ... for 32 bit platforms
   |  +- x86_64    ... for 64 bit platforms
   |  +- ubin      ... universal binaries on Mac OS X
   |                   (includes 32 and 64 bit)
   |
   +- doc          ... the documentation directory. Here the online
   |  |                and pdf version of the FaceVACS-SDK manual can
   |  +- html          be found. This includes user guide, API reference and
   |                   tutorial parts
   |
   +- etc          ... this directory contains some templates for the
   |                   algorithmic parts of FaceVACS-SDK and the
   |                   configuration file (frsdk.cfg)
   |
   +- examples     ... location for the examples
   |  |
   |  +- cpp       ... C++ sources
   |  |  +- x86_32 ... executable for 32 bit platforms
   |  |  +- x86_64 ... executable for 64 bit platforms
   |  |  +- ubin   ... universal binaries on Mac OS X
   |  |
   |  +- cs        ... C# sources
   |  |  +- x86_32 ... executable for 32 bit platforms
   |  |  +- x86_64 ... executable for 64 bit platforms
   |  |  +- ubin   ... universal binaries on Mac OS X
   |  |
   |  +- images    ... images used by the examples
   |
   +- hasp-driver  ... hasp driver (Dongle protection framework)
   |  |
   |  +- x86_32    ... for 32 bit platforms
   |  +- x86_64    ... for 64 bit platforms
   |  +- ubin      ... universal binaries on Mac OS X
   |
   +- include      ... the header files
   |  |
   |  +- frsdk     ... the c++ header files for FaceVACS-SDK
   |     +- bioapi ... the c header files for the bio api library
   |
   +- lib          ... the location of FaceVACS-SDK libraries
   |  |               *.lib/*.dll files for Windows dynamic link libraries,
   |  |               *.lib files for Windows static libaries,
   |  |               *.so for Linux and MacOS shared libaries)
   |  |
   |  +- x86_32    ... libraries for x86 32 bit platform
```

```
|   |   |
|   |   +- msc_12.0-ipp_crtdll ... for VC++ 12.0 (Visual C++ 2013) with IPP
|   |   +- msc_12.0-ipp-crtdll_g ... for VC++ 12.0 (Visual C++ 2013) debug
|   |   +- msc_11.0-ipp_crtdll ... for VC++ 11.0 (Visual C++ 2012) with IPP
|   |   +- msc_11.0-ipp-crtdll_g ... for VC++ 11.0 (Visual C++ 2012) debug
|   |   +- msc_10.0-ipp_crtdll ... for VC++ 10.0 (Visual C++ 2010) with IPP
|   |   +- msc_10.0-ipp_crtdll_g ... for VC++ 10.0 (Visual C++ 2010) debug
|   |   +- gcc-4.6-ipp       ... for Linux gcc 4.6 with IPP
|   |   +- gcc-4.0-ipp       ... for Mac OS X gcc 4.0 with IPP
|   |   +- gcc-4.2-ipp       ... for Mac OS X gcc 4.2 with IPP
|   |   +- gcc-4.3-ipp       ... for Linux gcc 4.3 with IPP
|   |   +- share             ... common libraries (IPP, MKL, Xerces, Qt)
|   |
|   +- x86_64                ... libraries for x86 64 bit platforms
|   |   |
|   |   +- msc_12.0-ipp_crtdll ... for VC++ 10.0 (Visual C++ 2013) with IPP
|   |   +- msc_12.0-ipp_crtdll_g ... for VC++ 10.0 (Visual C++ 2013) debug
|   |   +- msc_11.0-ipp_crtdll ... for VC++ 11.0 (Visual C++ 2012) with IPP
|   |   +- msc_11.0-ipp_crtdll_g ... for VC++ 11.0 (Visual C++ 2012) debug
|   |   +- msc_10.0-ipp_crtdll ... for VC++ 12.0 (Visual C++ 2010) with IPP
|   |   +- msc_10.0-ipp_crtdll_g ... for VC++ 12.0 (Visual C++ 2010) debug
|   |   +- gcc-4.6-ipp       ... for Linux gcc 4.6 with IPP
|   |   +- gcc-4.2-ipp       ... for Mac OS X gcc 4.2 with IPP
|   |   +- gcc-4.3-ipp       ... for Linux gcc 4.3 with IPP
|   |   +- share             ... common libraries (IPP, MKL)
|   |
|   +- ubin                  ... universal binaries for Mac OS X (32 & 64 bit)
|       |
|       +- gcc-4.2-ipp       ... for Mac OS X gcc 4.2 with IPP
|       +- gcc-4.0-ipp       ... for Mac OS X gcc 4.0 with IPP
|
+- tools       ... evaluation tools
    |
    +- x86_32                ... for 32 bit platforms
    +- x86_64                ... for 64 bit platforms
    +- ubin                  ... universal binaries for Mac OS X
```

The FaceVACS-SDK DLL's (shared libraries) are delivered in different versions. For Linux and Windows we provide libraries and executables for the x86 32 bit platform and for the x86 64 bit platform. For Linux there is a non-debug version only (there is no need for a debug version). For MS Windows both debug and release versions are provided to allow for creation of debug mode executables. Furthermore, there are different versions of the libraries built with Visual C++ 12.0 (aka Visual C++ 2013), Visual C++ 11.0 (aka Visual C++ 2012) and Visual C++ 10.0 (aka Visual C++ .NET 2010). For Mac OS X we provide the dynamic library as universal binary which includes the 32 and 64 bit code.

The Win32/64 libraries are dynamically linked to the C run-time import libraries provided by the compiler. In Win32/64, you have to indicate the type of C run-time library when **compiling** your application's object files (see below).

The SDK import libraries are located below the "lib" and the platform installation sub-directory in directories named after the compiler versions. To differentiate between release and debug versions the debug versions of the libraries contain an additional "d" at the end of their name. The names of the libraries contain the version number of the FaceVACS-SDK in form of <version> which stands for the actual product version. The following table shows the naming conventions for the directories and the names of the library files of the FaceVACS-SDK distribution:

	Linux	**Mac** OS X	**Windows**	
Platform sub-directories	x86_32, x86_64, ubin	x86_32, x86_64		
library/executable subdirs	gcc-4.6, gcc-4.6-ipp, gcc-4.3 or gcc-4.3-ipp, gcc-4.4 (depending on whether to use IPP and which compiler version)	gcc-4.2-ipp	msc_12.0-ipp_-crtdll, msc_12.0_crtdll_g, msc_11.0-ipp_-crtdll, msc_11.0_crtdll_g, msc_10.0-ipp_-crtdll, msc_10.0_crtdll_g (depending on whether to use debug and which compiler version)	emsc_8.0_crtdll
import libraries (to link against)	libfrsdk-<version>.so	libfrsdk-<version>.so	libfrsdk-<version>{d}.lib	libfrsdk-<version>.lib

To compile and link applications using the FaceVACS-SDK, you have to

- add the "include" sub-directory of the FaceVACS-SDK installation to the include search path of the compiler.
- add the (full path of the) relevant FaceVACS-SDK library sub-directory (msc_12.0-ipp_crtdll/msc_11.0-ipp_crtdll/msc_10.0-ipp_crtdll or msc_12.0-ipp_crtdll_g/msc_11.0-ipp_crtdll_g/msc_10.0-ipp_crtdll_g) to the search path of the linker.
- add the FaceVACS-SDK library to the linker input list

(Win32/64 only)

The best way to get a proper project configuration is to start with one of the example project files located within examples\cpp.

You also can set the configuration settings manually within your project configuration. Open the project configuration dialog in the menu via **Project->"name" properties**

(where 'name' is the project name of your application).

The additional include path is to be entered into

Configuration Properties->C/C++->General->Additional Include Directories.

The library search path can be entered in **Configuration Properties->Linker->General->Additional Library Directories**.

The SDK library name is to be entered into

Configuration Properties->Linker->Input->Additional Dependencies.

There are some more settings which have to be configured appropriately to link with the FaceVACS-SDK library.

If you want to build a release mode version of your application, you have to:

- compile your code with the "Multithreaded DLL" C Runtime option setting
- link your application with the no-debug version of the FaceVACS-SDK DLL.

If you want to build a debug mode version of your application, you have to:

- compile your code with the "Multithreaded Debug DLL" C Runtime option setting
- link your application with the debug version of the FaceVACS-SDK DLL.

The C Runtime compile option is to be set in

Configuration Properties->C/C++->Code Generation->Runtime Library.

For all types of applications 2 additional settings are important:

- Enable C++ exceptions in **Configuration Properties->C/C++->Code Generation->Enable C++ Exceptions**.
- Enable Runtime Type Information (RTTI) in **Configuration Properties->C/C++->Language->Enable Run-Time Type Info**.

0.8.2 Run-Time Environment for FaceVACS-SDK Applications

For applications developed with the FaceVACS-SDK the runtime libraries of the SDK have to be available. For the Windows32 platform there are 4 different types of runtime libraries, for the Linux platform there is only one.

DLL/Shared libraries required to run FaceVACS-SDK applications

The following tables shows the DLL's or shared libraries required and additional libraries they depend upon and which are not part of the standard operating system environment or where there are varying versions of:

0.8.2.1 MS Windows

required libraries for Windows platform.

```
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
| FaceVACS-SDK  |                            |          |     |       |                       |                           |
| Library /     | File Name                  | Compiler | Ipp | Debug | Location              | Dependencies              |
| Binaries      |                            |          |     |       |                       |                           |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
|               |                            |          |     |       | /lib/x86_[32/64]/     |                           |
| FaceVACS-SDK  | libfrsdk-<version>.dll     | VC 12.0  | X   |       | msc_12.0-ipp_crtdll   | vc120                     |
| Core          | libfrsdk-<version>d.dll    |          | X   | X     | msc_12.0-ipp_crtdll_g | vc120d                    |
| Library       |                            |          |     |       |                       |                           |
|               | libfrsdk-<version>.dll     | VC 11.0  | X   |       | msc_11.0-ipp_crtdll   | vc110                     |
|               | libfrsdk-<version>d.dll    |          | X   | X     | msc_11.0-ipp_crtdll_g | vc110d                    |
|               |                            |          |     |       |                       |                           |
|               | libfrsdk-<version>.dll     | VC 10.0  | X   |       | msc_10.0-ipp_crtdll   | vc100                     |
|               | libfrsdk-<version>d.dll    |          | X   | X     | msc_10.0-ipp_crtdll_g | vc100d                    |
|               |                            |          |     |       |                       |                           |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
|               |                            |          |     |       | /lib/x86_[32/64]/     |                           |
| Evaluation    | liboutput-<version>.dll    | VC 12.0  | X   |       | msc_12.0-ipp_crtdll   | vc120                     |
| Tools         | liboutput-<version>d.dll   |          | X   | X     | msc_12.0-ipp_crtdll_g | vc120d                    |
| Output Lib    |                            |          |     |       |                       |                           |
|               | liboutput-<version>.dll    | VC 11.0  | X   |       | msc_11.0-ipp_crtdll   | vc110                     |
|               | liboutput-<version>d.dll   |          | X   | X     | msc_11.0-ipp_crtdll_g | vc110d                    |
|               |                            |          |     |       |                       |                           |
|               | liboutput-<version>.dll    | VC 10.0  | X   |       | msc_10.0-ipp_crtdll   | vc100                     |
|               | liboutput-<version>d.dll   |          | X   | X     | msc_10.0-ipp_crtdll_g | vc100d                    |
|               |                            |          |     |       |                       |                           |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
|               |                            |          |     |       | /lib/x86_[32/64]/     | libfrsdk-<version>{d}.dll |
| BIOAPI        | libbiospi-<version>.dll    | VC 12.0  | X   |       | msc_12.0-ipp_crtdll   | + vc120                   |
| Service       | libbiospi-<version>d.dll   |          | X   | X     | msc_12.0-ipp_crtdll_g | + vc120d                  |
| Provider      |                            |          |     |       |                       |                           |
| Interface     | libbiospi-<version>.dll    | VC 11.0  | X   |       | msc_11.0-ipp_crtdll   | + vc110                   |
|               | libbiospi-<version>d.dll   |          | X   | X     | msc_11.0-ipp_crtdll_g | + vc110d                  |
|               |                            |          |     |       |                       |                           |
|               | libbiospi-<version>.dll    | VC 10.0  | X   |       | msc_10.0-ipp_crtdll   | + vc100                   |
|               | libbiospi-<version>d.dll   |          | X   | X     | msc_10.0-ipp_crtdll_g | + vc100d                  |
|               |                            |          |     |       |                       |                           |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
|               |                            |          |     |       | /lib/x86_[32/64]/     | libfrsdk-version{d}.dll   |
| FaceVACS-SDK  | libfrsdknet-<version>.dll  | .NET 4.51| (X) |       | msc_12.0-ipp_crtdll   | + vc120                   |
| .NET Assembly | libfrsdknet-<version>d.dll |          | (X) |       | msc_12.0-ipp_crtdll_g | + vc120d                  |
|               | libfrsdknet-<version>.dll  | .NET 4.5 | (X) |       | msc_11.0-ipp_crtdll   | + vc110                   |
|               | libfrsdknet-<version>d.dll |          | (X) |       | msc_11.0-ipp_crtdll_g | + vc110d                  |
|               | libfrsdknet-<version>.dll  | .NET 4.0 | (X) |       | msc_10.0-ipp_crtdll   | + vc100                   |
|               | libfrsdknet-<version>d.dll |          | (X) |       | msc_10.0-ipp_crtdll_g | + vc100d                  |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
| Configuration |                            |          |     |       |                       |                           |
| Editor        | cfgedit.exe                | VC 10.0  |  X  |       | /bin/x86_[32/64]      | qt, vc100                 |
| Annotation    |                            |          |     |       |                       |                           |
| Tool          | annotator.exe              | VC 10.0  |  X  |       |                       | qt, vc100                 |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
| Tools         | cmcgen.exe,                | VC 10.0  |  X  |       | /tools/x86_[32/64]    | libfrsdk-<version>.dll,   |
|               | ratesgen.exe               |          |  X  |       |                       | liboutput-<version>.dll,  |
|               |                            |          |     |       |                       | vc100                     |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
| XML Tools     | dir2xml.exe                | VC 10.0  |  X  |       | /tools/x86_[32/64]    | xerces,                   |
|               | xmlfirgen.exe              |          |     |       |                       | libfrsdk-<version>.dll,   |
|               | xmlcompsim.exe             |          |     |       |                       | liboutput-<version>.dll,  |
|               | xmlcompmatch.exe           |          |     |       |                       | vc100                     |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
| Cpp Examples  | *.exe                      | VC 10.0  |  X  |       | /tools/cpp/x86_[32/64]| libfrsdk-<version>.dll,   |
|               |                            |          |     |       |                       | vc100                     |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
| Cs Examples   | *.exe                      | VC 10.0  |  X  |       | /tools/cs/x86_[32/64] | libfrsdk-<version>.dll,   |
|               |                            |          |     |       |                       | libfrsdknet-<version>.dll |
|               |                            |          |     |       |                       | vc100                     |
|---------------+----------------------------+----------+-----+-------+-----------------------+---------------------------|
```

Remarks: The lib sub-directories named msc_x.y-ipp_crtdll contain versions of the library linked with IPP support. The Intel IPP DLL's which are required to use these are linked statically into the FaceVACS frsdk libs directory. Therefore, for IPP support the libs are not needed to be added to the PATH anymore

```
|--------------------+---------------------------------------------+----------------------------------------|
| Dependency         | Libraries                                   | Location                               |
|--------------------+---------------------------------------------+----------------------------------------|
|                    |                                             | /bin/x86_[32/64]/                      |
| vc120              | msvcp120.dll, msvcr120.dll                  | + msc_12.0-ipp_crtdll as redistributable|
| vc110              | msvcp110.dll, msvcr110.dll                  | + msc_11.0-ipp_crtdll as redistributable|
| vc100              | msvcm100.dll, msvcp100.dll, msvcr100.dll    | + msc_10.0-ipp_crtdll as redistributable|
| vc120d             | msvcp120d.dll, msvcr100d.dll                | VS 2013, not distributed               |
| vc110d             | msvcp110d.dll, msvcr110d.dll                | VS 2012, not distributed               |
| vc100d             | msvcm100d.dll, msvcp100d.dll, msvcr100d.dll | VS 2010, not distributed               |
|--------------------+---------------------------------------------+----------------------------------------|
| xerces             | xerces-c_2_2_0.dll                          | /lib/x86_[32/64]/share                 |
|--------------------+---------------------------------------------+----------------------------------------|
| qt                 | qt-mt336.dll                                | /lib/x86_[32/64]/share                 |
|--------------------+---------------------------------------------+----------------------------------------|
```

0.8.2.2 Linux

required libraries for platform.

```
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
| FaceVACS-SDK  |                       |          |     |                   |                                |
| Library /     | File Name             | Compiler | Ipp | Location          | Dependencies                   |
| Binaries      |                       |          |     |                   |                                |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
|               |                       |          |     | /lib/x86_[32/64]/ |                                |
| FaceVACS-SDK  | libfrsdk-<version>.so | gcc 4.6  | X   | gcc-4.6-ipp       | ipp[32/64] (static)            |
| Core          |                       |          |     |                   |                                |
| Library       |                       | gcc 4.3  | X   | gcc-4.3-ipp       | ipp[32/64] (static)            |
|               |                       |          |     |                   |                                |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
| CameraDrivers | fg-ueye.so            | gcc 4.3  | X   | /lib/x86_[32/64]/ | ipp[32/64] (static)            |
|               |                       |          |     | gcc-4.3-ipp       |                                |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
| Evaluation    | liboutput-<version>.so | gcc 4.3 | X   | gcc-4.3-ipp       | libfrsdk-<version>.so          |
| Tools         |                       |          |     |                   |                                |
| Output Lib    |                       |          |     |                   |                                |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
|               |                       |          |     | /lib/x86_[32/64]/ |                                |
| BIOAPI        | libbiospi-<version>.so | gcc 4.6 | X   | gcc-4.6-ipp       | libfrsdk-<version>.so          |
| Service       |                       |          |     |                   |                                |
| Provider      |                       | gcc 4.3  | X   | gcc-4.3-ipp       |                                |
| Interface     |                       |          |     |                   |                                |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
| Configuration | cfgedit               | gcc 4.3  | X   | /bin/x86_[32/64]  | qt                             |
| Editor        |                       |          |     |                   |                                |
|               |                       |          |     |                   |                                |
| Annotation    | annotator             | gcc 4.3  | X   |                   | liboutput-<version>.so,        |
| Tool          |                       |          |     |                   | libfrsdk-<version>.so          |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
| Tools         | cmcgen,ratesgen       | gcc 4.3  | X   | /tools/x86_[32/64] | libfrsdk-<version>.so,        |
|               |                       |          |     |                   | liboutput-<version>.so         |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
| XML Tools     | *xml*                 | gcc 4.3  | X   | /tools/x86_[32/64] | xerces, libfrsdk-<version>.so, |
|               |                       |          |     |                   | liboutput-<version>.so         |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
| Cpp Examples  | *                     | gcc 4.3  | X   | /tools/cpp/x86_32 | libfrsdk-<version>.dll         |
| Cpp Examples  | *                     | gcc 4.3  | X   | /tools/cpp/x86_64 | libfrsdk-<version>.dll         |
|---------------+-----------------------+----------+-----+-------------------+--------------------------------|
```

```
|------------+------------------+-------------------------------------------|
| Dependency | Location         | Libraries                                 |
|------------+------------------+-------------------------------------------|
| *          | System           | libpthread.so.0, libdl.so.2, libm.so.6,   |
|            |                  | libgcc_s.so.1, libc.so.6                  |
| x86_32/*   |                  | linux-gate.so.1, /lib/ld-linux.so.2       |
| x86_64/*   |                  | /lib64/ld-linux-x86-64.so.2               |
| */gcc-4.6  |                  | libstdc++.so.6                            |
| */gcc-4.3  |                  | libstdc++.so.6                            |
|------------+------------------+-------------------------------------------|
| qt         | /lib/x86_32/share | libqt-mt.so.3                            |
|------------+------------------+-------------------------------------------|
| xerces     | /lib/x86_32/share | libxerces-c.so.21                        |
|------------+------------------+-------------------------------------------|
```

0.8.2.3 Mac OS X

required libraries for platform.

```
|--------------------+------------------------+----------+-----+-----------------------------+-----------------------|
| FaceVACS-SDK       |                        |          |     |                             |                       |
| Library /          | File Name              | Compiler | Ipp | Location                    | Dependencies          |
| Binaries           |                        |          |     |                             |                       |
|--------------------+------------------------+----------+-----+-----------------------------+-----------------------|
|                    |                        |          |     | /lib/[x86_32/x86_64/ubin]/  |                       |
| FaceVACS-SDK       | libfrsdk-<version>.so  | gcc 4.2  | Ipp | gcc-4.2-ipp                 |                       |
| Core Library       |                        |          |     |                             |                       |
|--------------------+------------------------+----------+-----+-----------------------------+-----------------------|
|                    |                        |          |     | /lib/ubin/                  |                       |
| Evaluation         | liboutput-<version>.so | gcc 4.2  | Ipp | gcc-4.2-ipp                 | libfrsdk-<version>.so |
|                    |                        |          |     |                             |                       |
|--------------------+------------------------+----------+-----+-----------------------------+-----------------------|
|                    |                        |          |     | /lib/ubin/                  |                       |
| BIOAPI Service     | libbiospi-<version>.so | gcc 4.2  | Ipp | gcc-4.2-ipp                 | libfrsdk-<version>.so |
| Provider Interface |                        |          |     |                             |                       |
```

```
|-------------------+-----------------------+----------+-----+----------------------------+----------------------|
| Cpp Examples      | *                     | gcc 4.2  | Ipp | /tools/cpp/ubin            | libfrsdk-<version>.so |
|-------------------+-----------------------+----------+-----+----------------------------+----------------------|
```

```
|------------+-------------------+------------------------------------|
| Dependency | Location          | Libraries                          |
|------------+-------------------+------------------------------------|
| *          | System            | /usr/lib/libstdc++.6.dylib         |
|            |                   | /usr/lib/libSystem.B.dylib         |
|------------+-------------------+------------------------------------|
```

0.8.2.4 DLL/Shared library path configuration

The location of the FaceVACS-SDK DLL's resp. shared libraries must be known to the system when starting a FaceVACS-SDK application. In the next sections it is described how to ensure this for Windows and Linux, respectively.

MS Windows

For MS Windows platforms, there is a system defined search order for DLL's:

```
- The directory of the executable for the process that is loading the DLL
- The current directory of the process that is loading the DLL
- The \WINNT\SYSTEM32 directory
- The \WINNT directory
- The PATH environment variable for the process
```

(all those are searched only unless the DLL's name is not contained in the registry entry

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\KnownDlls
```

which supersedes all of them).

Since it is generally a bad idea to put DLL stuff specific to a given application into the system directories, we recommend either to put the FaceVACS-SDK DLL's along with the application binaries into a common directory or to set the PATH variable accordingly. For testing binaries in the MSVC debugger during development phase, you also can use the "Program Working Directory" setting in the "Debug" tab of "Project Settings" to set it to the directory where the FaceVACS-SDK DLL's reside.

```
Note: Please watch for providing the correct version
(Debug/Release, Toolchain) of runtime libraries to your executable.
Providing wrong versions can cause strange errors.
```

To change the PATH environment variable, invoke the "System" applet of the Control Panel. Select the PATH system variable (creating it before if not yet existing) and push "Edit". To the current setting, append a ";" and the complete path name name to the appropriate directory. If you dislike the idea of changing the PATH for all executable running on the system, you can make a cmd-script, which adjusts the PATH variable and then starts the FaceVACS-SDK application.

Since the FaceVACS-SDK DLL's in turn use the C run-time DLL's provided by the compiler, they too have to be in the DLL search path of your compiler. On the computer where your development environment resides, usually all necessary C run-time libraries have been installed. You are responsible to provide the necessary runtime environment to your customers in your deployment procedure.

For debug mode applications built with MSVC120, MSVC110 or MSVC100 you will need the Debug versions of the C runtime libraries, which are named "msvcr90d.dll", "msvcp90d.dll" or "msvcr80d.dll", "msvcp80d.dll", msvcm80d.dll. These should be part of your Visual C++ .NET installation and reside in the system32 directory or in some other place where they had been installed to upon Visual Studio .NET Installation.

Linux

For Linux the path to the compiler specific FaceVACS-SDK shared library (lib/[compiler])and the path to the compiler independent library dir (lib/share) must be in the LD_LIBRARY_PATH environment variable of the application to be run. You can set the environment variable for a given account in the file .profile or .login (depending on the login shell), which resides in the home directory of this account. Create the file if it is not there already.

In the description below '<FVSDKDIR>' is used as a placeholder for the full path name of your FaceVACS-SDK installation.

If the login shell is one of **bash, ksh, zsh or sh**, add the following lines to the .profile file:

- for 32 bit environment:

```
LD_LIBRARY_PATH=<FVSDKDIR>/lib/x86_32/gcc-4.6-ipp:\
                <FVSDKDIR>/lib/x86_32/share:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

- for 64 bit environment:

```
LD_LIBRARY_PATH=<FVSDKDIR>/lib/x86_64/gcc-4.6-ipp:\
                <FVSDKDIR>/lib/x86_64/share:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

If the login shell is one of **csh or tcsh**, add the following lines to the .login file:

- for 32 bit environment:

```
setenv LD_LIBRARY_PATH <FVSDKDIR>/lib/x86_32/gcc-4.6-ipp:\
                       <FVSDKDIR>/lib/x86_32/share:$LD_LIBRARY_PATH
```

- for 64 bit environment:

```
setenv LD_LIBRARY_PATH <FVSDKDIR>/lib/x86_64/gcc-4.6-ipp:\
                       <FVSDKDIR>/lib/x86_64/share:$LD_LIBRARY_PATH
```

After applying the changes, you will have to login again, or re-source the profile before continuing, in order to make the settings effective.

Mac OS X

For Mac OS X the path to the compiler specific FaceVACS-SDK shared library (lib/[compiler]) must be in the DYL-D_LIBRARY_PATH environment variable of the application to be run (see Linux for further instructions).

0.8.3 FaceVACS-SDK application configuration

Every application using the FaceVACS-SDK runtime libraries must have access to a FaceVACS-SDK configuration file containing a valid license. In addition to the license, this file contains settings regarding the whole installation as well as settings to control particular software functionality.

After installation, there is a configuration file **frsdk.cfg** in the **etc** installation subdirectory.

You can view and change configuration settings with the Configuration Editor.

- **Linux:** Start Configuration Editor with the command **cfgedit -f** <**path to your configuration file**>. Note that the LD_LIBRARY_PATH setting must contain the Linux library directory of your SDK installation, as explained above.

- **Windows:** There is an entry in the Start Menu to start the Configuration Editor: **Start->Programs->FVSD-K_X_Y->Configuration Editor**

The settings are visible in a tree view control in the central pane of the Editor.

To change any settings, expand the tree view and select the item to be changed. Invoke the editing dialog by clicking the item with the **right** mouse button.

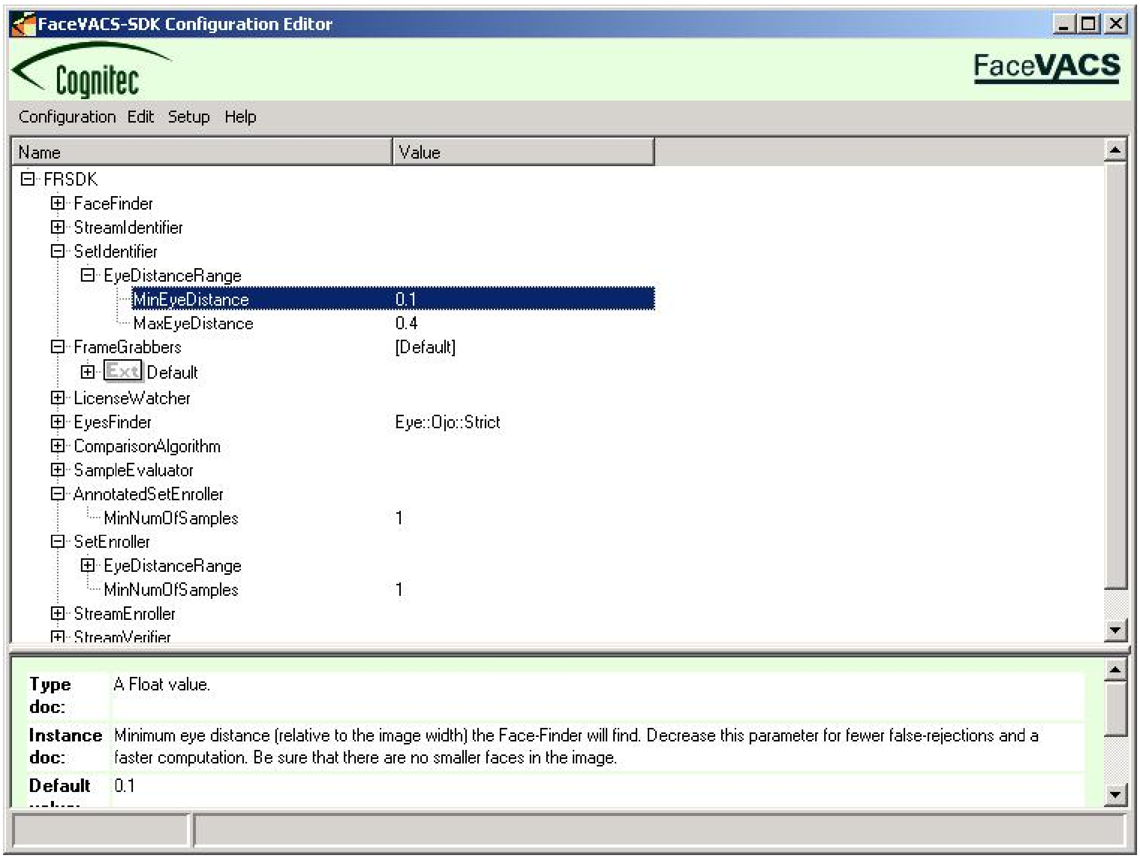


Figure 4: Expanded Settings Tree in Configuration Editor

The settings visible in the pane are explained below (they are arranged here by topic, not in the order of the settings view):

0.8.3.1 Installation Settings

Installation Directory

All algorithms depending strongly on a well configured installation directory. For Win32/64 platforms, the installation procedure will properly set these values.

On Linux is has to be done manually. The place holder MAINDIR has to be replaced by the installation directory. Use Configuration Editor or perl to accomplish this.

hint: Escape the directory name separator with \ in order to replace MAINDIR with the complete path.

Example: *perl -pi.bak -e "s/MAINDIR/\/home\/john\/FVSDK/g" etc/frsdk.cfg*

Remark: In former versions of FaceVACS-SDK the placeholder was part of a lot of keys. This has changed now, because of a new substitution feature in the configuration system of FaceVACS-SDK. There are only two keys to modify:

- **FRSDK.Substitutions.INSTALLDIR** = MAINDIR
- **CorporateIdentity.LogoPath.default** = MAINDIR/etc/pict

The first is the most important and set the installation path for all required templates.

0.8.3.2 Configuration of Comparison Algorithm

By default, FaceVACS-SDK is configured to perform intensity image processing only using the FaceVACS B8 matching algorithm. The configuration key controlling the type of the comparison algorithm involved is

FRSDK.ComparisonAlgorithm

By right-clicking this leaf within Configuration Editor, you can choose the algorithm type to be used:

- B8ComparisonAlgorithm (default)
- A15ComparisonAlgorithm
- CompositeComparisonAlgorithm (3D algorithm)

Don't change this setting unless you really intent to use A15 or shape data in your application (Composite-ComparisonAlgorithm). Note that in case of CompositeComparisonAlgorithm each FRsdk::Sample provided has to contain both intensity and shape data (see Input Data - Samples) FIRs created with different settings of the comparison algorithm type are not compatible.

B8 and A15 algorithm are similar but differ in some points:

- size of FIRs
- biometric performance
- speed of comparison

The following table contains size of a FIR in memory (memory footprint) and in platform independent format (package size).

Algorithm	memory footprint (1 Image)	package size (1 Image)	memory footprint (5 Images)	package size (5 Images)
B8	2720	2833	13584	13985
A15	1424	1547	7104	7563

The difference in biometric performance can be seen in diagrams of section FAR, FRR and Score.

In general B8 is more precise than A15. A15 has a smaller memory footprint. B8 requires more calculation power and memory throughput due to the bigger memory consumption. How identification speed differs depends strongly from the chosen platform (CPU, memory controller and memory chips).

0.8.3.3 Face Finder Settings

- The settings for the minimum and maximum eye distance of the face finder describes the range of faces in percent relative to the image width which are searched for. A value of 0.1 for the MinEyeDistance and a value of 0.4 for MaxEyeDistance parameter (the default settings) sets the range of faces to be searched for to 10% ... 40% of the image width. This parameters are runtime arguments of the FRsdk::Face::Finder::find() function. Note that those settings do not imply that any face with an eye distance outside that range cannot be found; slightly smaller and larger faces in the image can also be detected.
- However, for the Processor classes using face finders internally, these properties can now be configured as static properties. See below.
- In addition, there are a lot of internal algorithmic parameters of the face finder visible in the Configuration Editor. These will not be documented and should not be touched unless directed otherwise.

0.8.3.4 Eyes Finder Settings

- The eyes finder is configurable to use a **'strict'** or a **'tolerant'** parameter set. Tolerant in that manner means that the false rejection rate is very low, even noisy or blurred eyes will be found most of the time. The price to pay is a high false acceptance rate, i.e. non-eye image sections are often accepted as eyes. The strict setting could be considered as a good compromise between the error rates: a somewhat higher but still low false rejection rate is combined with a low false acceptance rate. The configuration can be changed by right-clicking the **EyesFinder** sub-node and choosing the appropriate parameter set.

- Both parameter sets present some low-level algorithmic parameters. Like with the face finder, they will not be documented here and should not be touched unless directed otherwise.

In addition you can configure static properties of the FaceVACS-SDK biometric transaction processors:

0.8.3.5 FRsdk::Enrollment::Processor Settings

- **FRSDK.SetEnroller.EyeDistanceRange.MaxEyeDistance** = [positive float]
  - Hint for the face finding algorithm for the maximum eye distance to be searched for (relative to the width of the image)
  - will internally be used as the 3rd argument of the FRsdk::Face::Finder::find() function
  - Default setting: 0.4
- **FRSDK.SetEnroller.MinNumOfSamples** = [positive integer]
  - When calling the process() interface of the Enrollment::Processor taking an image set as the argument, it can happen that in one or more images of the set no face and eye can be found. This parameter defines the minimum number of images where eyes have to be found.
- **FRSDK.StreamEnroller.EyeDistanceRange.MinEyeDistance** = [positive float]
  - Hint for the face finding algorithm for the minimum eye distance to be searched for (relative to the width of the image)
  - will internally be used by the face finder facility
  - Default setting: 0.1
- **FRSDK.StreamEnroller.EyeDistanceRange.MaxEyeDistance** = [positive float]
  - Hint for the face finding algorithm for the maximum eye distance to be searched for (relative to the width of the image)
  - will internally be used by the face finder facility
  - Default setting: 0.4
- **FRSDK.StreamEnroller.MinNumOfSamples** = [positive integer]
  - When calling the process() interface of the Enrollment::Processor taking a CaptureDevice as the argument, it can happen that in one or more images captured from the stream no face and eye can be found. This parameter defines the minimum number of images where eyes have to be found during the stream enrollment.
  - Default setting: 1
- **FRSDK.StreamEnroller.NumOfAcquisitions** = [positive integer]
  - Determines the number of images captured from the stream during enrollment.
  - Default setting: 8
- **FRSDK.StreamEnroller.TotalEnrollmentTime** = [positive integer]
  - Defines the maximum time in msecs the enrollment can last. Once this period has expired, the enrollment will terminate.
  - Default setting: 60000
- **FRSDK.StreamEnroller.MinImageTime** = [positive integer]
  - Defines the minimum time period in msecs between 2 consecutive image captures. This setting is to ensure that on fast machines enrollment images are not fetched too fast, causing them to be almost identical in spite of users head movements.
  - Default setting: 1000

0.8.3.6 FRsdk::Verification::Processor Settings

- **FRSDK.StreamVerifier.EyeDistanceRange.MinEyeDistance** = [positive float]
  - Hint for the face finding algorithm for the minimum eye distance to be searched for (relative to the width of the image)
  - will internally be used by the face finder facility
  - Default setting: 0.1
- **FRSDK.StreamVerifier.EyeDistanceRange.MaxEyeDistance** = [positive float]
  - Hint for the face finding algorithm for the maximum eye distance to be searched for (relative to the width of the image)
  - will internally be used by the face finder facility
  - Default setting: 0.4
- **FRSDK.StreamVerifier.MaxNumOfImages** = [positive integer]
  - When calling the process() interface of the Enrollment::Processor taking a CaptureDevice as the argument, the maximum number of images to be captured for one verification can be limited by this setting.
  - Default setting: 2147483647
- **FRSDK.StreamVerifier.ImagePeriod** = [positive integer]
  - Defines the minimum time period in msecs between 2 consecutive image captures.
  - Default setting: 200
- **FRSDK.StreamVerifier.TotalTime** = [positive integer]
  - Defines the maximum time in msecs a verification can last. Once this period has expired, the verification will terminate.
  - Default setting: 10000
- **FRSDK.StreamVerifier.UseLifeCheck** = [True, False]
  - Whether to apply live check in stream verifications. Prevents intrusion based on photo prints, but requires users to make appropriate face movements.
  - Default setting: False

0.8.3.7 FRsdk::Identification::Processor Settings

- **FRSDK.StreamIdentifier.EyeDistanceRange.MinEyeDistance** = [positive float]
  - Hint for the face finding algorithm for the minimum eye distance to be searched for (relative to the width of the image)
  - will internally be used by the face finder facility
  - Default setting: 0.1
- **FRSDK.StreamIdentifier.EyeDistanceRange.MaxEyeDistance** = [positive float]
  - Hint for the face finding algorithm for the maximum eye distance to be searched for (relative to the width of the image)
  - will internally be used by the face finder facility
  - Default setting: 0.4
- **FRSDK.StreamIdentifier.ImagePeriod** = [positive integer]
  - Defines the minimum time period in msecs between 2 consecutive image captures.
  - Default setting: 200

- **FRSDK.StreamIdentifier.TotalTime** = [positive integer]
  - Defines the maximum time in msecs a stream identification can last. Once this period has expired, the identification will terminate.
  - Default setting: 10000
- **FRSDK.StreamIdentifier.UseLifeCheck** = [True, False]
  - Whether to apply live check in stream identifications. Prevents intrusion based on photo prints, but requires users to make appropriate face movements.
  - Default setting: False

0.8.3.8 FRsdk::Face::Tracker Settings

- **FRSDK.FaceTracker.FaceFinder**
  - Settings of the face finder
- **FRSDK.FaceTracker.EyesFinder**
  - Settings of the eyes finder if used
- **FRSDK.FaceTracker.UseEyesFinder**
  - If true the eyes finder will be used for high precision eyes finding, otherwise only the results of the face finder are used with estimated eye positions
- **FRSDK.FaceTracker.FaceFinderRange**
  - limit the range of faces to be searched by relative eye distance
- **RealEyeDistance**, **Tolerance**, **MaxAllowedFaceMove**, **MaxAllowedFaceSpeed** could be used to setup tracking behaviour

0.8.3.9 Capture Device Configuration

The **createCaptureDevice()** function serves as factory for the creation of instances of configured capture devices referred by their assigned symbolic name. After a new clean installation of the FaceVACS-SDK there is only one pre-configured capture device named **Files** which is of type **ImageFiles**. In contrast with the "real" capture devices, the devices of this type does not acquire images from some imaging hardware but instead imports them from image files. According to your requirements and system configuration you may need to create additional capture devices from different types. We have to mention that the variety of capture device types which can be created depending upon the operating system platform that you are using.

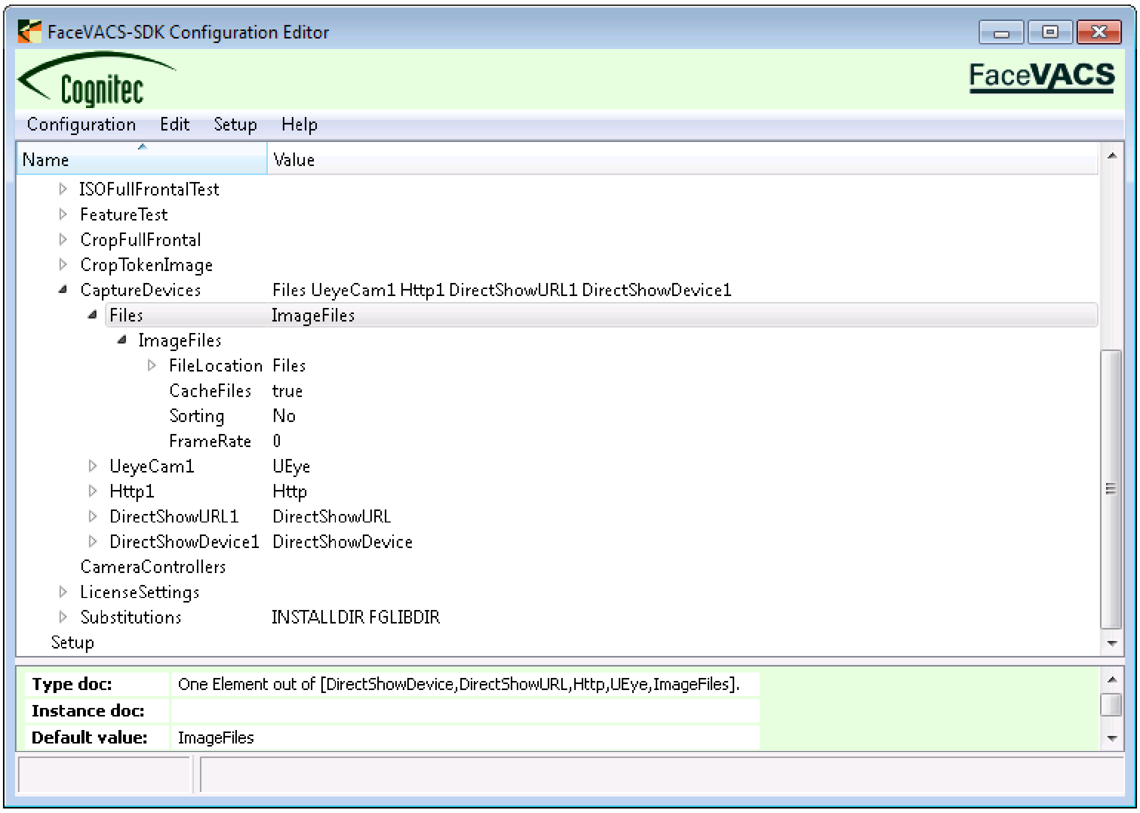


Figure 5: Sample Capture Devices configuration subtree

In order to add new capture devices in your configuration you have to append a corresponding number of user-defined symbolic names for these capture devices to the **FRSDK.CaptureDevices** list:

- **FRSDK.CaptureDevices** = [list of strings]
  - Contains a list of all configured capture devices represented by user-defined symbolic names
  - Default setting: only one pre-configured capture device named "Files"

After you have appended the list of capture devices and created this way a new capture devices with desired from you names, you may also need to change their types. To change the type of the symbolic capture device double click on the root of the tree which represents it:

- **FRSDK.CaptureDevices.[Device Name]**
  - choose the desired new Capture Device Type

Supported Capture Device Types are:

- **ImageFile**
  - Use this type for capture devices which import images from image files.
- **DirectShowDevice**
  - Use this type for devices accessible through the DirectShow API.
- **DirectShowURL**
  - Use this type for DirectShow devices which aquire images from streams accessible through the local file system or through the HTTP protocol.

- **Http**
  - Use this type for devices accessible through the HTTP protocol.
- **UEye**
  - Use this type for IDS uEye cameras.

0.8.3.9.1 ImageFile Capture Device Configuration

As we already mentioned above the ImageFile Capture Devices are useful for importing from image files. This type of capture device is available on all supported Windows, Linux and OS X platforms (both 32-bit and 64-bit).

- **FRSDK.CaptureDevices.[Device Name].ImageFiles.FileLocation** = [list of strings]
  - Contains a list of all configured files to be imported represented by user-defined symbolic names
- **FRSDK.CaptureDevices.[Device Name].ImageFiles.FileLocation.[Symbolic Name]** = [list of strings]
  - The actual image file location associated with this Symbolic Name
- **FRSDK.CaptureDevices.[Device Name].ImageFiles.CacheFiles** = [True, False]
  - This setting allows to choose whether to cache the files to be imported.
  - Default setting: True
- **FRSDK.CaptureDevices.[Device Name].ImageFiles.Sorting** = [No, Ascending, Descending]
  - This setting allows to choose the sorting order of the files to be imported. Allowed values are No, Ascending and Descending.
  - Default setting: No
- **FRSDK.CaptureDevices.[Device Name].ImageFiles.FrameRate** = [floating point number]
  - This setting allows to choose the rate with which the files would be imported. Zero means import the images at maximum possible rate.
  - Default setting: 0.0

0.8.3.9.2 DirectShowDevice Configuration

DirectShow Capture Devices are only available on MS Windows platforms which have at least version 9 of DirectX. Only imaging devices which provide DirectShow compliant drivers will work properly.

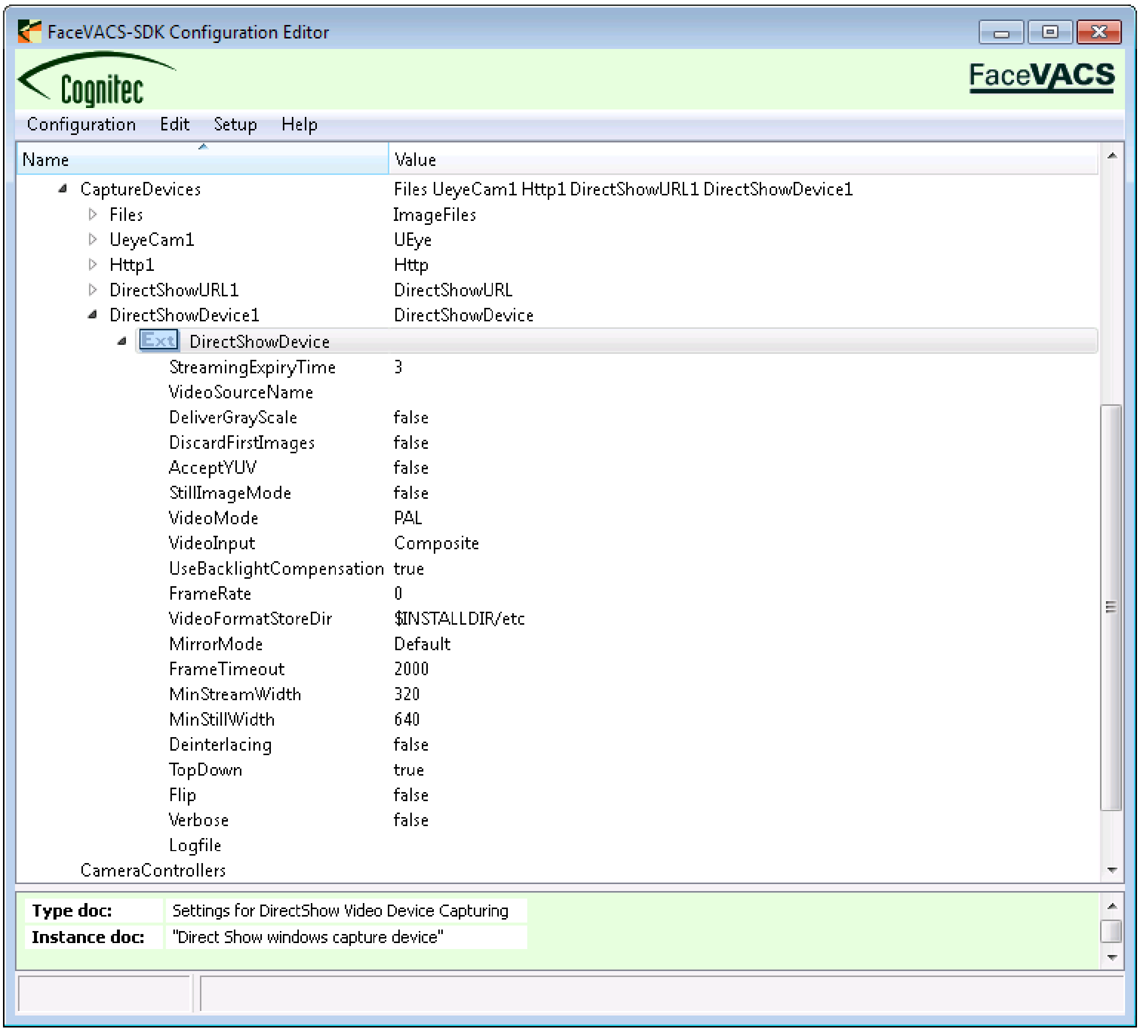


Figure 6: Sample DirectShowDevice configuration subtree

- **FRSDK.CaptureDevices.[Device Name].DirectShowDevice**
  - Even though you can manually choose a video source driver name for the capture device by editing directly the underlying **VideoSourceName** property field, for convenience as well as to avoid mistakes, it is advised to assign this value through a dedicated function of the Configuration Editor. You can invoke this function by right-clicking on the DirectShowDevice node and selecting Run Extension from the context menu. This will open a dialog titled cfgedit which would contain a drop-down list box with all currently available imaging devices to choose from. The dialog also contains buttons allowing to call the device and video format specific configuration dialogs of the currently selected imaging device. These device and video format configuration dialogs are part of the DirectShow driver software provided by the Independent Hardware Vendors (IHVs) of the imaging devices.

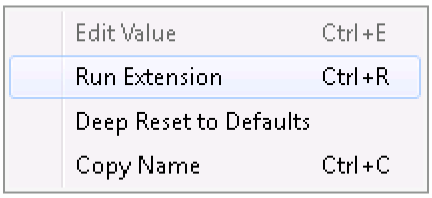


Figure 7: The DirectShow Capture Device context menu

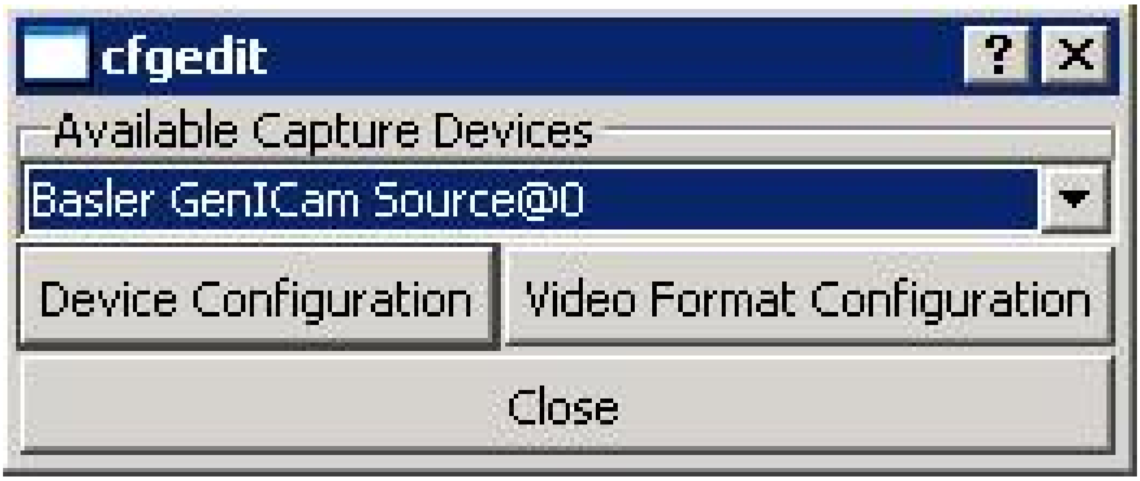


Figure 8: The DirectShowDevice Configuration Dialog

- The **DirectShowDevice Configuration Dialog** usage
  - To assign a physical DirectShow device to the current symbolic capture device invoke the **DirectShow-Device Configuration Dialog** as described above, then select the desired device from the "Available Capture Devices" drop-down list. After closing the dialog the selected there physical DirectShow device name will be transferred to the **VideoSourceName** property of the DirectShow capture device.
  - The button labeled "Device Configuration" in the same dialog invokes the IHV's configuration settings dialog of the currently selected DirectShow device. This dialog gives access to the various vendor dependent configuration parameters of the imaging devices.
  - Correspondingly, the button labeled "Video Format Configuration" invokes the IHV's video format configuration dialog of the currently selected DirectShow device. This dialog allows configuration of the video format provided by the DirectShow layer of the device.
- **Using several instances of the same DirectShow device**
  - If you, for example, have connected several USB cameras of the same type, you might want to use all of them simultaneously. As you can see on the **DirectShowDevice Configuration Dialog** screenshot above, each DirectShow device name is suffixed by "@" character and a number representing the position of the device in the enumeration list. This allows for handling multiple instances of the same type device exactly the same way as if they were multiple instances of different devices. To put it simply, for each physical DirectShow imaging device you have to create a corresponding symbolic capture device and set it's **VideoSourceName** property to be equal to the physical name of the device it represents. Because the physical name of the device is by default extended with the abovementioned suffix it is unique for each computer configuration. After symbolic capture devices are once created, your applications can instantiate and refer to the imaging devices by their symbolic capture device names.

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.StreamingExpiryTime = [positive integer]

- DirectShow video devices can be used only in exclusive mode. As long as a video capture device is opened by an application, it cannot be used by another one. This implies that ideally, in order to minimize contention for this shared resource, device should be closed every time after an image has been captured. However, for continuous video capturing this is quite expensive and will result in low performance. This is exactly where the configuration setting **StreamingExpiryTime** comes into play. By setting it to a some positive number denoting time in miliseconds, after a capture() call, the capture device will be kept open for the specified amount of time and only then it will be released.
- Default setting: 3

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.MirrorMode

- Do not use this feature at the time being.

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.FrameTimeout = [nonnegative integer]

- Some video device drivers (for example, Belkin USB VideoBus II) block frame delivery when no video signal is received, while others return black images instead. To prevent your calling thread from being blocked in a capture() call, you can limit the time the CaptureDevice will wait for the device frame. If this time expires, a black image will be returned by the CaptureDevice, thus simulating the behavior of non-blocking video device drivers.
- Default setting: 0 (no timeout behavior)

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.VideoSourceName

- Here we can be manually enter a name of the physical DirectShow video driver (e.g "Philips ToUcam Pro Camera, Video") associating it this way with this symbolic capture device. Additionally, in order be able to handle multiple imaging devices with the same driver name, you have to append an "@" character and a zero-based index number to the driver common name. There are no gaps in the valid index range (i.g. unused indexes) and the ordering of the physical devices matches this of the DirectShow System Device Enumerator. As we already note above, automatic assigment of this value is recommended over manual editing.

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.StartStopMode

- Obsolete. Changing this setting has no effect.

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.DiscardFirstImages = [True, False]

- Due to a software bug, some USB camera drivers return old images when called after a longer period of inactivity. To circumvent this behavior set the "DiscardFirstImages" to "True", which causes the CaptureDevice to discard the first 2 images if more than 1 second elapsed since the last capture() call.
- Default setting: True

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.AcceptYUV = [True, False]

- Whether to accept YUV video formats from the DirectShow driver.
- Default setting: False

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.VideoMode = [NTSC, PAL]

- This setting does apply to frame-grabber types of devices only, which have analog video inputs. You can use it to choose an appropriate video standard.
- Default setting: PAL

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.VideoInput = [Composite, Tuner, SVideo]

- This setting does apply to frame grabber-like devices having analog video inputs. You can use it to select an appropriate video input type.
- Default setting: Composite

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.UsebacklightCompensation = <True, False>

- This setting does apply to some devices only, e.g. some USB cams. You can use it to enable or disable the backlight compensation function of the camera.
- Default setting: True

FRSDK.CaptureDevices.[Device Name].DirectShowDevice.FrameRate = [nonnegative integer]

- The DirectShow video device frame rate can be configured (up to a limit imposed by the imaging device and/or the system). Use this setting if you want to limit the frame rate of your video device.
- Default setting: 0 (no change of the device configuration)

0.8.3.9.3 DirectShowURL Configuration

DirectShowURL capture device is only available on MS Windows platforms which have at least version 9 of DirectX. It is a pseudo device for playing HTTP and file video streams in video formats supported by the DirectShow infrastructure. Another important prerequisite for the proper operation of the DirectShowURL capture device is the presence of all necessary for the video streams decompression DirectShow decoding filters.

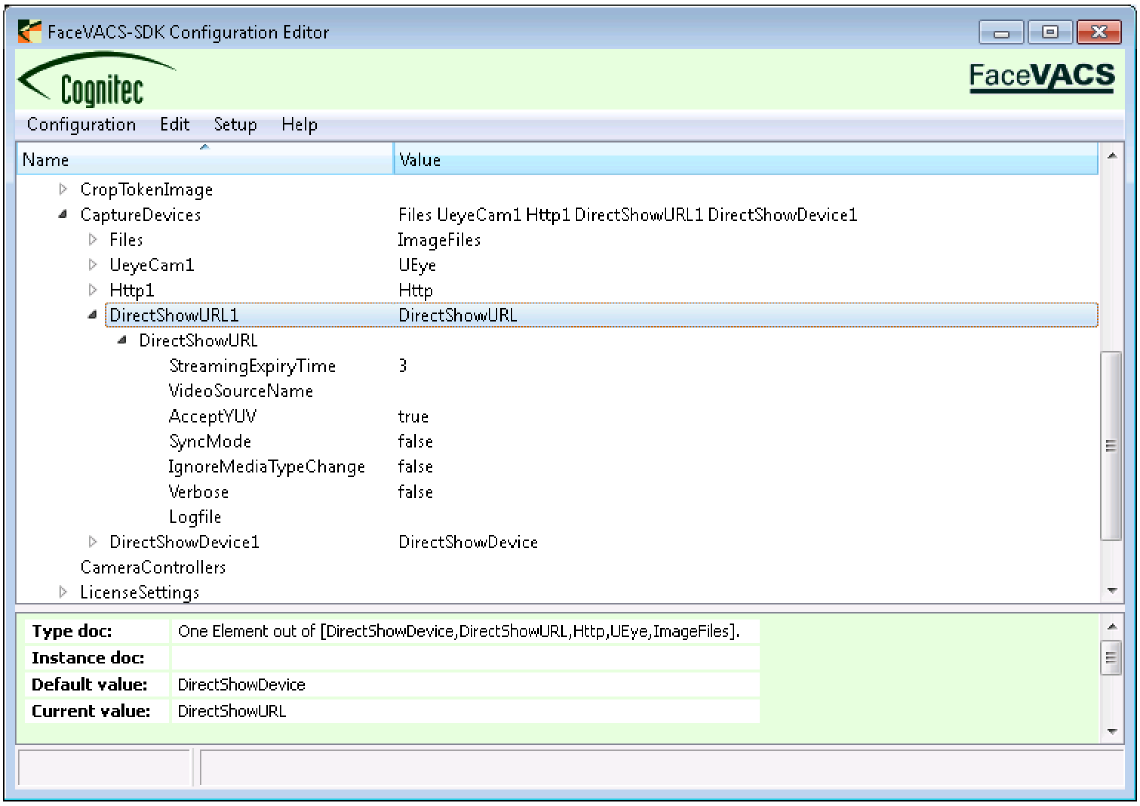


Figure 9: DirectShowURL Capture Device Configuration Subtree

- **FRSDK.CaptureDevices.[Device Name].DirectShowURL.VideoSourceName** = [string]
  - must contain a local file name or remote HTTP URL pointing to a valid video stream. The required codec must be istalled on the machine and supported by DirectShow layer
- **FRSDK.CaptureDevices.[Device Name].DirectShowURL.StreamingExpiryTime** = [positive integer]
  - DirectShow video devices can be used only in exclusive mode. As long as a video capture device is opened by an application, it cannot be used by another one. This implies that ideally, in order to minimize contention for this shared resource, device should be closed every time after an image has been captured. However, for continuous video capturing this is quite expensive and will result in low performance. This is exactly where the configuration setting **StreamingExpiryTime** comes into play. By setting it to a some positive number denoting time in miliseconds, after a capture() call, the capture device will be kept open for the specified amount of time and only then it will be released.
  - Default setting: 3
- **FRSDK.CaptureDevices.[Device Name].DirectShowURL.AcceptYUV** = [True, False]

- Whether to accept YUV video formats from the DirectShow driver.
- Default setting: False

- **FRSDK.CaptureDevices.[Device Name].DirectShowURL.SyncMode** = [True, False]
  - Hold the DirectX filter graph synchronized with image fetching. No frame will be dropped.
  - Default setting: False

0.8.3.9.4 HTTP Capture Device Configuration

The HTTP Capture Devices for the HTTP enabled cameras are available on all supported Windows, Linux and OS X platforms (both 32-bit and 64-bit).

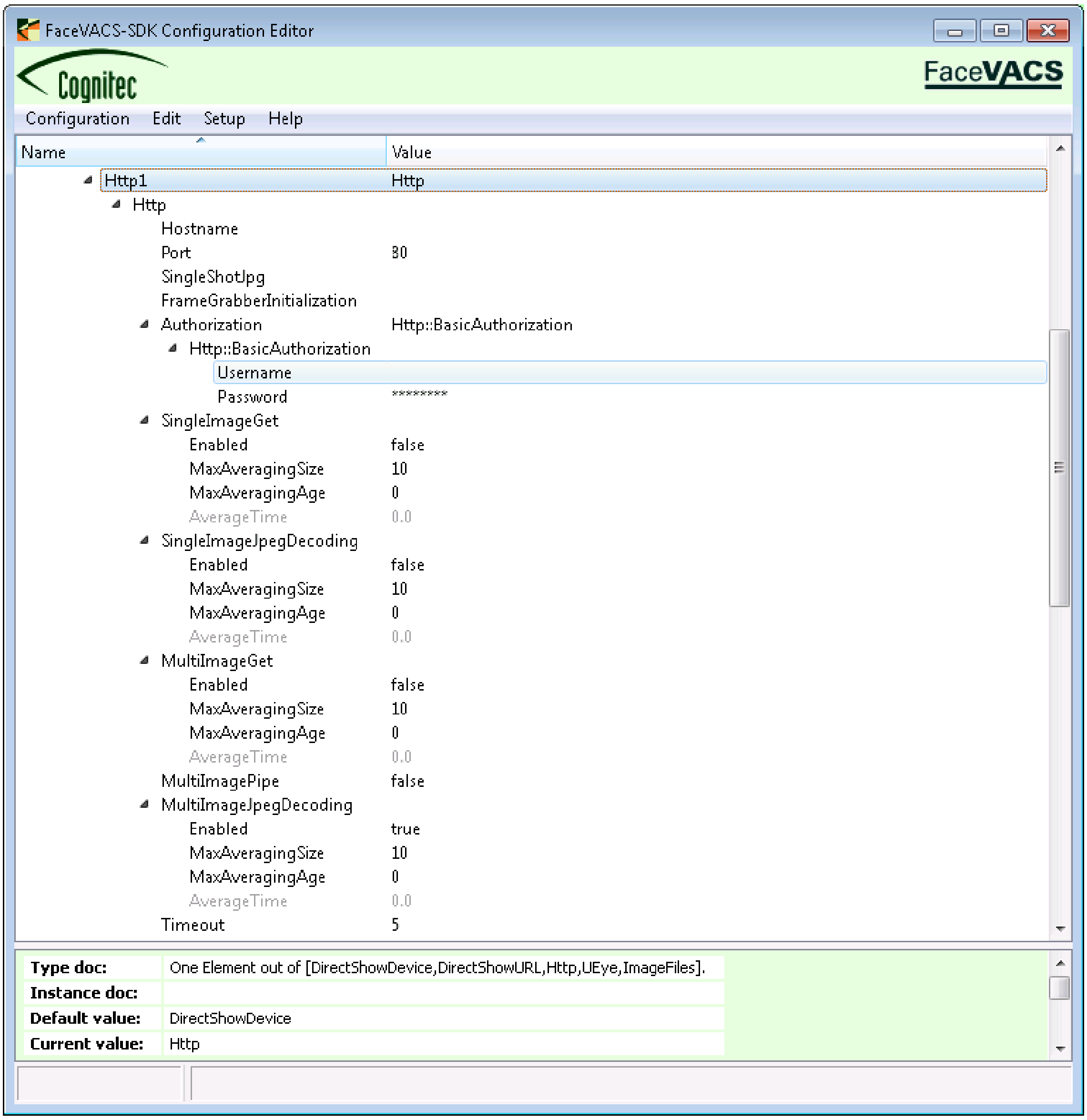


Figure 10: HTTP Capture Device Configuration Subtree

The most important configuration settings of this type capture devices are:

- **FRSDK.CaptureDevices.[DeviceName].Http.Hostname**
  - Hostname or IP address of the HTTP-accessible imaging device

- **FRSDK.CaptureDevices.[DeviceName].Http.Port**
  - Port of the HTTP-accessible imaging device
- **FRSDK.CaptureDevices.[DeviceName].Http.SingleShotJpg**
  - The local portion of the URL for fetching a JPG image, known also as the local address.
- **FRSDK.CaptureDevices.[DeviceName].Http.FrameGrabberInitialization**
  - List of strings which is internally transformed to a list of URLs by prepending the domain (or IP-address) and port portion. Therefore, the strings effectively constitute the local portion of these resulting URLs. During the initialization phase of the HTTP capture device the resulting list of URLs is sent in a form of initializing sequence of HTTP GET requests to the imaging device.
- **FRSDK.CaptureDevices.[DeviceName].Http.Authorization**
  - Sets the desired an authentication mode. Permitted values are **None** or **HTTP::BasicAuthorization**.
- **FRSDK.CaptureDevices.[DeviceName].Http.Username**
  - Sets the username to be used for authorization on the
- **FRSDK.CaptureDevices.[DeviceName].Http.Password**
  - Sets the password of server's account

0.8.3.9.5 uEye Camera Capture Device Configuration

UEye Camera Capture Devices for the cameras of IDS Imaging are available on all supported Windows and Linux platforms (both 32-bit and 64-bit). The biggest advantage of this type of capture device is the adaptive adjustment of the gain and the exposure time of the camera which helps to achieve the highest dynamic range of the facial images. In order to create such capture device a uEye Camera from IDS Imaging and its corresponding software driver are required.

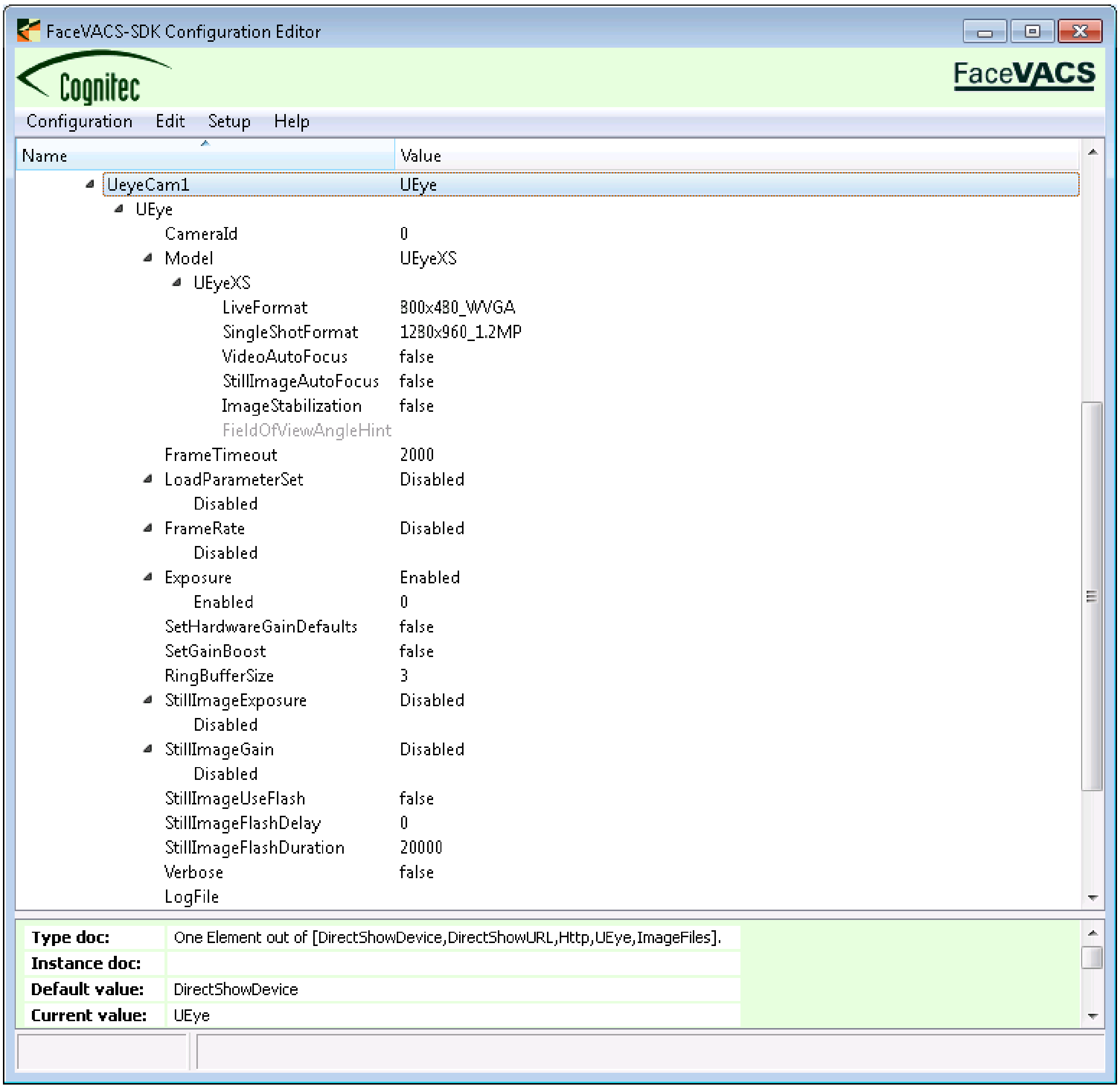


Figure 11: UEye Capture Device Configuration Subtree

For each camera some processing properties can be modified and additionally a Controller can be configured which automatically controls the exposure time and the gain parameters of the camera. The most important configuration settings of this type capture device are:

- **FRSDK.CaptureDevices.[DeviceName].UEye.CameraId** = [positiv integer]
  - Contains a unique index ID for distinguishing among the available IDS UEye cameras. An index value of 0 means the first available camera
  - Default: 0
- **FRSDK.CaptureDevices.[DeviceName].UEye.Model**
  - Allows choosing between Generic or uEye XS specific camera drivers

Here are some more settings which could be usable in particular use cases:

- **FRSDK.CaptureDevices.[DeviceName].UEye.Generic.PixelClock** = [Enabled/Disabled], [positiv integer]
  - If Enabled then Frequency (in MHz) with which the sensor is read out. Influences the maximum framerate and the exposure time. Too high values can result in transmission errors.

  - Default: Disabled (PixelClock will be set to maximum value for the camera.)

- **FRSDK.CaptureDevices.[DeviceName].UEye.FrameRate** = [Enabled/Disabled], [positiv integer]

  - If Enabled then the desired frame rate for the camera. The frame rate is influenced by the current PixelClock.
  - Value of 0 will set the maximal available frame rate
  - Default: Disabled

- **FRSDK.CaptureDevices.[DeviceName].UEye.FrameTimeout** = [positiv integer]

  - Maximum time in msec to wait for a camera frame. If expired, an exception will be thrown. With a value of 0 no timeout is applied. This is dangerous and should be avoided.
  - Default: 2000

- **FRSDK.CaptureDevices.[DeviceName].UEye.Exposure** = [Enabled/Disabled], [positiv integer]

  - If Enabled then Exposure time in milliseconds. The value depends on the current PixelClock and FrameRate.
  - Default: Disabled (the exposure time is set to the maximum possible for the current configuration)

0.8.3.10 Camera Controler Configuration

A controller to adapt dynamically exposure and gain can be parameterized using the subtree **FRSDK.CameraControllers.[DeviceName].[DeviceType]**. In order to activate the camera control function for an capture device with symbolic *DeviceName* this name has to be in the **FRSDK.CameraControllers** list. If it is not there, then the control call will not result in any controling action. Note also does it only make sense to use the controler for devices which allow for setting their gain and exposure values. In order to get a dynamic adaption the controller has to be called in application (see example tracklife.cc). The Controller try to adapt exposure time and gain of the camera that the dynamic range in the face area is optimal for face recognition. The control logic is implemented with a constant, proportional, integral and differential part. The parameters of these part can be configured.

A background area can be defined to adapt to if no face is available (full scene or a virtual face area defined by eye positions). Also available is a variation of the background area when it is active (control in background mode). The following screenshot shows the configurable parameters of such a controller.

Whether exposure or gain or both are controlled is defined by the control path which will be followed during regulation.

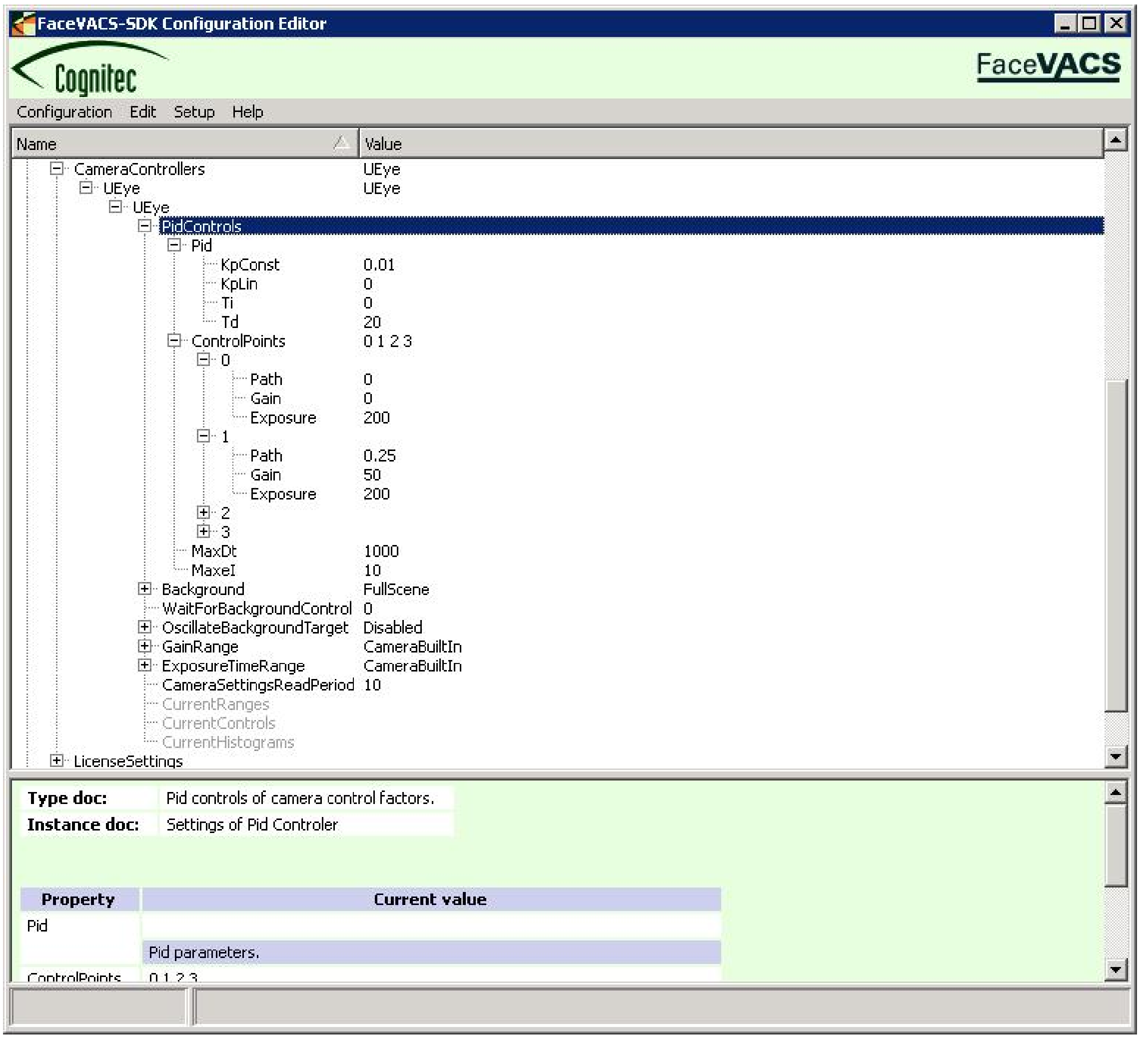


Figure 12: Camera Control Configuration Subtree

- **FRSDK.CameraControllers.[DeviceName]** = [DeviceType]
  - should correspond to the device type or be Generic. This settings differs in their defaults.
- **FRSDK.CameraControllers.[DeviceName].[DeviceType].PidControls.Pid**
  - **.KpConst** = [float]
  - **.KpLin** = [float]
    - Kp = measure ∗ KpLin + KpConst
  - **.Ti** = [float]
    - integral part [ms]
  - **.Td** = [float]
    - differential part [ms]
- **FRSDK.CameraControllers.[DeviceName].[DeviceType].PidControls.ControlPoints**
  - settings defining the control path
- **FRSDK.CameraControllers.[DeviceName].[DeviceType].GainRange** = [CamerBuildIn, UserDefined]
  - Range of gain control

- **FRSDK.CameraControllers.[DeviceName].[DeviceType].ExposureTimeRange** = [CamerBuildIn, UserDefined]
  - Range of exposure time control
- **FRSDK.CameraControllers.[DeviceName].[DeviceType].Background** = [FullScene, UserDefined]
  - FullScene means the complete image to which the controller adapts if no face is available.
  - UserDefined allows to define two eye positions for a virtual face area.
- **FRSDK.CameraControllers.[DeviceName].[DeviceType].OscillateBackgroundTarget** = [enabled, disabled]
  - if enabled a horizontal variation and a speed of variation can be defined.
  - Default: disabled
- **FRSDK.CameraControllers.[DeviceName].[DeviceType].WaitForBackgroundControl** = [positive integer]
  - The time in milliseconds after the controller should measure the background region when no real face was processed
  - Default: 0

0.8.3.11 FaceVACS-SDK Feature enabling/disabling, Limits

FaceVACS-SDK features may be enabled or disabled. This becomes important in the context of licensing and exception handling. A valid license required for using the FaceVACS-SDK usually contains a set of enabled features which are set via a license signature. To get information which features are "protected" via license signature the Configuration object provides the FRsdk::Configuration::protectedItems() member function. It returns a list of key-value pairs some of which represent state of feature enabling according to the list below:

- **FRSDK.LicenseSettings.Finding** = [True, False]
  - If true, automatic face and eyes finding is enabled
- **FRSDK.LicenseSettings.Tracking** = [True, False]
  - If true, automatic face and eyes tracking on video streams is enabled
- **FRSDK.LicenseSettings.Verification** = [True, False]
  - If true, verification is enabled
- **FRSDK.LicenseSettings.Identification** = [True, False]
  - If true, identification is enabled
- **FRSDK.LicenseSettings.Encoding** = [True, False]
  - If true, FIR generation is enabled
- **FRSDK.LicenseSettings.Characteristics** = [True, False]
  - If true, measuring characteristics of portrait images is enabled
- **FRSDK.LicenseSettings.MaxFIRInstances** = [positive integer]
  - Maximum allowed FIR instances per process

Manual modification of these keys leads to a corrupt license information and the complete license is invalid. If a feature was disabled (e.g. by license) and will be requested or accessed (e.g.using a Identification::Processor with disabled Identification feature) a FeatureDisabled exception will be thrown containing a textual description of the disabled feature which was requested. For details see Exception Handling.

If at any time using the FaceVACS-SDK libraries this limit(s) exceeds a LimitExceeded exception will be thrown containing a description of the limit.

0.8.4 Activating the FaceVACS-SDK License

The functionality of the software is controlled by the license settings described above. To run programs linked with the FaceVACS-SDK libraries you have to purchase a software license and to transfer the license activation key to the FaceVACS configuration file(s) used by your programs. The activation key transfer can be accomplished in one of 2 ways:

- By using the license dialog of the Configuration Editor
- By using the 'liccopy' command line utility which is placed in the "bin" subdirectory of your FaceVACS-SDK installation.

With the Configuration Editor, invoke the license dialog with Menu Setup->License and press button "Import Activation Key from File".

With 'liccopy', copy the license activation key file to your PC and transfer the license to your FaceVACS-SDK configuration file with

liccopy -f <FaceVACS-SDK configuration file> -l [license file]

For both files you have to provide the complete path name unless they do not reside in the working directory of your command line invocation. (For the location of the files see section Redistribution.)

Please note, that the license activation transfer removes any comments (lines starting with #) from the configuration file.

Since you might want to have several FaceVACS-SDK configuration files (using different settings for different applications), you will have to transfer the license to each of them.

In case of a missing or corrupt license the FaceVACS-SDK libraries may throw LicenseSignatureMismatch exceptions. For details see Exception Handling. You can check license validity at program startup by doing a trial instantiation of a FRsdk::Configuration object within a try-catch block catching license exceptions.

0.8.5 Redistribution

To provide FaceVACS-SDK applications to customers you will have to redistribute a valid SDK configuration file and some SDK components. Here is the list of what is needed:

- the "etc" SDK subdirectory
- either a (possibly customized) FaceVACS-SDK configuration file or a (possibly compiled in) input stream containing a valid license, for details see FRsdk::Configuration
- the FaceVACS-SDK runtime library resp. libraries and the Intel IPP libraries when you use the IPP versions of the FaceVACS-SDK runtime libraries.
- other runtime libraries (e.g. if using BioAPI interface or .NET runtime library)

  On the target machine you have to setup the shared library search path (see Run-Time Environment for FaceVACS-SDK Applications).

The distributed application directory tree might look like:

```
- <application install root>  ... the root directory of the application
     |                            installation
     +- frsdk.cfg (can reside anywhere, or compiled in)
     |
     +- bin
     |   +-<your application>
     |   +-<runtime FaceVACS-SDK DLL's/shared libraries>
     |   +-<additional runtime libraries> (supporting .NET or BioAPI ...)
     |
     +- etc
         +- portrait
         |    +- age-primary.dat
```

```
|    +- age-secondary-1-10.dat
|    +- age-secondary-1-20.dat
|    +- age-secondary-1-30.dat
|    +- age-secondary-1-40.dat
|    +- age-secondary-1-50.dat
|    +- age-secondary-1-60.dat
|    +- age-secondary-1-70.dat
|    +- age-secondary-1-80.dat
|    +- age-secondary-1-above.dat
|    +- age-secondary-2-10.dat
|    +- age-secondary-2-15.dat
|    +- age-secondary-2-20.dat
|    +- age-secondary-2-25.dat
|    +- age-secondary-2-30.dat
|    +- age-secondary-2-35.dat
|    +- age-secondary-2-40.dat
|    +- age-secondary-2-45.dat
|    +- age-secondary-2-50.dat
|    +- age-secondary-2-55.dat
|    +- age-secondary-2-5.dat
|    +- age-secondary-2-60.dat
|    +- age-secondary-2-65.dat
|    +- age-secondary-2-70.dat
|    +- age-secondary-2-75.dat
|    +- age-secondary-2-80.dat
|    +- age-secondary-2-85.dat
|    +- age-secondary-2-above.dat
|    +- cedisc.dat
|    +- chinest.dat
|    +- cmdisc.dat
|    +- crownest.dat
|    +- ethadisc.dat
|    +- ethbdisc.dat
|    +- ethwdisc.dat
|    +- fpdisc.dat
|    +- gadisc.dat
|    +- gedisc.dat
|    +- gldisc.dat
|    +- learest.dat
|    +- ludisc.dat
|    +- rearest.dat
|    +- sphartr.dat
|
+- cara
|    +- gs1717fc.dat
|    +- gs1717fl.dat
|    +- gs1717fr.dat
|    +- gs2323fc.dat
|    +- is1717da.dat
|    +- is1717dc.dat
|    +- is1717dl.dat
|    +- is1717dr.dat
|    +- is1717fc.dat
|    +- is1717fl.dat
|    +- is1717fr.dat
|
+- ojo
|    +- lddisc.dat
|    +- sfdisc.dat
|    +- eyedisc.dat
|
+- boca
|    +- mouthdisc.dat
|
+- raiz
|    +- bridgedisc.dat
|
+- cmp
     +- a15-0-meansdevimgs.dat
     +- a15-0-regressorX.dat
     +- a15-0-regressorY.dat
     +- b8-0-meansdevimgs.dat
     +- b8-0-regressorX.dat
```

```
+- b8-0-regressorY.dat
+- farmap-a15.dat
+- farmap-b6.dat
+- farmap-b6-sn.dat
+- farmap-b615.dat
+- farmap-b8-0.dat
+- frrmap-a15.dat
+- frrmap-b6.dat
+- frrmap-b6-sn.dat
+- frrmap-b615.dat
+- frrmap-b8-0.dat
+- isotdb-15x.dat
+- isotdb-15z.dat
+- smap-a15.dat
+- smap-b6.dat
+- smap-b6-sn.dat
+- smap-b615.dat
+- smap-b8-0.dat
+- trans-a15.dat
+- trans-b6.dat
+- trans-b8-0.dat
+- trans-15x.dat
+- trans-15z.dat
```

Section Creating shrinked redistributable Packages describes how to shrink a FaceVACS-SDK redistributable package

0.8.6 FaceVACS-SDK and Multithreading

Most of the FaceVACS-SDK interfaces are thread-safe in terms of concurrency. Concurrent function calls from different threads are allowed. Threading behaviour is documented in the class or function documentation section marked with the "MT-safe" flag. No "MT-safe" statement means that it is not safe to call the function from different threads. In some cases the explicitly MT-unsafe flag is set in order to emphasize some points or reasons of the unsafeness.

There are two kinds of processing of multi thread safe function calls:

- *reentrant*, which means the calls from different threads are processed in parallel
- *serialization*, which means the calls from different threads are serialized and processed after each other

The MT-safe flag contains remarkes which of both is available for the thread safe calls.

0.8.7 Exception Handling

Potentially on all operations on library objects (creation, member and static function calls) std::exception objects might be thrown. The what() string may give an hint about the reason of the exception.

Other FRsdk related exceptions which can be thrown at any time are the following (all of them inherits std::exception):

- FRsdk::LicenseSignatureMismatch
  - thrown in case of a corrupt or missing license signature
- FRsdk::FeatureDisabled
  - thrown in case of requesting or accessing a disabled FaceVACS-SDK configurable feature. For features which are possible in that context see FaceVACS-SDK Feature enabling/disabling, Limits
- FRsdk::LimitExceeded
  - thrown in case of exceeding a configured limit (e.g. the maximum number of FIR instances was exceeded during generating a FIR with the Enrollment::Processor)

0.8.8 Other Programming Concepts

The FaceVACS-SDK uses some common programming idioms which will be described in this section.

Reference counting idiom

The principle behind reference counting is to keep a running reference count of an object so that when the reference count false to zero the object is known to be unused and can be deleted. This makes memory management easier for dynamically allocated objects: only keep a count of the number of references to the object and delete the object if the reference count reaches zero. Since simple pointers do not support this behavior we use the class CountedPtr as an level of indirection to manage the count. This technique known as the PROXY pattern is well described in [Gamma, Helm, Johnson & Vlissides, "Design Patterns", Addison-Wesley, ISBN 0-201-63361-2]. The main idea is given as

- **"Provide a surrogate or placeholder for another object to control access to it."**

The FRsdk::CountedPtr class implements the so called DETACHED COUNTED HANDLE/BODY technique. The count of the references and the intelligence for the reference counting is placed outside the counted object into the counted pointer and therefore this technique can be used with classes which do not know of the counted pointer class.

Body/Handle idiom

It is possible to split a single object into two parts - a so called HANDLE which defines the interface of the object and a BODY (often called Implementation) which actually implements the object interface. The only relation between B-ODY and HANDLE is a pointer in the HANDLE which references the BODY. The functionality of the handle interface is obtained by "delegating" to the body. The main advantages of this technique are:

- inexpensive copy of handles by value,
- complete hiding of the body (and, therefore, the implementation)
- sharing bodies between handles.

Many of the main abstractions of the FaceVACS-SDK use the body/handle idiom.

A more detailed explanation of the body/handle idiom can be found at [Coplien, "Advanced C++ Programming Styles and Idioms", Addison-Wesley, ISBN 0-201-54855-0] and [Stroustrup, "The C++ Programming language, - Special Edition", Addison-Wesley, ISBN 0-201-70073-5].

[Previous chapter] [Next chapter]

0.9 User Guide - Biometric Evaluation Tool Suite

[Previous chapter]

0.9.1 Overview and Terms

The biometric evaluation tools provided with the FaceVACS-SDK allow for the efficient computation of identification match lists or score matrices starting from image databases. The tool suite comprises computational tools as well as tool for creation of FIR's from labelled image data.

The PersonID

Starting point of any biometric evaluation is the labelling of the data, i.e. the images in our case. All images which belong to one person have to be associated with an ID designating this person. We will call this ID the **PersonID**

The RecordID

To make statistical analysis resulting in curves like FAR/FRR or CMC several FIR's per person are required. To make all FIR's which belong to the sampe person distinguishable, another ID is required which we call 'RecordI-D'. Since there is a 1 : 1 relation between RecordID's and FIR's, RecordID's have to be unique within one set of evaluation data.

The ImageID

In some use cases, e.g. for storing annotations, it is required to uniquely designate the images, though individuality of single images used to create a FIR is irrelevant to the biometric evaluation.

Structuring Data

All data used within biometric evaluations has to be structured by associating it with PersonID's and RecordID's. While association of data with a PersonID is implied with generating the raw data, it is up to the user how data is associated with RecordID's. While within one test scenario you might want to combine several images to belong to the same RecordID, in another scenario you might associate them different RecordID's to investigate variations in score values. In either case you have to provide a description, how data is organized. For data residing in relational databases, the association of data with PersonID's and RecordID's is given by relations between tables and one can easily construct queries returning data which is appropriately organized. For data available in file format, the relations have to be defined elsewhere. The tools provided by the FaceVACS-SDK utilize an XML format to describe file based data. This format is an extended version of a xml format developed at the USA NIST as part of a "Human Evaluation Framework" for specifying biometric data. It was the format used for the data description in the Face Recognition Vendor Test 2002 (www.frvt.org).

Primary and Secondary biometric data

The raw data containing all biometric information available are images. However, for computing scores between faces in images, most face recognition technologies do not use the original images, but instead some feature sets extracted from these, which reduces memory requirements and allows for faster processing. The extracted feature set, which we call **Facial Identification Record** (FIR) contains all data required by the comparison algorithm in compact form. Moreover, faces of several images can be combined in one FIR by means of clustering techniques. A presupposition of feature extraction out of facial images is the availability of 'annotations', which indicate where there is a 'face' in the image. With the current state of Cognitec software, these annotations are eye positions. They can be detected automatically or explicitely specified in result of a visual inspection by a human operator. It is important to note, that it is the association of the raw images along with the annotation what constitutes the primary biometric data.

Changing or upgrading the algorithms involved can render FIR's unusable, which requires their recreation. In this sense they are 'secondary' biometric data as opposed to the images being the 'primary' biometric data. The process of creating the FIR from one or more annotated images of the same person is called FIR generation. In many application the first step - annotating the images - and the second step - FIR generation - are merged into one procedure, which is called enrollment.

Probe Set and Gallery Set

When making biometric evaluations, one usually makes comparisons of 2 sets: one varying **probe set** of images resp. FIR's representing the test sample and a virtually invariant **gallery set** they are compared with. Of course these terms are interchangeably to a certain extent, especially when creating a score matrix (see below), the difference disappears. Both of probe and gallery set in turn consist of data labelled by PersonID's and RecordID's.

Identification Match List and **Score Matrix**

In many cases one is interested in identifying a probe set of images against a different (large) database. In these cases the objective is to compute for each element of the probe set a list of the most similar elements in the gallery set. Such a list, which contains the RecordID of each gallery set item along with the score and which is ordered by the scores achieved is called an **identification match list**. For practical reasons the length of such a list has to be limited, the limitation criteria which control the lists computation are a threshold for the minimum score to be achieved for a gallery FIR to be included into the list and a maximum number of scores to be contained in this list.

A different task is to evaluate the biometric performance of a technology on a given set of data. For this purpose, one can compute the scores for all pairs of elements of the probe and the gallery set. If there are n elements in the probe and m elements in the gallery sets, the result is a set of n x m comparison values, which is called the **score matrix**. In many cases probe and gallery set are the same for this scenario. However, with larger probe and gallery one will rapidly run against limits of computational power and storage requirements: computing a score matrix from 100.000 elements will require 10 Billion comparision and produce 40 GByte of data, which is still feasible; doing the same with 1 Million elements already produces 4 TByte of data, not to mention computing time.

FAR/FRR and CMC curves

For evaluating biometric systems, dependent on the use case intended either **FAR/FRR curves**, i.e. curves presenting False Acceptance and False Rejection Rate as a function of a pretended threshold (see FAR, FRR and Score for details) or **CMC curves** (Cumulative Match Characteristics) are used. While the former are useful to evaluate performance in access control scenarios, the latter are used to evaluate performance of identification systems. A CMC curve presents the cumulative identification rate as a function of the ranks considered starting with 1, i.e. it tells about the probability to find a match within the first n ranks. For n tending towards infinity, the probability will tend towards 1, so the CMC curve is a monotonically increasing function. The faster a CMC curve approaches 1, the better is the identification system. (See below for more formal definitions of what the **ratesgen** and **cmcgen** tools produce).

It is important to note that in order to compute either False Rejection curves or CMC curves you need more than one image for a representative part of persons in your collection. That's because comparing a single FIR against itself yields a score which is virtually 1, so it will be successfully recognized or identified in every case.

0.9.2 Biometric Evaluation Workflow

Any biometric evaluation with facial data will comprise these steps:

Creating Structured Data Description (For file based data only)

- Input: image directories with defined naming conventions
- Output: XML files describing PersonID/RecordID associations for images

Annotating raw data

- Input: raw data
- Output: stored annotations

FIR generation and storage

- Input: annotated raw data
- Output: stored FIR's

Creating score matrices or identification match lists

- Input: 2 sets of FIR's
- Output: stored score matrices or identification match lists

Analysis

- Input: stored score matrices or identification match lists
- Output: FAR/FRR curves, CMC curves,...

If processing is to be done automatically, the first 2 steps usually will be merged since there is no reason to create processing overhead by dividing the process into 2 stages. The tools provided by the SDK allow for automated annotation and FIR generation during one invocation, allowing for optional storing of annotation data from the first substep into the database.

A question might arise why FIR storage is required. One might design a program which generates FIR's from both the probe and the gallery set, holds them in memory and makes all relevant computations. This approach, however, has several flaws. For the first, test scenarions often involve testing varying probe sets against the same gallery. Recomputing gallery FIR's each time might result in a considerable performance loss. Moreover, since in many cases probe and gallery sets comprise hundred of thousands or even millions of images whose processing is a lengthy procedure persistent storage of FIR's increases stability of the process against failures, allowing for restarting the process where it had been stopped.

Tools provided with the FaceVACS-SDK

```
In the current version of FaceVACS-SDK the SQL based tools are available
for MS Windows platforms only.
```

The tools delivered with the FaceVACS-SDK are with the exception of the annotator command line tools, i.e. they are to be invoked from a command window. They are designed to be used in scripts automating evaluation tasks.

Workflow Step	Tool for SQL Data	Input	Output	Tool for File Data	Input	Output
Creating structured description in XML for image data	-	-	-	**dir2xml**	image directories obeying defined naming conventions	xml file associating images with PersonID's and RecordID's
Annotating raw data	**annotator**	images, optionally with eye positions from SQL database	eye positions in SQL database	-	-	-
Image assessment	**dbassess**	images with id from SQL database	assessment results SQL database / ASCII-file	-	-	-
Annotating raw data/FIR generation and storage	**dbfirgen**	SQL query specifying sets of images, optionally with eye positions `<b>or</b>` `XML file specifying image file names`	FIR's, optionally eye positions in SQL database	**xmlfirgen**	xml file describing sets of images (currently, eye positions cannot be specified)	FIR's stored in a format useable by xml computation tools below
Creating score matrices	**dbcompsim**	SQL query specifying 2 sets of FIRs (probe/gallery)	one binary score values file per probe set element	**xmlcomp-sim**	XML files specifying probe and gallery sets, FIR's stored by **xmlfirgen**	one binary score values file per probe set element
Creating identification match lists	**dbcomp-match**	SQL query specifying 2 sets of FIRs (probe/gallery)	one textual identification match list file per probe set element	**xmlcomp-match**	XML files specifying probe and gallery sets, FIR's stored by **xmlfirgen**	one binary score values file per probe set element

The analysis tools use the output generated from either **dbcompsim/xmlcompsim** or **dbcompmatch/xmlcompmatch** tools:

Workflow Step	Analysis Tool	Input	Output
Analysis: Creating FAR/FRR	**ratesgen**	Set of score values files produced by **dbcomp-match/xmlcompmatch**	List of FAR and FRR data in textual format
Analysis: Creating CMC	**cmcgen**	Set of identification match list files produced by **dbcomp-match/xmlcompmatch**	List of CMC data in textual data

Additional tools to extract results and to post process scores

Workflow Step	Postprocessing Tool	Input	Output

Score Normalization	**scorematproc.exe**	score stored by db/xmlcompsim db/xmlcompmatch	score similar to db/xmlcompsim db/xmlcompmatch output
Score dumping	**writecsvscores.exe**	scores stored by db/xmlcompsim db/xmlcompmatch	CSV file containing scores
Score dumping	**dumpsim.exe**	score files stored by db/xmlcompsim db/xmlcompmatch	to stdout/console

0.9.3 SQL Based Evaluation Tools

SQL Data Types used

For any ID's - RecordID, PersonID, ImageID - SQL string or integer types are admissable, e.g. SQL VARCHAR or LONG.

Image and FIR data have to be represented as a 'blob' type, e.g. "LONGVARBINARY" (also 'IMAGE' or 'BLOB' with some databases)

Eye positions are floating point values, e.g. SQL FLOAT or REAL

Annotator - Annotating raw data

Command line arguments

```
Flag    Item                                                 Remark
-dsn    ODBC data source name or file                        mandatory
-u      ODBC user name                                       optional
-p      ODBC user password                                   optional
-q      SQL query returning ImageID, image,                  mandatory
        optionally eye positions
-s      SQL statement to store eye positions into database   mandatory
```

-dsn command line argument

Specifies a valid ODBC data source (which can be a file data source, too). Depending on the database backend, logon credentials (user name, password) may be required, which may be supplied by the **-u** and **-p** flags.

-q command line argument

Specifies the SQL query for loading images. Each row of the result set has to contain

- an ImageID (SQL string or int type), which is reused in the storing query
- the image data as an SQL "blob" type,
- optionally, the eye positions LeftEyeX, LeftEyeY, RightEyeX, RightEyeY as floating point types in the order specified.

-s command line argument

Specifies the SQL statement to store the eye positions. It has to contain 5 parameters which are assigned to the eye position values LeftEyeX, LeftEyeY, RightEyeX, RightEyeY, and the ImageID, respectively.

The Annotator will display each image from the result set of the input query. If the input query returns eye positions, they are displayed, too. By clicking at appropriate locations new eye positions can be generated or current ones can be moved. Changes are written to the database when clicking the "Apply" button.

Example: (All is on one line in the command line window)

```
annotator -dsn FVDB -q "SELECT ImageId, Img FROM Images WHERE Lx IS NULL" - s "UPDATE Images SET Lx = ?, Ly =
```

This invocation selects for processing all images which still do not have eye positions set. Results are written back to the same table.

DBAssess: Image Assessment

Command line arguments

```
Flag    Item                                                  Remark
-dsn    ODBC data source name or file                         mandatory
-u      ODBC user name                                        optional
-p      ODBC user password                                    optional
-iq     SQL query returning ImageID, image                    mandatory
-oq     SQL statement to store assessment result(replaces builtin default)      optional
-ct if given creates the Assessment result table    optional
-ctq statement for creating result table (replaces builtin default) optional
-of file name for ASCII format result file          optional
```

-dsn command line argument

Specifies a valid ODBC data source (which can be a file data source, too). Depending on the database backend, logon credentials (user name, password) may be required, which may be supplied by the **-u** and **-p** flags.

-q command line argument

Specifies the SQL query for loading images. Each row of the result set has to contain

- first an ImageID (SQL string), which is reused in the storing query
- 2nd the image data as an SQL "blob" type, or a file name as SQL type "string"
- optionally a third columns as same type as 2nd one to used as alternative on null value in the 2nd column

-oq command line argument

Specifies the SQL statement to store the assessment resulte. It has to contain the parameters containing the result values (RecordID first), respectively. If not given a default will be used.

Available fields

Column	Database Type	Explanation
RecordID	VARCHAR(255)	Unique ID of the record/image
IsColor	INTEGER	imag eis encoded as color(1) or not(0)
Width	INTEGER	The width of the portrait image in pixels.
Height	INTEGER	The height of the portrait image in pixels.
Eye0X	REAL	X Coordinate of Right eye center
Eye0Y	REAL	Y Coordinate of Right eye center
Eye1X	REAL	X Coordinate of Left eye center
Eye1Y	REAL	Y Coordinate of Left eye center
EyeDistance	REAL	Distance of the Eye centers in pixel
FaceCenterX	REAL	X Coordinate of face center
FaceCenterY	REAL	Y Coordinate of face center
NumberOfFaces	INTEGER	Number of face found in the image
Glasses	REAL	Returns a measure for the probability of the person in the portrait to wear glasses See ISO standard A.3.2.4.

Eye0Open	REAL	Returns the confidence for the person's left eye being open
Eye1Open	REAL	Returns the confidence for the person's right eye being open
Eye0GazeFrontal	REAL	Value of confidence thal left eye is gazing frontal
Eye1GazeFrontal	REAL	Value of confidence that right eye is gazing frontal
Eye0Red	REAL	redness of left eye
Eye1Red	REAL	redness of right eye
Eye0Tinted	REAL	how tinted is region around left eye
Eye1Tinted	REAL	how tinted is region around right eye
Exposure	REAL	Returns average gray value within facial region
GrayScaleDensity	INTEGER	Gray scale density (number of different gray values) within facial region
NaturalSkinColor	REAL	Value how natural the skin color is
HotSpots	REAL	amount of hot spots in face region
BackgroundUniformity	REAL	how uniform is the background
WidthOfHead	REAL	Horizontal distance between the points where the external ear connects the head in pixels.
LengthOfHead	REAL	Vertical distance between base of the chin and the crown in pixels
PoseAngleRoll	REAL	Returns the tangent of the Pose Angle - Roll
Chin	REAL	distance of pixel from face center to chin
Crown	REAL	distance of pixel from face center to crown
Ear0	REAL	distance inpixel from face center to left ear to head connection
Ear1	REAL	distance inpixel from face center to right ear to head connection
DeviationFromFrontalPose	REAL	Returns a measure for the deviation from frontal pose
DeviationFromUniformLighting	REAL	Returns a measure for the deviation from uniform lighting in the face area (focus on global lighting symmetry)
Sharpness	REAL	Returns a measure for focus and depth of field according to specification of ISO standard 7.3.3
MouthClosed	REAL	Returns the confidence for the person's mouth being closed
ISOOnlyOneFace	INTEGER	Test i fonly one face is visible
ISOGoodVerticalFacePosition	INTEGER	Test the vertical position of the face for ISO compliance
ISOHorizontallyCenteredFace	INTEGER	Test whether the face is centered in the image

ISOWidthOfHead	INTEGER	head width within range of ISO standard
ISOWidthOfHeadBP	INTEGER	head width within range of ISO standard appendix (best practice recommendation)
ISOLengthOfHead	INTEGER	head length within range of ISO standard
ISOLengthOfHeadBP	INTEGER	head length within range of ISO standard appendix (best practice recommendation)
ISOResolution	INTEGER	resolution (ear to ear distance) fites ISO standard
ISOResolutionBP	INTEGER	resolution (ear to ear distance) fites ISO standard appendix (best practice)
ISOImageWidthToHeight	INTEGER	Paragraph A3.2.1 of ISO standard describes a best practice of ratio between image height and width
ISOGoodExposure	INTEGER	True' means there is no over or under exposure
ISOGoodGrayScaleProfile	INTEGER	According to ISO standard 7.4.2.1 and 7.4.2.2 'True' will be returned only if the face area has a intensity resolution of at least 7 bits (128 intensity values)
ISONaturalSkinColor	INTEGER	Skin color of face has natural look according to ISO standard
ISONoHotSpots	INTEGER	There are no hot spots according to ISO standard
ISOIsBackgroundUniformBP	INTEGER	Background is uniform according to best practice recommendations
ISOIsFrontal	INTEGER	The face is frontal according to specifications in ISO standard (without appendix)
ISOIsFrontalBP	INTEGER	The face is frontal according to specifications in ISO standard including best practice recommendations (with appendix)
ISOIsLightingUniform	INTEGER	Returns true if lighting is equally distributed in the face area
ISOEyesOpenBP	INTEGER	Returns true if the both eyes of the person are open
ISOEyesgazeFrontalBP	INTEGER	True if both eyes gazing frontal
ISOEyesNotRedBP	INTEGER	true if eyes have no red reflections
ISONoTintedGlasses	INTEGER	true if no tinted glasses are waered
ISOIsSharp	INTEGER	Returns true if the face area (from chin to crown and from left to right ear) fits the focus and depth in field characteristics (see ISO_19794_5 section 7.3.3)

ISOMouthClosed	INTEGER	Returns true if mouth is closed according to ISO_19794_5 section 7.2.3
ISOIsCompliant	INTEGER	Returns true if the images is compliant with the ISO_19794_5 requirements
ISOBestPractice	INTEGER	Returns true if the images is compliant with the ISO_19794_5 requirements and the Best Practice recommendations
FeatureWearsGlasses	INTEGER	Returns true if the person in the portrait wear glasses
FeatureGender	INTEGER	Gender: male, female
FeatureEthnicity	INTEGER	ethnicity: white, black, asian
FeatureIsChild	INTEGER	true if detected as child
FeatureIsToddler	INTEGER	true if detected as toddler
FeatureIsInfant	INTEGER	true if detected as infant
FeatureIsBelow26	INTEGER	true if deteced as below 26 years
FeatureIsBelow36	INTEGER	if detected as below 36
ProcessingError	VARCHAR(255)	returns a textual description of an processing error

Example: (All is on one line in the command line window)

```
dbassess -f etc/frsdk.cfg -dsn FVDB -u DBUser -p dbpasswd -iq "SELECT RecordID, Img, ImgRef FROM Images" -of a
```

This invocation selects for processing all images. Results are written to the AssessmentResult table.

dbfirgen - Automated annoatation and/or FIR generation

Command line arguments

```
Flag    Item                                             Remark
-f      FaceVACS-SDK configuration and license file      mandatory
-dsn    ODBC data source name or file                    mandatory
-u      ODBC user name                                   optional
-p      ODBC user password                               optional
-outdsn 2nd ODBC data source for result output           optional
-outu   2nd ODBC user name                               optional
-outp   2nd ODBC user password                           optional
-ix     XML file specifying images to be enrolled        }one of these
-iq     SQL query specifying images to be enrolled       }is mandatory
-oq     SQL statement to store FIR's into database       mandatory
-eq     SQL statement to store images and/or eye positions      optional
        into database
-nullfirs       if no FIR could be generated, write a null      optional
        value to the database
```

-f command line argument

Specifies a valid FaceVACS-SDK configuration file.

The **-dsn**, **-u**, **-p** arguments are to be used as with the annotator.

The images to be enrolled have to be specified with either the **-iq** or the **-ix** command line argument. Only one of these is admissable.

-ix command line argument

The xml file given by the **-ix** command line argument has to specify image file names and their association with RecordID's and PersonID's in the format defined for the SDK tools (see below). Such files can be created with **dir2xml**.

-iq command line argument

Specifyies the SQL query returning images and additional data to be processed. See table below for admissable number of columns in the result set and the interpretation of each column dependent on number of columns:

#Columns in Result Set	Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8
2	RecordI-D	Image	-	-	-	-	-	-
3	PersonI-D	RecordI-D	Image	-	-	-	-	-
3	RecordI-D	Image	ImageID	-	-	-	-	-
4	PersonI-D	RecordI-D	Image	ImageID	-	-	-	-
6	RecordI-D	Image	Eyes-LeftX	Eyes-LeftY	Eyes-RightX	Eyes-RightY	-	-
7	PersonI-D	RecordI-D	Image	Eyes-LeftX	Eyes-LeftY	Eyes-RightX	Eyes-RightY	-
7	RecordI-D	Image	ImageID	Eyes-LeftX	Eyes-LeftY	Eyes-RightX	Eyes-RightY	-
8	PersonI-D	RecordI-D	Image	ImageID	Eyes-LeftX	Eyes-LeftY	Eyes-RightX	Eyes-RightY

Remarks The values returned in the RecordID and PersonID columns are filled in the respective parameter values of subsequent SQL storing statements (see below).

Any **consecutive** rows of the result set having the **same values of RecordID and PersonID** will be aggregated and used to create a **single FIR**.

If no PersonID is returned by the statement, it is implicitely assumed to be equal to the RecordID.

If eye positions are available for **all** images which belong to the same RecordID, FIR's are generated from the annotated images. In all other cases automatic annotation (detection of eye positions) is run prior to FIR generation.

-oq command line argument

With the **-oq** command line argument the SQL statement to store the FIR is specified. This statement can contain 2 up to 4 parameters:

#Parameters	Parameter1	Parameter2	Parameter3	Parameter4
2	FIR	RecordID	-	
3	FIR	RecordID	PersonID	
4	FIR	Status	RecordID	PersonID

Remarks: If 4 parameters are used, for the 2nd one, 'Status', a string describing the enrollment result will be provided. This is useful with the **-nullfirs flag** only, since it then allows for some diagnostics of enrollment failure reasons. The Status column can contain the values enumerated below:

Status String	Explanation
OK	Successful enrollment
NOIMG	There was no image available for the currentRecordID
FFEF	Face or eye finder failure
EMAP	Face too close to image border
ERROR	A software exception occured during processing. This is mostly due to image format corruption.

For successful enrollments, the result status will be 'OK'.

Note that the order of values is reverse to that in the query statement. This is required to allow for UPDATE statements, where the RecordID/PersonID would be used within a WHERE clause, which implies that the corresponding SQL parameter has to be the last one.

The parameters for the RecordID and PersonID are filled with the corresponding values of the image query. If a parameter is provided for PersonID, but no PersonID value had been returned in the image query, the value of the RecordID will be used.

-eq command line argument

Specifies the SQL statement to be used to store the image and/or eye positions. Again, there is much variability with the number of parameters:

#Parameters	Param1	Param2	Param3	Param4	Param5	Param6	Param7	Param8	Remark
2	**Image**	**ImageID**	**-**	**-**	**-**	**-**	**-**	**-**	To store images
3	**Image**	**ImageID**	**RecordID**	**-**	**-**	**-**	**-**	**-**	To store images
4	**Image**	**ImageID**	**RecordID**	**PersonID**	**-**	**-**	**-**	**-**	To store images
5	**LeftEyeX**	**LeftEyeY**	**RightEyeX**	**RightEyeY**	**ImageID**	**-**	**-**	**-**	To store annotations when images are already in database
6	**LeftEyeX**	**LeftEyeY**	**RightEyeX**	**RightEyeY**	**ImageID**	**RecordID**	**-**	**-**	To store annotations when images are already in database
7	**Image**	**LeftEyeX**	**LeftEyeY**	**RightEyeX**	**RightEyeY**	**ImageID**	**RecordID**	**-**	To store images along with annotations
8	**Image**	**LeftEyeX**	**LeftEyeY**	**RightEyeX**	**RightEyeY**	**ImageID**	**RecordID**	**PersonID**	To store images along with annotations

Remark: The manifold of parameters involved may be confusing, but it is the price to be paid for sake of flexibility.

-nullfirs flag This flag controls the behaviour of the tool in case no FIR could have been generated. In this case, if **-nullfirs** is given a record containing an SQL null value instead of a FIR will be written to the database. Otherwise processing is continued with the next record. Inserting null values can be helpful to detect which RecordID's could not yet be successfully processed.

dbcompsim - computing a score matrix

dbcompsim takes 2 sets - probe and gallery - of FIR's labelled by RecordID's and, optionally, PersonID's and compares each element of the probe set with each element of the gallery set, thus producing the score matrix. There will be one binary score file written for each probe element set, which contains as many score values as there are elements in the gallery set. The order of the score values corresponds to the order of the items in the population, i.e. the order how they are returned by the gallery query. The name of the score file is the probe set element's RecordID.

dbcompsim command line flags:

```
Flag    Item                                                  Remark
-f      FaceVACS-SDK configuration and license file           mandatory
-dsn    ODBC data source name or file                         mandatory
-u      ODBC user name                                        optional
-p      ODBC user password                                    optional
-pq     SQL query specifying probe set                        mandatory
-gq     SQL query specifying gallery set                      mandatory
-odir   output directory for storing result files             mandatory
-ignore ignore any errors upon building the gallery set or    optional
        processing the probe set ID's
```

-f, -dsn, -u, -p command line arguments These arguments are used in the same way as with **dbfirgen**

-pq/-gq command line arguments

Specify the SQL queries defining the probe and the gallery set for the score matrix computation, respectively. The result sets of both queries have to contain either 2 or 3 columns. If only a RecordID is available (as it is the case if there is only one image per person available), the result set columns are:

RecordID, FIR.

If PersonID's are available, the result set columns are:

PersonID, RecordID, FIR

The PersonID and RecordID columns in the result set have to be of either string (e.g. SQL VARCHAR) or integer type. The FIR has to be of a BLOB type, e.g. SQL_Longvarbinary (aka 'Image' with some RDBMS). There are no conventions imposed on column names. There can be identical queries for probe and gallery sets.

-odir command line argument

Specifies the output directory for score matrix computation. The result files described above will be written into this directory. The directory must be writeable at the time of the invocation. Names of the output files are derived from the RecordID. Existing result files may be overwritten without confirmation request. However, the tool recognizes when parts of the score matrix already have been computed and skips them upon reinvocation.

-ignore flag

Especially when processing huge data sets, which usually will take too much time to be supervised, it is important to be robust against unforeseeable error conditions. If the **-ignore** flag is provided, processing will not be stopped when the tool encounters any noncritical error condition. Otherwise, the program will write an error message and stop processing.

A **noncritical** error condition in this sense is for example the presence of a null value in the result set instead of a valid FIR.

When the error occurs when building the population from the gallery set, an 'empty FIR' will be inserted into the population. Empty FIR's are special FIR's which are defined by the property, that they compare to 0 with any other FIR. Hence, in the resulting score matrice one would find a '0' on these positions.

When the error occurs upon comparing a probe set element against the gallery, no score file will be created for this element.

dbcompmatch - computing identification match lists

dbcompmatch takes 2 sets - probe and gallery - of FIR's labelled by RecordID's and compares each element of the probe set with the gallery. The best matches of this comparison are written into a textual identification match list file, which contains the RecordID's of the gallery set elements matched and the score values. Only matches whose score is above a given threshold are returned, and the number of matches included into the match list is limited by a maximum size parameter.

There will be one match list file written for each probe element set, which can contain any number of matches starting from zero up to the aforementioned maximum size. The name of the match list file is that of the RecordID of the probe set element plus an '.lst' extension.

dbcompmatch command line flags:

```
Flag    Item                                                  Remark
-f      FaceVACS-SDK configuration and license file           mandatory
-dsn    ODBC data source name or file                         mandatory
```

```
-u      ODBC user name                                       optional
-p      ODBC user password                                   optional
-pq     SQL query specifying probe set                       mandatory
-gq     SQL query specifying gallery set                     mandatory
-thr    floating point value for match list threshold        mandatory
-max    integer specifying maximum number of matches         mandatory
-odir   output directory for storing result files            mandatory
-ignore ignore any errors upon building the gallery set or   optional
        processing the probe set ID's
```

-f, -dsn, -u, -p, -pq, -gq, -ignore command line arguments These arguments are used in the same way as with **dbcompsim**

-thr/-max command line arguments The values specified with these arguments control the size of the identification match list computed. With **-thr** a threshold value is specified. Only matches whose score is bigger than that threshold are included in the match list.

With the **-max** parameter the overall size of the match list is limited. The match list will not contain more items than specified here.

Note that in any case the probe set element is compared with **all** elements of the gallery set. Cutting off the matchlist will occur afterwards.

0.9.4 File Based Evaluation Tools

dir2xml - Creating xml descriptions of image data

The file based evaluation tools as well as **dbfirgen** when started with the **-ix** option use xml descriptions of image data to define image sets to be processed. Since the xml format is described in detail (see XML file format) you can easily compile format file for any image data. To ease this task for you, the **dir2xml** tool is provided which automatically generates xml description files from image data obeying certain naming conventions.

Remember that the purpose of the xml description is to associate images with PersonID's, RecordID's and ImageID's. So these have to be coded with the image file name or location. The **dir2xml** tool supports 2 naming schemes:

- a naming scheme based solely on the image file base name
- a naming scheme based on image file base name and parent directory name

The 'basename' of a file is the filename without directory specification and without everything starting with the first dot in the filename.

Example:

```
Filename: D:\Images1\p127-r20021012-it1.jpg
Basename: r20021012-it1
```

File base name based naming scheme

With the first naming scheme, file base names are supposed to be composed in this way:

```
<PersonID><DELIM>RecordID<DELIM><anyString>

<DELIM>
A single character used as a delimiter between different name
components.

<PersonID>
any string designating the PersonID. Obviously, this string cannot contain
the delimiter defined above.

<RecordID>
any string designating the RecordID and not containing <DELIM>.
```

Example image set using hyphen delimiter

```
Using <DELIM> = '-':

Filename: D:\Images1\P11-r20021012-it1.jpg
Filename: D:\Images1\P11-r20021012-it2.jpg
Filename: D:\Images1\P12-r20021112-it21.jpg
Filename: D:\Images1\P15-r20020812-it2.jpg
Filename: D:\Images1\P15-r20020912-st2.jpg
Filename: D:\Images1\P15-r20020912-st6.jpg
```

This example images set contains

2 images with PersonID P11 and RecordID r20021012

1 image with PersonID P12 and RecordID r20021112

1 image with PersonID P15 and RecordID r20020812

2 images with PersonID P15 and RecordID r20020912

File base name/directory based naming scheme

With the second naming scheme, image files are supposed to be located in subdirectories with their PersonID's, while their base names are composed in this way:

```
Directory       File Base Name

<PersondID>---|- RecordID<DELIM><anyString>
              |- RecordID<DELIM><anyString>
              |- RecordID<DELIM><anyString>
<PersondID>---|- RecordID<DELIM><anyString>
              |- RecordID<DELIM><anyString>
```

Restructuring the image set from the example above in this way yields:

```
Using directory/base name coding, <DELIM> = '-':

Filename: D:\Images2\P11\r20021012-it1.jpg
Filename: D:\Images2\P11\r20021012-it2.jpg
Filename: D:\Images2\P12\r20021112-it21.jpg
Filename: D:\Images2\P15\r20020812-it2.jpg
Filename: D:\Images2\P15\r20020912-st2.jpg
Filename: D:\Images2\P16\r20020912-st6.jpg
```

Variants

When parsing the image name from left, all up to the first delimiter or the dot will be matched to form the PersonID. If there is only one delimiter or no delimiter at all, the RecordID will be set to equal the PersonID.

The example below shows, how an image name containing 2, 1 and no delimiter(s), respectively will be parsed to form the PersonID and the RecordID:

```
<DELIM> = '-':
P11-r20021012-it1: PersonID = P11, RecordID = r20021012
P11_r20021012-it1: PersonID = RecordID = P11_r20021012
P11_r20021012_it1: PersonID = RecordID = P11_r20021012_it1
```

After these preliminary remarks now the invocation of **dir2xml** will be described:

Command line arguments

```
Flag    Item                                            Remark
-idir   directory with image data                       mandatory
-ox     XML file specifying images to be enrolled       mandatory
-delim  delimiter to be used for ID derivation          optional
-xsd    path to XML schema description files            optional
-pidfromdir     use parent directory of image files as PersonID optional
```

-idir command line argument

Specifies the directory where the image data resides. All images have to reside below this directory. The program looks for names of plain files, parsing their names according to the rules explained above and adding them to the xml file. However, no check of any kind is made of the plain file types. You have to ensure, that any plain files below the starting directory indeed are image files. PersonID and RecordID derivation is done according to the rules explained above. Which of the 2 naming schemes is applied depends on whether the **-pidfromdir** flag below is provided.

-ox command line argument

Specifies the name of the xml file where the image description is to be written to. If the file does not exist, it will be created, otherwise it will be overwritten.

-delim command line argument

Specifies the delimiter to be used for ID derivation. If none is given, the whole file base name will be used.

-pidfromdir flag

Specifies that the base name/directory naming scheme is to be applied instead of the 1st one. In this case all subdirectories of the starting directory are parsed for plain files. In case some are found in a given subdir [dirname], their names are added to the xml file, whereby the PersonID is set to [dirname] and the RecordID is derived from the file base name using the delimiter defined. If no delimiter is given, both PersonID and RecordID are set to 'dirname'.

-xsd command line argument

Specifies the directory with the files containing the xml schema descriptions the xml format is based upon. These are provided along with the SDK and installed in the etc/xml subdirectory of your installation. The directory will be included into the xml file and used by the evaluation tools when parsing the xml file. Enter the full path name to the 'etc/xml' subdirectory of the FaceVACS-SDK installation. Default is "etc/xml".

Examples:

(The SDK is assumed to be installed in C:\FVSDK_8_5_0)

To make an xml description of the image set of the 1st naming scheme example

```
dir2xml -idir D:\Images1 -ox C:\temp\images1.xml -xsd C:\FVSDK_8_5_0\etc\xml -delim -
```

To make an xml description of the image set of the 2nd naming scheme example

```
dir2xml -idir D:\Images2 -ox C:\temp\images2.xml -podfromdir -xsd C:\FVSDK_8_5_0\etc\xml -delim -
```

xmlfirgen - Automated annoatation and/or FIR generation

Command line arguments

```
Flag    Item                                                 Remark
-f      FaceVACS-SDK configuration and license file          mandatory
-ix     XML file specifying images to be enrolled            mandatory
-odir   output directory for storing FIR's created           mandatory
-probe  create a FIR for every single image specified        }one of these
-gallery create a FIR per set of images with the same RecordID  }is mandatory
```

-f command line argument

Specifies a valid FaceVACS-SDK configuration file.

-ix command line argument

Specifies image file names and their association with RecordID's and PersonID's in the format defined for the SDK tools.

-odir command line argument

Specifies directory where the FIR's created can be stored. Disc space available within this directory has to be about 2 kB ∗ (Number of Images). The directory must exist before starting **xmlfirgen**.

-probe, -gallery

Defines whether FIR's are to be created for every single image (**-probe**) or on a per record base (**-gallery**). Typically, FIR's generated with the **-probe** option will be specified as probe sets for the evaluation tools **xmlcompsim**/**xmlcompmatch**.

With the **-gallery** option, only one FIR is generated for all images within the image set associated with the RecordID. Obviously, within this process (which internally is implemented by means of a clustering algorithm) some information will be lost. To circumvent this information loss and to increase recognition performance, for probe sets no clustering is applied. This reflects the behaviour of real-world applications, where upon enrollment usually one FIR is created out of several enrollment images, whereas in verification and identification attempts one FIR is created for every sample image.

Remark

In addition to the PersonID/RecordID/Image association the XML specification can include eye annotations, too (see see XML file format) below). The <xmlfirgen> will use them to bypass automatic annotation provided that **all** images which belong to the same RecordID have annotations.

xmlcompsim - computing a score matrix

xmlcompsim takes 2 sets - probe and gallery - of FIR's labelled by RecordID's and compares each element of the probe set with each element of the gallery set, thus producing the score matrix. There will be one binary score file written for each probe element set, which contains as many score values as there are elements in the gallery set. The name of the score file is the probe set element's RecordID. The order of the score values in this file corresponds to the order of the items in the population, i.e. the order how they were specified in the XML file used when creating the gallery FIR's with **xmlfirgen**.

xmlcompsim command line flags:

```
Flag    Item                                                Remark
-f      FaceVACS-SDK configuration and license file         mandatory
-px     XML file specifying probe set                       mandatory
-pdir   directory with probe FIR's                          mandatory
-gdir   directory with gallery FIR's                        mandatory
-odir   output directory for storing result files           mandatory
-debug  output extended progress and diagnostic information optional
```

-f command line argument

Specifies a valid FaceVACS-SDK configuration file.

-px command line argument

Specifies the XML file describing the probe image set. This can be a subset of the image set used to create the probe FIR's with **xmlfirgen**.

-pdir command line argument

Specifies the directory where the probe FIR's (generated previously with **xmlfirgen -probe ...**) are stored.

-gdir command line argument

Specifies the directory where the gallery FIR's (generated previously with **xmlfirgen -gallery ...**) are stored.

-odir command line argument

Specifies the output directory for score matrix computation. The result files described above will be written into this directory. The directory must exist before starting **xmlcompsim**. Names of the output files are derived from the RecordID. Existing result files may be overwritten without confirmation request. However, the tool recognizes when parts of the score matrix already have been computed and skips them upon reinvocation.

xmlcompmatch - computing identification match lists

xmlcompmatch takes 2 sets - probe and gallery - of FIR's labelled by RecordID's and compares each element of the probe set with each element of the gallery set. The best matches of this comparison are written into a textual identification match list file, which contains the RecordID of the gallery set element matched and the score value. Only matches whose score is above a given threshold are returned, and the number of matches included into the match list is limited by a maximum size parameter.

There will be one match list file written for each probe element set, which can contain any number of matches starting from zero up to the aforementioned maximum size. The name of the match list file is that of the RecordID of the probe set element plus an '.lst' extension.

xmlcompmatch command line flags:

```
Flag    Item                                               Remark
-f      FaceVACS-SDK configuration and license file        mandatory
-px     probe set xml ID/image file                        mandatory
-pdir   directory with probe FIR's                         mandatory
-gdir   directory with gallery FIR's                       mandatory
-thr    floating point value for match list threshold      mandatory
-max    integer specifying maximum number of matches       mandatory
-odir   output directory for storing result files          mandatory
-debug  output extended progress and diagnostic information     optional
```

-f command line argument

Specifies a valid FaceVACS-SDK configuration file.

-px command line argument

Specifies the XML file describing the probe image set. This can be a subset of the image set used to create the probe FIR's with **xmlfirgen**.

-pdir command line argument

Specifies the directory where the probe FIR's (generated previously with **xmlfirgen -probe ...**) are stored.

-gdir command line argument

Specifies the directory where the gallery FIR's (generated previously with **xmlfirgen -gallery ...**) are stored.

-odir command line argument

Specifies the output directory to store identification match list files. The directory must be writeable at the time of the invocation. Names of the output files are set to the RecordID of the probe set element plus an '.lst' extension. Existing result files may be overwritten without confirmation request. However, the tool recognizes when output files already have been computed and skips them upon reinvocation.

-thr/-max command line arguments The values specified with these arguments control the size of the identification match list computed. With **-thr** a threshold value is specified. Only matches whose score is bigger than that threshold are included in the match list.

With the **-max** parameter the overall size of the match list is limited. The match list will not contain more items than specified here.

Note that in any case the probe set element is compared with **all** elements of the gallery set. Cutting off the matchlist will occur afterwards.

0.9.5 Convert binary score matrix files into CSV representation

For performance and output file size reduction reasons, tools **xmlcompsim** and **dbcompsim** produce their output in binary representation. However, a representation of the output matrix in a CSV (comma separated values) file is sometimes required. This can be produced with **writecsvscores**.

writecsvscores command line flags:

```
Flag    Item                 Remark
-idir   input directory      mandatory
-ofile  CSV file name        mandatory
-delim  delimiter            optional
```

-idir input directory

Specifies the directory containing a binary score matrix as produced by **xmlcompsim** or **dbcompsim**.

-ofile CSV file name

Specifies the name of the file to write the CSV representation of the score matrix to.

-delim delimiter

By default, data items are separated by a comma (","). With this option another delimiter can be provided. Note that the delimiter to be used must not be part of any PersonID or RecordID.

0.9.6 Statistic Tools (CMC, FAR / FRR)

The following section describes how to use the CMC and FAR/FRR tools

We assume we are given two sets:

```
G - gallery, a set of FIR's
P - probes, also a set of FIR's
```

Each (non-void) FIR f in (G | P) is derived from one or more images of some person (or subject) s(f). Let S be the set of all subjects belonging to the FIR's in (G | P), i.e. S = s(G | P).

A single comparison of two FIR's f1 and f2 yields a score value score(f1,f2). score is symmetric, i.e. score(f2,f1) = score(f1,f2).

We have the following sets of score values:

```
Same = { score(p,g) | p in P, g in G, s(p) == s(g) }

Diff = { score(p,g) | p in P, g in G, s(p) != s(g) }
```

Thus, the set Same corresponds to comparisons of FIR's of the same person, and the set Diff to comparisons of FIR's of different persons.

Now we can define FRR and FAR at any threshold t in R (the real numbers):

```
FRR(t) = | { c in Same | c < t  } | / | Same |

FAR(t) = | { c in Diff | c >= t } | / | Diff |
```

(Here, | ... | means set cardinality)

Since score(f1,f2) always lies in the interval [0,1], it is sufficient to consider thresholds in that interval. In particular, we have: FRR(0) = 0 FRR(1) = 1 (actually, this is not true if there is an FIR that occurs both in P and G, since score(f,f) = 1; however, comparing an FIR with itself isn't of practical relevance anyway) FAR(0) = 1 FAR(1) = 0

Crossover point: If there is t_0 $>$ 0 with FAR == FRR

ROC: { (FAR(t), FRR(t)) | t in [0,1] }

CMC:

Define for each probe FIR p in P:

```
Same_p = { score(p,g) | g in G, s(p) = s(g) }

Diff_p = { score(p,g) | g in G, s(p) != s(g) }

rank_p = | { c in Diff_p | c >= max Same_p } | + 1
```

rank_p lies between 1 and |G|. rank_p = 1 iff the highest score is obtained for a comparison with an FIR of the same person. If Same_p is empty, rank_p isn't well-defined, so such p should be ignored (resulting in a smaller set P' of usable probe FIR's) and not contribute to the CMC.

Now we can define the CMC function:

```
CMC: { n | n in 1, ..., |G| } -> [0,1]

CMC(n) = | { p in P' | rank_p <= n } | / | P' |
```

We also can define a truncated CMC function, that only considers ranks up to some k:

```
CMC_k: { n | n in 1, ...,k } -> [0,1]
CMC_k(n) = CMC(n)
```

The statistic tools **cmcgen** and **ratesgen** supports the following command line options:

```
Flag    Item                                                  Remark
-idir   input directory for reading score/match list files   mandatory
-sm     the input dir contains score files                   optional
-ml     the input dir contains match list files                   optional
-ofile  the output file which contains the plot curve bins        mandantory
```

One of the options **-sm** and **-ml** must be provided. The output files generated by the tools the curve bins seperated by spaces. The first entry in the line is the represents the x-axis value, the other ones the corresponding y-axis value (each for every curve).

0.9.7 XML file format

The xml files useable with the files use an schema which defines an entity named "signature-set". Signature sets are collection of **signatures**, where each signature represents a set of related biometric data.

Each signature has a "subject_id" and a "name" attribute, which correspond to the PersonID and the RecordID defined above, respectively.

Any signature contains of one or more **datasets**, which comprise the actual biometric data described. For facial data, datasets actually are sets of image files.

Each file element has a mandatory **name** attribute, which is set to the image file path name. Both absolute and relative path names are allowed, if relative names are used, they must be relative to the working directory of the tool using the XML file. The specification allows for providing with the file element optional eye annotation attributes **LeftEyeX**, **LeftEyeY**, **RightEyeX**, **RightEyeY**, which are floating point types. However, current versions of the XML based evaluation tools do not make use of any eye annotations.

Invariant Header Part

```
<?xml version="1.0" encoding="UTF-8"?>
<signature-set xmlns="http://www.nist.gov/humanid/hef/xml/0.99.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.nist.gov/humanid/hef/xml/0.99.0
```

This header part is the same in all files generated. It provides essential XML declarations for the xml type 'signature set'.

Variable Header Part providing xsd schema file to be used

Example:

```
C:\FVSDK_8_5_0\etc\xml\annotated-multifile-video-set.xsd">
or
/etc/facevacs/xml/annotated-multifile-video-set.xsd">
```

Note that both Win32 and UNIX-like notations are admissable. The latter can be used on Win32 platforms, too.

The variable header part defines the xml schema file to be used. This file (which references further files) is provided with the SDK and installed within the etc/xml subdirectory. The directory part of the schema file path name (which has been set to **C:\FVSDK_8_5_0\etc\xml** in this example) has to be set according to the path of the etc/xml subdirectory of the SDK installation or whereever you put the xml schema files. If relative names are provided, they have to be relative to the working directory of any tool referencing the XML file.

Signature parts

Following to the header part, there are one or more signature definitions, which conform to the following format:

```
 <signature name="r20021012" subject_id="P11">
```

```
  <sigmember modality="annotated multifile face video" metadata="annotated-multifile-face-video-type">
   <dataset media="digital still" datatype="jpeg">
     <file name="D:\Images2\P11\r20021012-it1.jpg"/>
     <file name="D:\Images2\P11\r20021012-it2.jpg"/>
     <file name="D:\Images2\P11\r20021012-it3.jpg" LeftEyeX="58.7" LeftEyeY="144.5" RightEyeX="106.90" RightEy
   </dataset>
  </sigmember>
 </signature>
```

This xml sequence defines a signature for PersonID P11 and RecordID r20021012, which contains one dataset comprising 3 images. The 3rd image element of the dataset contains eye annotations.

There can be an arbitrary number of signatures within a signature set. Several signatures can have the same subject_id (= PersonID), but signature names (= RecordID's) have to be unique within a signature set.

Closing the signature set (and the XML file

Finally the signature set and the XML file are closed with the signature set tag closure:

```
</signature-set>
```

0.9.8 Output file formats

Match List files: Match List files will contain one line for each element of the gallery set. Each line contains separated by a TAB (0x9) the ID of the gallery set element and the score (score) achieved by the probe set element the match list file has been generated for. The name of the match list file will be that of the probe set ID followed by **.LST**. The entries are arranged in descending orders of their scores, so the rank of an entry within the gallery set is simply its position in the match list file.

Example: Probe set contains RecordID's 70014503M, 70016321M. Gallery set contains RecordID's 61016303S, 67514503M, 23016321M.

Running **dbcompmatch** will generate 2 files: 70014503M.LST, 70016321M.LST. Assuming that the match list threshold has been quite low and the maximum match list size is more than or equal 3, each of the files contained 3 lines.
The contents of 70014503M.LST might look like

```
61016303S       0.567423
23016321M       0.435522
67514503M       0.22005
```

Score Files: Since generating score matrices can produce onsiderable amount of data a binary format has been chosen for score files. Actually, this format is virtually the same as proposed for the Face Recognition Vendor Test (FRVT) 2002. Each file essentially consists of as many floats as there are elements in the gallery set. There is no information on the gallery set ID's within the score files, this information is available from the gallery contents file instead, see below. The order of the score values in these files is exactly the same as it had been specified when describing the gallery sets, i.e. the order within the result set returned by the SQL statement specifying the gallery set. The name of the score file will be the probe set elements RecordID.

Example using the probe and gallery set elements from above:

Running **dbcompsim** will create 2 files: 70014503M, 70016321M. Each file will contain 3 score values which are the scores achieved by the probe ID against gallery set elements 61016303S, 67514503M, 23016321M, respectively.

The score file format in detail (N is the number of float values stored):

```
Byte Offset     \#of Bytes      Type    Contents
0               8               char    Magic Number = äFRSDKSIMô
8               4               int     Endian Indicator = 0x12345678 on Intel Platform
12              4               int     N (Number of floats contained)
16              4               int     Polarity  = 0 (Score or distance )
20              4 * N           float   Score Value 1...N
4 * N + 20      8               char    Magic Number = 'FRSDKSIM'
```

The Endian indicator may be used to rearrange the bytes for the values starting at offset 12. As long Intel platforms are involved, no rearrangement is required.

There is a little helper tool **dumpsim** which dumps a score file to stdout:

```
dumpsim <name of score file>
```

Gallery/Probe Contents Files: The **dbcompsim** and **dbcompmatch** tools writes two set description files describing the contents of the used probe and gallery set. The names of the files are **probe.toc** and **gallery.toc**. This files contains the descriptions of all used FIRs of the set , one per line, in the order as returned by the SQL statement and used for computation of the score values or match lists. Each line of the files contains 3 elements: the first is the position in the set (starting with 0), the second is a unique record id of the fir (auto generated or from input query), the third is the person Id belonging to that FIR. The entries are delimited by a '|' character.

Using the gallery contents file and the binary score files, you can exactly reproduce the relation between their score values and the PersonID's of the FIR's they have been generated from.

Example: Gallery set query returns 3 rows containing the following entries: [RecordId,PersonID,FIR] like [61016303S,P0,FIR0], [67514503M,P1,FIR1], [23016321M,P2,FIR2].

gallery.toc will look like

```
+----------------- index in gallery set
| +--------------- unique RecordID of the FIR
| |         +----- associated PersonID
| |         |
| |         |
0|61016303S|P0
1|67514503M|P1
2|23016321M|P2
```

Customizing the Output Format

To allow for maximum flexibility in specifying output file formats, the output of the match list and score files is accomplished by use of a shared library, which is mapped by the tools and which can be reimplemented and replaced by the user if required. The library implements the following functions:

```
bool FRsdk::Tools::storeScoreValueFile( ...);

bool FRsdk::Tools::loadScoreValueFile( ...);

bool FRsdk::Tools::storeMatchListFile( ...);

bool FRsdk::Tools::loadMatchListFile( ...);
```

To provide a starting point for customized implementations, the source code of the FaceVACS-SDK implementation is supplied with the SDK. In the **tools** subdirectory of the SDK installation there are the source file **output.cpp** and the Visual C++ workspace and project files **outputlib.dsw** and **outputlib.dsp**.

[Previous chapter]

0.10 Creating shrinked redistributable Packages

Often it is an important issue to shrink the SDK to the components really required.

The following chapters describes which part of the SDK relates to which functionality and which kind of dynamic library hast to be part of the package.

- Before Shrinking
- Compile Libraries to be redistributed
- Compile templates to be redistributed

0.10.1 Before Shrinking

Shrinking of the SDK to the required components only needs a careful analysis of the target platform and on which functionality of the SDK the application is based on.

The following aspects have a strong influence of the size of the shrinked SDK:

- **compiler to develop own applications**: depending on operating system different compiler are supported. msc 12.0, msc 11.0 and msc 10.0 on MS Windows, gcc 4.6 and gcc 4.3 on Linux. So often all other libraries than the one of your choice can be removed from final redistribution package
- **Is Configuration Editor required**: Mostly, end users of applications based on FaceVACS-SDK do not need to modify license or any setting stored in frsdk.cfg. So the Configuration Editor should not be redistributed, too.
- **IPP support available for specific platforms and processors**: On Windows with Intel compatible CPUs, Mac OS X with Intel compatible CPUs and Linux with Intel compatible CPUs. FaceVACS SDK is now statically linked with IPP and MKL libraries and redistribution of related DLLs is not required anymore. On embedded platforms IPP/MKL is not available, which results in a smaller executables and DLLs.
- **which parts of the SDK are really required**: In most cases only a subset of the functionality of FaceVACS SDK is used in end user applications. So, some templates (comparison, portrait characteristics, face/eye finder) are never loaded and can be removed from final redistribution package.

0.10.2 Compile Libraries to be redistributed

Based on simple examples this section will describe how to collect required libraries for redistributable packages:

Example 1: shrinked, small pc hardware with a IA-32 compatible processor (Pentium II, IA-32 compatible), memory footprint is the primary issue. A gcc 4.6 based Linux as operating system.

Example 2: PC hardware (Core 2 Duo), speed is the most important issue. Operating system is MS Windows and development platform is MS Visual Studio 2010 C#

As recommended in Redistribution all required libraries and executables are to be placed in a bin folder in the installation path. Prepare such a directory tree and than copy the executables of your application into the bin folder.

After that copy the FaceVACS SDK library of your choice (according to compiler chain and usage of IPP, see Run--Time Environment for FaceVACS-SDK Applications for a detailed list of available libraries). On Windows take care that the required redistribution packages of VS 2013, VS 2012 or VS 2010 are installed. They are provided with FaceVACS-SDK and located in:

```
bin/x86_[32/64]/msc_[12/11/10]0-ipp_crtdll/vcredist-[x86/x64].exe
```

FaceVACS SDK library delivered for VS2013, VS2012 and VS2010 are build with IPP/MKL support, so these libraries has to be copied, too. It is currently only libiomp5md.dll located inthe reletaed share folder.

Finally copy Configuration Editor binary cfgedit(.exe) and the Qt Library qt-mt336.dll to the bin folder if configuration is required at end user.

Steps for Example 1: After preparing the target directory tree the developed application binary and libfrsdk-8.9.5.-so from FaceVACS SDK in lib/x86_32/gcc-4.6-ipp are to be copied to the target folder and the libiomp5md.dll from share folder.

Steps for Example 2: After preparing the target directory tree the developed application binary and libfrsdk-8.9.5.-dll libfrsdk-8.9.5.dll.manifest from FaceVACS SDK in lib/x86_32/msc_12.0-ipp_crtdll/ are to be copied to the target folder. Additionally libiomp5md.dll has to be copied, too. libfrsdknet-8.9.5.dll should be located in the directory of the devloped application.

0.10.3 Compile templates to be redistributed

All templates are located in the etc sub folder of the FaceVACS installation and it is recommended to copy this structure to the target platform, too.

```
+- etc
  |
  +-cara         contains templates for face finder
  |
  +-ojo          contains templates for face finder
  |
  +-cmp          contains templates for various comparison algorithms
     |           files related to Algorithm A15
     +-a15-0-eyepositions
     +-a15-0-feature-vector.pack
     +-a15-0-Gabor-feature-vector.pack
     +-a15-0-inputimg.jpg
     +-a15-0-inputimg.png
     +-a15-0-inputimg-to-preproc.pack
     +-a15-0-meanstdevimgs.dat
     +-a15-0-regressorX.dat
     +-a15-0-regressorY.dat
     +-farmap-a15-0.dat
     +-frrmap-a15-0.dat
     +-smap-a15-0.dat
     +-trans-a15-0.dat
     |
     |           files related to Algorithm B8
     +-b8-0-eyepositions
     +-b8-0-feature-vector.pack
     +-b8-0-Gabor-feature-vector.pack
     +-b8-0-inputimg.jpg
     +-b8-0-inputimg.png
     +-b8-0-inputimg-to-preproc.pack
     +-b8-0-meanstdevimgs.dat
     +-b8-0-regressorX.dat
     +-b8-0-regressorY.dat
     +-farmap-b8-0.dat
     +-frrmap-b8-0.dat
     +-smap-b8-0.dat
     +-trans-b8-0.dat
     |
     |           files related to 3D Algorithm
     +-farmap-b6ml5.dat
     +-frrmap-b6ml5.dat
     +-smap-b6ml5.dat
     +-isotdb-l5x.dat
     +-isotdb-l5z.dat
     +-trans-l5x.dat
     +-trans-l5z.dat
  |
  +-portrait contains templates for portrait characteristics
     |           templates files of age detector
     +- age-primary.dat
     +- age-secondary-1-10.dat
     +- age-secondary-1-20.dat
     +- age-secondary-1-30.dat
     +- age-secondary-1-40.dat
     +- age-secondary-1-50.dat
     +- age-secondary-1-60.dat
     +- age-secondary-1-70.dat
     +- age-secondary-1-80.dat
     +- age-secondary-1-above.dat
     +- age-secondary-2-10.dat
     +- age-secondary-2-15.dat
     +- age-secondary-2-20.dat
     +- age-secondary-2-25.dat
     +- age-secondary-2-30.dat
     +- age-secondary-2-35.dat
     +- age-secondary-2-40.dat
     +- age-secondary-2-45.dat
     +- age-secondary-2-50.dat
     +- age-secondary-2-55.dat
     +- age-secondary-2-5.dat
     +- age-secondary-2-60.dat
     +- age-secondary-2-65.dat
     +- age-secondary-2-70.dat
     +- age-secondary-2-75.dat
```

```
+- age-secondary-2-80.dat
+- age-secondary-2-85.dat
+- age-secondary-2-above.dat
|           templates files of ethnicity detector
+-ethadisc.dat
+-ethbdisc.dat
+-ethwdisc.dat
|           templates files of test for uniformity of lighting
+-ludisc.dat
+-sphartr.dat
|           templates files of closed eyes detector
+-cedisc.dat
|           templates files of openmouth detector
+-cmdisc.dat
|           templates files of frontal pose detector
+-fpdisc.dat
|           templates files of gender detector
+-gedisc.dat
|           templates files of glasses detector
+-gldisc.dat
|           templates files of gaze frontal detector
+-gadisc.dat
```

There are several ways to reduce the space occupied by the **etc** folder.

- if the comparison algorithm is fixed and will not be changed at end user, all templates file related to other algorithms can be removed
- if no portrait characteristics is required the sub-folder **portrait** can be removed completely
- if comparison (either identification nor enrollment or verification) the sub-folder **cmp** can be removed completely.
- if no eye finding is required the sub-folder **ojo** can be removed completely
- if no face finding is required the sub-folder **cara** can be removed completely

In the following for three use cases the sub folders and files

- **Face Tracking:** only eyes and Face finder templates are used by the face tracker of FaceVACS SDK. So the sub folder **cmp** and **portrait** of **etc** can be removed completely.
- **ISO Test:** Mostly in the images to be processed faces and eyes positions musts be annotated at first. So face and eyes finder is required and than all portrait characteristic templates. Only the sub folder **cmp** can be removed completely.
- **Access Control:** Access control often contains of enrollment, verification and/or identification of given images. Face and eyes positions must be annotated automatically. If no explicit ISO-Test of the images is required the sub folder **portrait** can be removed completely. In sub-folder **cmp** the template file of the unused algorithms can be removed, too.

0.11 Contribution

This chapter describes the contribution package and what contribution contains of and how it can be installed on target platforms.

The following section dscribes the applications, how they can be installed on target platforms and how they can be used.

After installing FaceVACS-SDK contribution is located in subdirectory ***contrib***. all of te contributions requires a working redistribution package as described in section Redistribution

0.12 Tutorial - Overview

This tutorial gives an introduction to programming using the FaceVACS-SDK.

It does not cover all aspects of the FaceVACS-SDK; the objective is teaching the programming philosophy of FaceVACS-SDK and introducing main use cases of face recognition as well as some programming idioms used by the library.

The tutorial is divided into sections describing different use cases or features of the FaceVACS-SDK library.

Tutorial sections:

- Finding faces
- Locating eyes
- Making set enrollments
- Making set verifications
- Making set identifications
- Using eye annotations
- Using the low level match interfaces
- Writing an image acquisition application
- Simple application to crop full frontal images
- Simple application to blend image margin
- How to write a CaptureDevice
- How to write an ImageBody

Remark: Most of the examples uses a helper class CmdLine to interprete the command line parameter (see: CmdLine)

0.13 Tutorial - Finding faces

[Next section]

This small program is a simple face finder, which locates human faces in images.

Here you can find the full source code for the face finding program.

Walk through step by step

```
#include <frsdk/config.h>
```

This line includes library initialization code, mainly FRsdk::Configuration. Creation of most algorithmic classes requires a Configuration argument.

```
#include <frsdk/face.h>
```

This includes the Face::Finder and some supporting classes. The Face::Finder is used to find faces in images. It provides a find() method which takes an image and returns a set of locations in that image where potential faces are located. The Face::Location encapsulates the position of the face as the center point of a line across both eyes in the face.

```
int usage()
```

This is for printing the usage of the program.

```
int main( int argc, char** argv)
{
```

The `main()` function is the entry point of the program.

```
  try {
```

We do the whole work in a try/catch block to catch std::exceptions, which contain error messages if an exception has been raised.

```
    FRsdk::Configuration cfg( argv[1]);
```

Here we create the global library configuration object. This will load and initialize the configuration to be used by the classes where it is passed to upon construction. The argument of the constructor is the filename of a FaceVACS-SDK configuration file, which contains algorithmical settings and the license. Since FRsdk::Configuration is a handle type, copying is inexpensive. (For details regarding the body/handle idiom see body/handle.) you can arbitrarily copy it around to create as many configuration instances as required. Each copy of a given configuration represents the same set of settings and the same license. Of course you can also create different Configurations from different configuration files (e.g in order to create instances of classes with different behaviour within your application).

```
    FRsdk::Face::Finder faceFinder( cfg);
```

Here we get a reference to the Face::Finder. The face finder is build with the settings of the given configuration. It is a handle type, which means that the destruction of the last reference will destroy the build face finder.

Here we create an FRsdk::Image, which is also a handle type, by loading it from the given filename.

optionally the minimal and maximal eye distance of the face finder can be given as command line option.

We call the find method of the finder to get the potential face locations. Face::LocationSet is a commonly used container (STL) for Face::Locations.

We iterate over the face locations returned and print them out in readable format

Major biometric functionalities of FaceVACS-SDK like Acquisition (Face/Eyes Finding), Enrollment etc. have to be explicitely enabled within the license contained in the configuration file. Using an unlicensed feature will cause a FRsdk::FeatureDisabled exception to be thrown.

If a corrupt license or one that has been issued for a different machine is used, a license violation exception will be thrown.

Catching these two particular exceptions will allow for better diagnostic of application errors.

Here we catch std::exceptions and print the reason for the exception. This is crucial for making error diagnostics.

[Previous section] [Next section]

0.14 Tutorial - Locating eyes

[Previous section] [Next section]

This small program is a simple eye finder, which locates eyes in images of human faces.

Here you can find the full source code for the eyes finder.

Walk through step by step

```
#include <frsdk/eyes.h>
```

This line includes the Eyes name-space and some supporting classes. The Eyes::Finder is used to locate the eyes in images of human faces. For that purpose it supports a find method, which takes an image and a face location and returns a list of potential eyes positions.

```
ostream& operator<<( ostream& o, const FRsdk::Position& p)
```

This is a small helper operator for formatted output of FRsdk::Position objects.

```
usage()
```

This is for printing the usage of the program.

```
int main( int argc, const char* argv[] )
{
```

The `main()` function is the entry point of the program.

```
  try {
```

We do the whole work within a try/catch block to catch std::exceptions, which contain error messages if something goes wrong.

```
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));
```

Here we call the global library initialization function. This will load and initialize the algorithmic resources which will be needed by the FaceVACS-SDK. The argument of the init function is the filename of the FaceVACS-SDK configuration file. This file contains configuration keys which are needed to load resources and setup the algorithmic functionality.

```
    FRsdk::Face::Finder faceFinder (cfg);
```

Here we get a reference to the Face::Finder. The face finder is implemented as a singleton. This means that in every case there is only one instance of the face finder during the runtime of the FaceVACS-SDK library. Every call of the finder method returns a reference to that object.

```
FRsdk::Eyes::Finder eyesFinder( cfg);
```

Here we get a reference to the Eyes::Finder. The eyes finder is implemented as a singleton, too.

```
FRsdk::Image img( FRsdk::Jpeg::load( string( cmd.getspaceflag("-img"))));
```

Here we create an image representation (handle) from a jpeg image given by filename. For body/handle idiom see body/handle.

```
float mindist = 0.1f;
if( cmd.getspaceflag("-mineye")){
  mindist = atof( cmd.getspaceflag("-mineye"));
}
float maxdist = 0.4f;
if( cmd.getspaceflag("-maxeye")){
  maxdist = atof( cmd.getspaceflag("-maxeye"));
}
```

optionally the minimal and maximal eye distance of the face finder can be given as command line option.

```
  faceFinder.find (img, mindist, maxdist);
```

We call the find method of the face finder to get the potential face locations. Face::LocationSet is a commonly used container (stl) for Face::Locations.

```
FRsdk::Face::LocationSet::const_iterator faceIter = faceLocations.begin();

while( faceIter != faceLocations.end()) {

  cout << "Face location: " << (*faceIter).pos << endl;

  // doing eyes finding
  FRsdk::Eyes::LocationSet eyesLocations =
    eyesFinder.find( img, *faceIter);
```

We iterate over the returned face locations and do eyes finding. For this purpose we call the find method of the eyes finder to get the potential eyes locations. Eyes::LocationSet is a commonly used container (stl) for Eyes::Locations.

```
  FRsdk::Eyes::LocationSet::const_iterator eyesIter =eyesLocations.begin();
  while( eyesIter != eyesLocations.end()) {
    cout << "Eye locations: [ first " << (*eyesIter).first
         << " confidence=" << (*eyesIter).firstConfidence
         << ", second " << (*eyesIter).second
         << " confidence=" << (*eyesIter).secondConfidence
         << "]" << endl;
    eyesIter++;
  }
```

We iterate over the eye locations returned and print them out.

```
catch( exception& e) {
  cout << e.what() << endl;
  return EXIT_FAILURE;
}
```

Here we catch std::exceptions and print the reason for the exception

[Previous section] [Next section]

0.15 Tutorial - Making set enrollments

[Previous section] [Next section]

This small program makes it possible to enroll a person from a set of images. To run the program see usage().

The example consists of two source files. The file edialog.h implements a dedicated enrollment feedback class and the file enroll.cc contains full source code for the enroller.

Walk through step by step

First we implement our own Enrollment::Feedback class to observe the enrollment processing. An instance of this class prints some messages to stdout and stores the processed FIR to a given file.

```
#include <frsdk/enroll.h>
```

This line includes the enrollment name-space. The main abstractions of this name-space are:

- Enrollment::Processor, which handles the algorithmic part of the enrollment with the enroll function,
- Enrollment::Feedback, which implements observing of the enrollment procedure and returns the enrollment results to the application

```
class EnrolCoutFeedback : public FRsdk::Enrollment::FeedbackBody
{
```

This class implements a dedicated Enrollment::FeedbackBody. The main functionality of this class is to print out some messages to stdout which makes it possible to trace the enrollment and additionally to store the created FIR to a file given by filename.

```
  void start() {
```

This function will be called at the beginning of the enrollment. While in this example only some text is printed, another implementation might provide graphical feedback to the user, e.g. simulate a flashing LED.

```
  void processingImage( const FRsdk::Image& img)
  {
    std::cout << "processing image[" << img.name() << "]" << std::endl;
  }
```

This function indicates that the enrollment algorithm is processing the image img. This can be used to display the image.

```
  void eyesFound( const FRsdk::Eyes::Location& eyeLoc)
  {
    std::cout << "found eyes at ["<< eyeLoc.first
              << " " << eyeLoc.second << "; confidences: "
              << eyeLoc.firstConfidence << " "
              << eyeLoc.secondConfidence << "]" << std::endl;
  }
```

This function indicates that eyes have been found in the current image. Using the eye positions and the image supplied before some graphical feedback might be implemented in this function.

```
  void eyesNotFound()
  {
    std::cout << "eyes not found" << std::endl;
  }
```

This function will be called if the eyes could not be located in the image. The enrollment algorithm will stop processing this image because it does presumably not contain a human face.

```
void success( const FRsdk::FIR& fir_)
{
  fir = new FRsdk::FIR(fir_);
  std::cout
    << "successful enrollment";
  if(firFN != std::string("")) {

    std::cout << " FIR[filename,id,size] = [\""
    << firFN.c_str() << "\",\"" << (fir->version()).c_str() << "\","
    << fir->size() << "]";
    // write the fir
    std::ofstream firOut( firFN.c_str(),
                          std::ios::binary|std::ios::out|std::ios::trunc);
    firOut << *fir;
  }
```

On successful enrollment the FIR of the person enrolled will be returned to the application. Here we store the FIR to a file whose name was given as an argument to the Feedback constructor.

```
void failure() { std::cout << "failure" << std::endl; }
```

If something goes wrong during enrollment this function will be called.

```
void end() { std::cout << "end" << std::endl; }
```

Indicates the end of the enrollment.

```
const FRsdk::FIR& getFir() const {
```

Gets access to the FIR after successfully processing of the given images. If the processor was not able to generate a FIR calling this function will throw an exception

Now we are ready to write the real enroller.

```
#include "edialog.h"
```

The enrollment feedback class definition must be included.

```
int usage()
```

Print out the usage of the program.

```
int main( int argc, const char* argv[] )
{
```

The program's entry point.

```
  FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));
```

Here we call the global library initialization function. This will load and initialize the algorithmic resources which will be needed by the FaceVACS-SDK. The argument of the init function is the filename of the FaceVACS-SDK configuration file. This file contains configuration keys which are needed to load resources and setup the algorithmic software.

```
FRsdk::SampleSet enrollmentImages;
```

We create a container for the enrollment images.

Here we load the jpeg images given by command line.

Create an enrollment processor.

Now we have to create a FRsdk::Feedback object which later can be used as an input argument to the enroll function of the Processor. According to the body/handle pattern we create an Enrollment::Feedback from a pointer to an object which is inherited from Enrollment::FeedbackBody.

If you take a look at the Feedback constructor: FRsdk::Feedback::Feedback(const CountedPtr<FeedbackBody>&); it actually takes a CountedPtr as its argument. This is created as a temporary object by the compiler. We have to create the EnrolCoutFeedback dynamically (with 'new'). Otherwise, i.e. if the address of a stack object was passed to CountedPtr constructor, an error will occur upon destruction of the CountedPtr, since this tries to delete the object passed.

To do the enrollment we only have to call the process function of the Enrollment::Processor. The first two arguments of the process function describes the set of enrollment images. The last argument is the feedback object for observing the enrollment and returning results to the application.

We catch exceptions and print out the reasons here.

[Previous section] [Next section]

0.16 Tutorial - Making set verifications

[Previous section] [Next section]

This small program makes it possible to verify (authenticate) a person from a set of images.

The example consists of two source files. The file vdialog.h implements a dedicated verification feedback class and the file verify.cc contains full source code for the verifier.

Walk through step by step

First we implement our own Verification::Feedback class to observe the verification processing. An instance of this class prints some messages to stdout.

```
#include <frsdk/verify.h>
```

This line includes the Verification name-space. The main abstractions of this name-space are:

- Verification::Processor, which handles the algorithmic part of the verification with the Verification::Processor's process function,

- Verification::Feedback, which supports the watching of the verification procedure and returns the authentication result to the application

```
class VerifyCoutFeedback : public FRsdk::Verification::FeedbackBody
{
```

This class implements a dedicated Verification::FeedbackBody. The main functionality of this class is to print out some messages to stdout which makes it possible to trace the verification.

```
  void start() { std::cout << "start" << std::endl; }
```

This function will be called at the beginning of the verification. We are only printing out text here but another implementation could give some graphical user interface feedback, e.g. switching some state indicator.

```
  void processingImage( const FRsdk::Image& img)
  {
    std::cout << "processing image[" << img.name() << "]" << std::endl;
  }
```

This function indicates that the verification algorithm is processing the image img. This can be used to display the image.

```
  void eyesFound( const FRsdk::Eyes::Location& eyeLoc)
  {
    std::cout << "found eyes at ["<< eyeLoc.first
              << " " << eyeLoc.second << "; confidences: "
              << eyeLoc.firstConfidence << " "
              << eyeLoc.secondConfidence << "]" << std::endl;
  }
```

This function indicates that eyes have been found in the current image. Using the eye positions and the image supplied before some graphical feedback might be implemented in this function.

```
  void eyesNotFound() {
    std::cout << "eyes not found" << std::endl;
  }
```

This function will be called if the eyes could not be located in the current image. The verification algorithm will stop processing this image because presumeably there is no human face in the image.

```
  void match( const FRsdk::Score& s)
  {
    std::cout << "match got Score[" << (float)s << "]" << std::endl;
  }
```

The result of the verification is given back to the application. The current image has been compared with the FRsdk::FIR to verify against and obtained the score s.

```
  void success() {
    std::cout << "successful verification." << std::endl;
  }
```

The success function will be called only if one of the images taken for verification gets a score higher than the threshold value passed in process().

```
  void failure() { std::cout << "failure" << std::endl; }
```

If something goes wrong during verification this function will be called.

```
  void end() { std::cout << "end" << std::endl; }
```

Indicates the end of the verification.

Now lets write the real verifier.

```
#include "vdialog.h"
```

The verification feedback class definition must be included.

```
int usage()
```

Print out the usage of the program.

```
int main( int argc, const char* argv[] )
{
```

The program's entry point.

```
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));
```

Here we call the global library initialization function. This will load and initialize the algorithmic resources which will be needed by the FaceVACS-SDK. The argument of the init function is the filename of the FaceVACS-SDK configuration file. This file contains configuration keys which are needed to load resources and setup the algorithmic software.

```
    FRsdk::SampleSet verificationImages;
```

We create a container for the enrollment images.

Here we load the jpeg images to be verified given by command line.

The FIR to verify the verification images against will be loaded.

The result of the comparison between the FIR to be verified and the FIR created from the verification images is a score. To decide if the person matches we need a threshold. The FRsdk::requestFAR function returns a score value which can be used as a threshold for the verification. In the example we request a FAR (false acceptance rate) of 0.1 percent. That means that we require that on average there is only in 1 of 1000 cases a positive result of the verification if the person to be verified is **not** the person for which the verification FIR was generated. However, this applies to a specific database only (that one where the FAR/FRR curves have been sampled upon), results will vary with different image sets. For a more detailed explanation of FAR and FRR semantics see the FaceVACS-SDK User Guide.

We create the Verification::Feedback with an instance of the FeedbackBody VerifyCoutFeedback. This serves later as input parameter to the process function of the Verification::Processor.

Create an instance of a Verification::Processor

To start the verification we call the process function of the processor. The feedback object traces the progress of the verification process and prints the results.

We catch std::exception here to get an error message if something goes wrong during verification and set the program's return value accordingly.

[Previous section] [Next section]

0.17 Tutorial - Making set identifications

[Previous section] [Next section]

This small program makes it possible to identify a person from a population of FIR's. The identification takes a set of images which will be used for identification.

The example consists of two source files. The file idialog.h implements a dedicated identification feedback class and the file identify.cc contains full source code for the identifier.

Walk through step by step

First we implement our own Identification::Feedback class to observe the identification processing. An instance of this class prints some messages to stdout.

```
#include <frsdk/ident.h>
```

This line includes the Identification name-space. The main abstractions of this name-space are:

- Identification::Processor, which handles the algorithmic part of the identification with the Identification::-Processor's process function,
- Identification::Feedback, which supports the watching of the identification procedure and returns the authentication result to the application

```
class IdentifyCoutFeedback : public FRsdk::Identification::FeedbackBody
{
```

This class implements a dedicated Identification::FeedbackBody. The main functionality of this class is to print out some messages to stdout which makes it possible to trace the identification.

```
  void start() { std::cout << "start" << std::endl; }
```

This function will be called at the beginning of the identification. We are only printing out text here but another implementation could do some graphical feedback, e.g. switching some state indicator.

```
  void processingImage( const FRsdk::Image& img)
  {
    std::cout << "processing image[" << img.name() << "]" << std::endl;
  }
```

This function indicates that the identification algorithm is processing the image img. This can be used to display the image.

```
void eyesFound( const FRsdk::Eyes::Location& eyeLoc)
{
  std::cout << "found eyes at ["<< eyeLoc.first
            << " " << eyeLoc.second << "; confidences: "
            << eyeLoc.firstConfidence << " "
            << eyeLoc.secondConfidence << "]" << std::endl;
}
```

This function indicates that eyes have been found in the current image. Using the eye positions and the image supplied before some graphical feedback might be implemented in this function.

```
void eyesNotFound()
{
  std::cout << "eyes not found" << std::endl;
}
```

This function will be called if the eyes could not be located in the current image. The identification algorithm will further processing because presumeably there is no human face in the image.

```
void matches( const FRsdk::Matches& matches)
{
  FRsdk::Matches::const_iterator iter = matches.begin();

  while( iter != matches.end()) {
    FRsdk::Match m = *iter;
    std::cout << "match on fir[" << m.first << "] got Score["
         << (float)m.second << "]" << std::endl;
    iter++;
  }
} //matches
```

The result of the identification is given back to the application. From each of the images given as input to the identification a FIR is built and compared with the given population of FIR's. The matches function returns a stl container with Identification::Matches entries ordered by score value. The first entry has the best match and the last one the worst. The entries of that container contain an id of the FIR which matches along with the score of the match. We iterate the container here and print the matches.

```
void end() { std::cout << "end" << std::endl; }
```

Indicates the end of the identification.

Now let's write the identification program.

```
#include "idialog.h"
```

The identification feedback class definition must be included.

```
int usage()
```

Print out the usage of the program.

```
int main( int argc, const char* argv[] )
{
```

The program's entry point.

```
  FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));
```

Here we call the global library initialization function. This will load and initialize the algorithmic resources which will be needed by the FaceVACS-SDK. The argument of the init function is the filename of the FaceVACS-SDK configuration file. This file contains configuration keys which are needed to load resources and setup the algorithmic functionality.

```
FRsdk::SampleSet identificationImages;

identificationImages.push_back
  ( FRsdk::Sample(FRsdk::Jpeg::load( string( cmd.getspaceflag("-img")))));
```

We create a container for the images to be identified. Please note that we use only one image for that example. The jpeg image to identify given by command line will be loaded.

```
FRsdk::Population population( cfg);
FRsdk::FIRBuilder firBuilder( cfg);

std::string firs = cmd.getspaceflag("-firs");
size_t pos = 0;
size_t fpos = 0;

while( (pos = firs.find(',', fpos)) != std::string::npos){
  std::string firn = firs.substr( fpos, pos-fpos);
  fpos = pos +1;
  cout  << "[" << firn << "]" << endl;
  ifstream firIn( firn.c_str(), ios::in|ios::binary);
  population.append( firBuilder.build( firIn), firn.c_str());
}
if( fpos < firs.size()){
  std::string firn = firs.substr( fpos, pos-fpos);
  cout  << "[" << firn << "]" << endl;
  ifstream firIn( firn.c_str(), ios::in|ios::binary);
  population.append( firBuilder.build( firIn), firn.c_str());
} // population complete
```

We create a container for the FIR population and load the FIR's using a FRsdk::FIRBuilder.

```
FRsdk::ScoreMappings sm( cfg);
FRsdk::Score score = sm.requestFAR( 0.001f);
if( cmd.getspaceflag("-far")){
  if( cmd.getspaceflag("-frr") || cmd.getspaceflag("-score")) return usage();
}
if( cmd.getspaceflag("-frr")){
  if( cmd.getspaceflag("-score")) return usage();
}

if( cmd.getspaceflag("-frr")){
  score = sm.requestFRR( atof( cmd.getspaceflag("-frr")));
}
if( cmd.getspaceflag("-far")){
  score = sm.requestFAR( atof( cmd.getspaceflag("-far")));
}
if( cmd.getspaceflag("-score")){
  score = FRsdk::Score( atof( cmd.getspaceflag("-score")));
}
cout << "used matching threshold: " << score << endl;
```

The result of the comparison between the FIR's of the identification population and the FIR created from the identification images is a score. To decide if the person matches we need a threshold. The FRsdk::requestFAR function returns a score value which can be used as a threshold for the identification. In the example either a score can be selected by command line or a score can be selected by a given FAR or FRR.

```
unsigned int numofmatches = 3;
if( cmd.getspaceflag("-maxmatch")){
  numofmatches = atoi( cmd.getspaceflag("-maxmatch"));
}
```

Another parameter of identification is the maximal match list size. This parameter limits the returened match list to the best numofmatches matches.

```
FRsdk::Identification::Feedback feedback( new IdentifyCoutFeedback());
```

We create the Identification::Feedback with an instance of the FeedbackBody IdentifyCoutFeedback. Later this will be passed to the process function of the Identification::Processor.

```
FRsdk::Identification::Processor proc( cfg, population);
```

Create an instance to an Identification::Processor passing a population of FIR's to identify against. It can be useful to have several instances of the Identification::Processor in parallel created from different FIR populations to allow for customized identification queries.

```
proc.process( identificationImages.begin(), identificationImages.end(),
              score, feedback, numofmatches);
```

To start the identification we call the process function of the processor. The feedback object traces the progress of the identification process and prints the results. The maximum size of the Identification::MatchSet we want to receive in the feedback is passed as the last argument.

```
catch( exception& e) {
  cout << e.what() << endl;
  return EXIT_FAILURE;
}
```

We catch std::exception here to get an error message if something goes wrong during identification setting the return value of the program accordingly.

[Previous section] [Next section]

0.18 Tutorial - Using eye annotations

[Previous section] [Next section]

In some cases, it might be required to provide eye positions obtained from elsewhere to the algorithm instead of having it find themself. This might be the case if the FRsdk::Face::Finder or FRsdk::Eyes::Finder are not able to find the positions or provide erroneous results. Since the FRsdk::Sample type can optionally contain eye posisitions, the processor interfaces to support this use case are the same as with automatic eye finding. The only difference is that the Samples passed to the process() function are generated out of a FRsdk::AnnotatedImage instead of just a FRsdk::Image.

The example verifyan.cc illustrates the creation and use of annotated images for verification:

```
FRsdk::Eyes::LocationSet eyesLocations =
  eyesFinder.find( img, *faceIter);

FRsdk::Eyes::LocationSet::const_iterator eyesIter =
  eyesLocations.begin();
while( eyesIter != eyesLocations.end()) {
  cout << "Eye locations: [ first " << (*eyesIter).first
       << ", second " << (*eyesIter).second << "]" << endl;

  FRsdk::AnnotatedImage a(img, *eyesIter);
  verificationSamples.push_back( FRsdk::Sample( a));
  eyesIter++;
}
```

For this tutorial application, we use the face and eyes finder to get the eye positions. In a real-world application, the eye positions would have been obtained from elsewhere. The positions are stored along with the images they belong to as instances of FRsdk::AnnotatedImage, which are then used to create FRsdk::Sample s.

```
FRsdk::Verification::Processor proc( cfg);

// do the verification
proc.process( verificationSamples.begin(),
              verificationSamples.end(),
              fir, score, feedback);
```

The processor performs the verification for the image set.

[Previous section] [Next section]

0.19 Tutorial - Using the low level match interfaces

[Previous section] [Next section]

In cases where the FIRs of both probe and gallery elements can be precomputed, using these instead of images considerably accelerates biometric computations. This can be helpful e.g for implementing applications for fast search in face databases, but also to make biometric evaluations on large image sets (computation of score matrices or match lists). For this purpose, the low level matching functions FRsdk::compare(const FIR&, constFIR&), FRsdk::compare(const FIR&, const Population&) and FRsdk::bestMatches are provided. The FRsdk::oneToOne function returns the score between the given FIR's and the FRsdk::oneToMany function calculates a set of named scores (FRsdk::Matches) between a single FIR and a Population. FRsdk::bestMatches returns the FIR's best matches from a population comparing better than a certain threshold.

The following example match.cc demonstrates how to use these functions.

```
// load the fir
ifstream firStream( cmd.getspaceflag("-probe"), ios::in|ios::binary);
FRsdk::FIRBuilder firBuilder( cfg);
```

We load the single fir to be the person to test against the population.

```
// load the fir population for identification
std::string firs = cmd.getspaceflag("-gallery");
size_t pos = 0;
size_t fpos = 0;
FRsdk::Population population( cfg);
while( (pos = firs.find(',', fpos)) != std::string::npos){
  std::string firn = firs.substr( fpos, pos-fpos);
  fpos = pos +1;
  cout  << "[" << firn << "]" << endl;
  ifstream firIn( firn.c_str(), ios::in|ios::binary);
  population.append( firBuilder.build( firIn), firn.c_str());
}
```

The population is constructed by loading the given FIR's by their filenames and add them to the Population.

```
// bestMatches() takes care about the configured number of threads
// to be used in comparison algorithm.
FRsdk::CountedPtr<FRsdk::Matches> matches =
  me.bestMatches( fir, population, FRsdk::Score( score), numofmatches);
```

The call to the FRsdk::bestMatches function calculates a set of FRsdk::Match(es) which represents the named ordered scores for the match of each of the population FIR's against the FIR of the person to match against.

There is a faster function available if only an unordered list of scores is required.

```
//compare() does not care about the configured number of Threads
//for the comparison algorithm. It uses always one thrad to
//compare all inorder to preserve the order of the scores
//according to the order in the population (orer of adding FIRs to
//the population)
FRsdk::CountedPtr<FRsdk::Scores> scores = me.compare( fir, population);
```

It compares the FIR against the population and returns the scores in same order as FIR's within Population.

The code for doing a match between two FIR's looks like

```
FRsdk::FIR firA = ...
FRsdk::FIR firB = ...

FRsdk::Score s = FRsdk::compare( firA, firB);
```

[Previous section] [Next section]

0.20 Tutorial - Simple image acquisition application

[Previous section] [Next section]

This small program is a simple image acquisition application, which proceeds the following steps:

```
- load image from file
- locate face and eyes
- analyse portrait characteristics
- produce ISO 19794 conform Token Face Image
- store the image using ISO/IEC JTC 1 SC 37 minimum simple patron format
```

Here you can find the full source code for the acquisition program.

Walk through step by step

```
#include <frsdk/cbeff.h>
#include <frsdk/config.h>
#include <frsdk/eyes.h>
#include <frsdk/jpeg.h>
#include <frsdk/j2k.h>
#include <frsdk/bmp.h>
#include <frsdk/png.h>
#include <frsdk/pgm.h>
#include <frsdk/portrait.h>
#include <frsdk/portraittests.h>
#include <frsdk/tokenface.h>
```

These lines include the needed namespaces and supporting classes.

```
ostream& operator<<( ostream& o, const FRsdk::Position& p) {
```

This is a small helper operator for formatted output of FRsdk::Position objects.

```
  class AcquisitionError: public std::exception
```

This is a dedicated acquisition exception class

```
int usage()
```

This is for printing the usage of the program.

```
int main( int argc, const char* argv[] )
{
```

The `main()` function is the entry point of the program.

```
  try {
```

We do the whole work within a try/catch block to catch std::exeptions, which contain error messages if something goes wrong.

```
    FRsdk::Configuration cfg( configIStream);
```

Here we create the global library configuration (handle). This will load and initialize the configuration which will be used by the FaceVACS-SDK. The argument of the constructor is either the filename of the FaceVACS-SDK configuration file or a std::istream. This file/stream contains configuration keys which are needed to load resources and setup the algorithmic functionality.

```
FRsdk::Face::Finder faceFinder( cfg);
```

Here we get a reference to the Face::Finder. The face finder is implemented as a singleton. This means that in every case there is only one instance of the face finder during the runtime of the FaceVACS-SDK library. Every call of the finder method returns a reference to that object.

```
FRsdk::Eyes::Finder eyesFinder( cfg);
```

Here we get a reference to the Eyes::Finder. The eyes finder is implemented as a singleton, too.

```
FRsdk::Portrait::Analyzer portraitAnalyzer( cfg);
```

Now we create the Portrait::Analyzer facility which is a singleton too.

```
    ( FRsdk::Jpeg::load
```

Here we create an image representation (handle) from a jpeg image given by filename. For body/handle idiom see body/handle. We do so in a try block in case the image file is not in the jpeg format. We repeat this step for various other supported image file formats. As an alternative here a CaptureDevice could be used to grab some images from the video device.

```
  faceFinder.find (*img, mindist, maxdist);
```

We call the find method of the face finder to get the potential face locations. Face::LocationSet is a commonly used container (stl) for Face::Locations.

```
if( faceLocations.size() < 1) {
  throw AcquisitionError( "Unable to locate face");
} else {
  const FRsdk::Face::Location& l = faceLocations.front();
  cout << "Found face: " << l.pos <<", width="
       << l.width << ", confidence="
       << l.confidence << endl;
```

We check if we got a face location and do some debug output of the location of the found face, otherwise we throw a acquisition exception.

```
FRsdk::Eyes::LocationSet eyesLocations =
  eyesFinder.find (*img, faceLocations.front());
```

We use the first face location (the one with the best confidence) to locate the eyes.

```
if( eyesLocations.size() < 1) {
  throw AcquisitionError( "Unable to locate eyes");
} else {
  const FRsdk::Eyes::Location& l = eyesLocations.front();
  cout << "Found eyes: first " << l.first
       << " confidence=" << l.firstConfidence << endl
       << "             second " << l.second
       << " confidence=" << l.secondConfidence
       << endl;
```

We check if we got eyes locations and do some debug output of the location of the found eyes, otherwise we throw a acquisition exception.

```
FRsdk::AnnotatedImage annotatedImage( *img, eyesLocations.front());
```

The image together with the eye locations are bundled to a FRsdk::AnnotatedImage for convenience.

```
FRsdk::Portrait::Characteristics pc =
  portraitAnalyzer.analyze( annotatedImage);
```

We call the analyze function of the portraitAnalyzer to get the characteristics of that face and print them out.

```
FRsdk::Portrait::Feature::Set features = featureTest.assess( pc);
```

We test for features in the portrait using the returned Portrait::Characteristics:

```
FRsdk::ISO_19794_5::FullFrontal::Compliance
  isoCompliance = iso19794Test.assess( pc);
```

We determine compliance with ISO_19794_5 passing the Portrait::Characteristics again.

```
if( !isoCompliance.isCompliant()) {
  cout << "Acquired image " << cmd.getspaceflag("-img")
       << " not compliant with ISO 19794-5 requirements"
       << endl;
} else {
  cout << "Acquired image " << cmd.getspaceflag("-img")
       << " compliant with ISO 19794-5 requirements" << endl;
```

Test compliance with ISO_19794_5, determine reasons for problems and provide diagnostics to the user.

```
FRsdk::AnnotatedImage iso19794Img = tfcreator.extract( annotatedImage);
```

The annotated image we have now should be compliant to ISO_19794_5. This standard defines portrait properties required for electronic passports (TokenFace image type). One of the properties are relative or absolute eye positions and minimum image dimensions. The FRsdk provides functions to produce TokenFace images. We use the FRsdk::extractMinimalTokenFaceImage function here which generates a TokenFace image with minimum image dimensions to minimize the storage space.

```
FRsdk::AnnotatedImageSet annotatedImages;
annotatedImages.push_back( iso19794Img);
```

The CBEFF (Common Biometric Exchange Formats Framework) Token Face image patron format supports more than one face to be stored, so we have to bundle a list which in this case contains only one face.

```
  std::ofstream out( cmd.getspaceflag("-cbeff"),
                     std::ios::out | std::ios::binary);
  FRsdk::ISO_19794_5::TokenFace::write( out, annotatedImages);
```

The face is now stored to the file in CBEFF format.

```
    std::ofstream outimg (cmd.getspaceflag("-token"),
                          std::ios::out | std::ios::binary);
    // FRsdk::Jpeg::save (iso19794Img.first, outimg); // save jpeg image
    // FRsdk::Bmp::save (iso19794Img.first,
    //  std::string (cmd.getspaceflag("-token"))); // save bmp image
    // FRsdk::Pgm::save (iso19794Img.first,
    //  std::string (cmd.getspaceflag("-token"))); // save pgm image
    FRsdk::Png::save (iso19794Img.first, outimg); // save png image
```

Optionally the face can be saved in an image file. Here the png format is used. The other possible formats are shown as comments.

```
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }
```

Here we catch std::exceptions and print the reason for the exception

[Previous section] [Next section]

0.21 Tutorial - Simple application to crop full frontal images

[Previous section] [Next section]

This small program is a simple application to crop face images to fit to geometrical characteristics of full frontal specification, which proceeds the following steps:

- load jpeg image from file
- locate face and eyes
- extract full frontal image
- compliance test of modified characteristics to fit full frontal specifications

Here you can find the full source code for the cropping program.

Walk through step by step

```
#include <frsdk/cbeff.h>
#include <frsdk/config.h>
#include <frsdk/eyes.h>
#include <frsdk/jpeg.h>
#include <frsdk/portrait.h>
#include <frsdk/portraittests.h>
#include <frsdk/fullfrontal.h>
#include <frsdk/tokenface.h>
```

These lines include the needed namespaces and supporting classes.

```
ostream& operator<<( ostream& o, const FRsdk::Position& p)
```

This is a small helper operator for formatted output of FRsdk::Position objects.

```
  class AcquisitionError: public std::exception
```

This is a dedicated acquisition exception class

```
int usage()
```

This is for printing the usage of the program.

```
int main( int argc, const char* argv[] )
{
```

The `main()` function is the entry point of the program.

```
  try {
```

We do the whole work within a try/catch block to catch std::exeptions, which contain error messages if something goes wrong.

```
FRsdk::Configuration cfg( configIStream);
```

Here we create the global library configuration (handle). This will load and initialize the configuration which will be used by the FaceVACS-SDK. The argument of the constructor is either the filename of the FaceVACS-SDK configuration file or a std::istream. This file/stream contains configuration keys which are needed to load resources and setup the algorithmic functionality.

```
FRsdk::Face::Finder faceFinder( cfg);
```

Here we get a reference to the Face::Finder. The face finder is implemented as a singleton. This means that in every case there is only one instance of the face finder during the runtime of the FaceVACS-SDK library. Every call of the finder method returns a reference to that object.

```
FRsdk::Eyes::Finder eyesFinder( cfg);
```

Here we get a reference to the Eyes::Finder. The eyes finder is implemented as a singleton, too.

```
FRsdk::Portrait::Analyzer portraitAnalyzer( cfg);
```

Now we create the Portrait::Analyzer facility which is a singleton too.

```
FRsdk::ISO_19794_5::FullFrontal::Test iso19794Test( cfg);
FRsdk::Portrait::Feature::Test featureTest( cfg);
FRsdk::ISO_19794_5::FullFrontal::Creator ffcreator( cfg);
```

Now we create the class to extract full frontal images. Other classe are created previously in order to test resulting image.

```
FRsdk::Image img( FRsdk::Jpeg::load( string( cmd.getspaceflag("-img"))));
```

Here we create an image representation (handle) from a jpeg image given by filename. For body/handle idiom see body/handle. As an alternative here a CaptureDevice could be used to grab some images from the video device.

```
  faceFinder.find( img, mindist, maxdist);
```

We call the find method of the face finder to get the potential face locations. Face::LocationSet is a commonly used container (stl) for Face::Locations.

```
if( faceLocations.size() < 1) {
  throw AcquisitionError( "Unable to locate face");
} else {
```

We check if we got a face location and do some debug output of the location of the found face, otherwise we throw a acquisition exception.

```
FRsdk::Eyes::LocationSet eyesLocations =
  eyesFinder.find( img, faceLocations.front());
```

We use the first face location (the one with the best confidence) to locate the eyes.

```
    if( eyesLocations.size() < 1) {
      throw AcquisitionError( "Unable to locate eyes");
    } else {
```

We check if we got eyes locations and do some debug output of the location of the found eyes, otherwise we throw a acquisition exception.

```
    FRsdk::AnnotatedImage annotatedImage( img, eyesLocations.front());
```

The image together with the eye locations are bundled to a FRsdk::AnnotatedImage for convenience.

```
    annotatedImage = ffcreator.extract( annotatedImage);
```

extract full frontal image.

```
      portraitAnalyzer.analyze( annotatedImage);

    // Test features
    FRsdk::Portrait::Feature::Set features = featureTest.assess( pc);

    // Glasses
    if( features.wearsGlasses()) {
      cout << "Feature test: Person with glasses. (" << pc.glasses()
           << ")" << endl;
    } else {
      cout << "Feature test: Person without glasses. (" << pc.glasses()
           << ")" << endl;
    } // Glasses


    // Test compliance with ISO 19794-5 Full Frontal image requirements
    FRsdk::ISO_19794_5::FullFrontal::Compliance isoCompliance =
      iso19794Test.assess( pc);

    if( !isoCompliance.onlyOneFaceVisible()) {
      cout << "More than one face is visible!" << endl;
    }
    if( !isoCompliance.goodVerticalFacePosition()) {
      cout << "Bad vertical face position!" << endl;
    }
    if( !isoCompliance.horizontallyCenteredFace()) {
      cout << "Face not centered horizontally!" << endl;
    }
    if( !isoCompliance.widthOfHead())  {
        cout << "Bad sizing (Width)!" << endl;
    }
    if( !isoCompliance.lengthOfHead()) {
      cout << "Bad sizing (Height)!" << endl;
    }
    if( !isoCompliance.resolution()) {
      cout << "Bad resolution (not enough pixels of head width)!" << endl;
    }
    if( !isoCompliance.goodExposure()) {
      cout << "Bad exposure!" << endl;
    }
    if( !isoCompliance.goodGrayScaleProfile()) {
      cout << "Gray scale profile is not good!" << endl;
      cout << "Gray scale density: " << pc.grayScaleDensity() << endl;
    }
    if ( pc.isColor() ) {
      if( !isoCompliance.hasNaturalSkinColour()) {
        cout << "No natural Skin colour!" << endl;
        cout << "Natural skin colour: " << pc.naturalSkinColour() << endl;
      }
    }
    if( !isoCompliance.noHotSpots()) {
      cout << "Hot spots!" << endl;
      cout << "Hot spots: " << pc.hotSpots() << endl;
    }
    if( !isoCompliance.isFrontal()) {
      cout << "Face is not frontal!" << endl;
      cout << "Deviation from frontal pose: "
           << pc.deviationFromFrontalPose() << endl;
      cout << "Pose angle roll: " << pc.poseAngleRoll() << endl;
    }
#ifndef UNDER_CE
    if( !isoCompliance.isLightingUniform()) {
```

```
      cout << "Lighting is not uniform!" << endl;
      cout << "Deviation from uniform lighting: "
           << pc.deviationFromUniformLighting() << endl;
    }
#endif
    if( !isoCompliance.isSharp()) {
      cout << "Sharpness does not fit requirements!" << endl;
      cout << "Sharpness: " << pc.sharpness() << endl;
    }
    if( !isoCompliance.noTintedGlasses()) {
      cout << "Tinted glasses!" << endl;
      cout << "Tinted Eyes confidence: ["
           << pc.eye0Tinted() << ", " << pc.eye1Tinted() << "]" << endl;
    }
    if( !isoCompliance.isCompliant()) {
      cout << "Acquired image not compliant with ISO 19794-5 requirements!"
           << endl;
    } else {
      cout << "Cropped image seems to be ISO 19794-5 compliant." << endl;
    }

    cout << "Processing done" << endl;
```

now the extracted image will be tested to be compliant or not compliant. Because of the geometrical modification only done by the extraction process some tests may still fail: Glasses, mouth, lighting, sharpness, pitch are not modified.

optionally the extracted image will be saved

[Previous section] [Next section]

0.22 Tutorial - How to use vignetting function to blend image borders

[Previous section] [Next section]

This small programm is a simple application to show how image border can be blenden to a defined color. The method is named vignetting and well known in photographic areas. Vignetting means a intensity blending of margin regions by certain reasons (mechanical, optical, digital).

Let's take a look into the source code.

At first we have to read configuration

```
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));
```

Next step is reading the image in jpeg format. the second application argument contains the file name to be read

```
    FRsdk::Image inimg = FRsdk::Jpeg::load( string( cmd.getspaceflag("-img")));
```

Now the image to be processed is known und blending of margin can be done. There are three ways of blending the margin.

1) Gaussian: It means the margin is blend smoothly by changing the intensity of target color to existing pixel values. The function to blend from image to border is a half gaussian.

```
      ( inimg, FRsdk::Rgb( 0xff, 0xff, 0xff), 20, 10, FRsdk::Gaussian);
```

2) Linear: It means the margin is blend smoothly by changing the intensity of target color to existing pixel values. The function to blend from image to border is proportional to the border distanze.

```
      ( inimg, FRsdk::Rgb( 0xff, 0xff, 0xff), 20, 10, FRsdk::Linear);
```

3) Fixed: Linear: It means the margin is to border color.

```
( inimg, FRsdk::Rgb( 0xff, 0xff, 0xff), 20, 10, FRsdk::Fixed);
```

In all cases the third function parameter defines the margin width in pixel and the forth means the radius of round edges. The last parameter is the color to be blend to. see Frsdk::Rgb. Only red, green and blue channel will be used for blending.

[Previous section] [Next section]

0.23 Tutorial - How to write a CaptureDevice

[Previous section] [Next section]

All of the Processor classes provide interfaces to operate with video streams instead of image sets. The abstraction representing a video stream within FaceVACS-SDK is FRsdk::CaptureDevice. However, only video sources based on Win32 DirectShow are supported by FaceVACS-SDK. To provide a CaptureDevice supporting e.g. some particular frame grabber, a class derived from FRsdk::CaptureDeviceBody has to be implemented covering the specifics of this frame grabber. For sake of simplicity in this tutorial we demonstrate how to implement a simple file based capturing device. The FileCaptureDevice takes a list of file names in the constructor and by calling the capture() function one can iterate through this list, loading the image from the current file. If the end of the list is reached, the iteration starts from the beginning.

Let's see how the source code for the file capturing device looks like:

```
#include <frsdk/capdev.h>
```

We include the FRsdk::CaptureDevice and FRsdk::CaptureDeviceBody abstraction.

```
class FileCaptureDevice : public FRsdk::CaptureDeviceBody
{
```

The CaptureDevice uses the body/handle idiom. For that reason we have to inherit from the abstract FRsdk::-CaptureDeviceBody.

```
  FileCaptureDevice( const std::list< std::string>& filenames)
  {
    // construction with 0 filenames is not allowed
    if( filenames.size() == 0)
      throw new std::out_of_range( "We need at least one filename for "
                                   "the file capturing device.");

    // load images from file names
    std::list< std::string>::const_iterator it( filenames.begin());
    while( it != filenames.end()) {
      std::cout << "loading [" << (*it).c_str() << "] ";
      images.push_back( FRsdk::Jpeg::load( (*it++)));
      std::cout << "done." << std::endl;
    }

    imgIter = images.begin();
  }//c'tor
```

The c'tor takes a list of filenames as input. The filenames describe the set of images used for capture. First we make sure that at least one image was given. Then we iterate over the list of image names and load them. We set the internal iterator to the start position of the image set.

```
  FRsdk::Image capture() const
  {
    if( imgIter == images.end()) imgIter = images.begin();
    return *(imgIter++);
  }
```

We return the current image and if the internal iterator points to the last list entry it is resetted to the begin of the list.

```
private:
  FRsdk::ImageSet images;
  mutable FRsdk::ImageSet::const_iterator imgIter;
};
```

The images are stored internally in a FRsdk::ImageSet. Note that the iterator which caches the current image iterator position must be mutable due to the const definition of the capture() function.

The code for creating a CaptureDevice which uses the FileCaptureDevice as implementation (body) looks like:

```
list<string> imageNames;
// fill the list here
FRsdk::CaptureDevice capDev( new FileCaptureDevice( imageNames));
```

The FRsdk::CaptureDevice instance can then be passed e.g. to the process function of the FRsdk::Enrollment::-Processor.

```
// do the enrollment
proc.process( capDev, feedback);
```

[Previous section] [Next section]

0.24 How to write an ImageBody

[Previous section]

The FRsdk::Image class wraps image data and allows for exchanging data between library and application. F-Rsdk::Images are created of ImageBodies. The code below demonstrates how to write an ImageBody to create an FRsdk::Image of formats different from those supported by FaceVACS-SDK.

The full source code of the example can be found in imagebody.cc.

Walk through step by step

```
#include <frsdk/cptr.h>
#include <frsdk/image.h>
```

This line includes the FRsdk::CountedPtr and FRsdk::Image classes.

FRsdk::Images are handle types created from a FRsdk::ImageBody. More specifically, they are created from a FRsdk::CountedPtr, which holds a reference counted pointer to some instance of a class derived from FRsdk::-ImageBody.

While some particular intensity image formats are supported by FaceVACS-SDK, most are not. In order to use an additional image format, the programmer has to implement support of this format within a class derived from FRsdk::ImageBody. In this example we implement a 'MemoryImageBody'. This ImageBody can be created from bitmap data in some format, which has to be converted to the internal layout required by FRsdk::Image. In the example, it is assumed that the bitmap data has already the appropriate layout, but it is straightforward to modify the function filling the internal bitmap arrays to accomodate a different format.

```
  class MemoryImageBody: public ImageBody
  {
  public:
    // construct memory image from an RGB array.
    // memory will be copied (i.e. not owned)
    MemoryImageBody( const Rgb* b, unsigned int w, unsigned int h,
                     const std::string& name = "");
    ~MemoryImageBody();
    unsigned int height() const  { return h; }
    unsigned int width() const  { return w; }
    const Byte* grayScaleRepresentation() const { return b; }
    const Rgb* colorRepresentation() const  { return rgb; }
```

```
    // extended interface
    std::string name() const {
      return n;
    }

  private:
    void buildRgbRepresentation( const Rgb* rgb_);
    void buildByteRepresentation();
    unsigned int w;
    unsigned int h;
    Byte* b;
    Rgb* rgb;
    std::string n;
  };
```

Here we implement a concrete ImageBody holding both color and byte bitmap data in memory. The name() function returns a string which contains the address of this class instance and the name of the class.

```
MemoryImageBody::buildRgbRepresentation(const Rgb* rgb_)
{
  rgb = new Rgb[w * h * sizeof(Rgb)];
  // this assumes that the rgb array rgb_ points to
  // has the layout required by FRsdk::Images:
  // each pixel is [Blue Green Red NotUsed]; no padding
  //
  // otherwise, copy pixel-wise and rearrange
  // color pixels and/or cut padding

  memcpy(rgb, rgb_, w * h * sizeof(Rgb));
}
```

Function buildRgbRepresentation(const Rgb∗ rgb_) copies the bitmap data passed to the internal Rgb array. This is the point where to apply any modification to support different input bitmap formats, too.

```
MemoryImageBody::buildByteRepresentation()
{
  b = new Byte[w * h ];

  Rgb* colorp = rgb;
  Byte* grayp = b;

  for( unsigned int i = 0; i < h; i++ )
    for( unsigned int k = 0; k < w; k++ ) {

      float f = (float) colorp->r;
      f += (float) colorp->g;
      f += (float) colorp->b;
      f /= 3.0f;

      if( f > 255.0f) f = 255.0f;

      *grayp = (Byte) f;
      colorp++;
      grayp++;
    }
```

Function buildByteRepresentation() fills the byte array by converting the Rgb data previously acquired into 'unsigned char'. Note that in image processing more sophisticated conversions are in use which involve different factors for the individual colors.

Of course you might consider to make the implementation of this function 'lazy', i.e. to defer conversion up to grayScaleRepresentation() is called.

```
MemoryImageBody::~MemoryImageBody()
{
  delete[] rgb;
  delete[] b;
}
```

Upon destruction, don't miss to delete the memory allocated. By means of the CountedPtr you can construct an arbitrary number of images sharing only one instance of a MemoryImageBody. The MemoryImageBody destructor will not be be called unless the last of these images is destroyed.

[Previous section]

0.25 Small Command Line Parser

CmdLine is a simple class only to extend examples with a more flexible but also really simple way for command line options and command line flags.

The implementation is not failure tolerant and has no diagnostics. It is a simple and cheap helper class only.

Member Functions

The constructor takes the arguments argc and argv in the same way as main() does. Both variables will not be modified within class CmdLine.

This function tests if any item of the argv array matches exactly to the given string. (Remember that argv does only contain white-space-stripped items except the items are quoted with quotes.) If a matching item is found function returns true else false.

This function returns a pointer to the argument (next item in argv) of the found search pattern. If no item is found matching the search string a NULL pointer is returned.

0.26 Examples - How to execute

FaceVACS-SDK provides a number of different examples to illustrate its main use cases and its basic functionalities.

All examples are located in the "examples" directory of the FaceVACS-SDK installation folder:

```
- <install root>                the root directory of the FaceVACS-SDK installation
   |
   +-[...]
   |
   +- examples          location for the examples
   |  |
   |  +- cpp                    C++ sources
   |  |  |
   |  |  +- x86_32              executables for 32 bit platforms
   |  |  |
   |  |  +- x86_64              executables for 64 bit platforms
   |  |
   |  +- cs                             C# sources
   |  |  |
   |  |  +- x86_32              executables for 32 bit platforms
   |  |  |
   |  |  +- x86_64              executables for 64 bit platforms
   |  |
   |  +- images                 images used by the examples
   |
   +-[...]
```

Before you execute any example or one of your customized applications, please ensure that a valid SDK configuration file and the following SDK components are available to use:

- the "etc" SDK subdirectory
- either a (possibly customized) FaceVACS-SDK configuration file or a (possibly compiled in) input stream containing a valid license
- the FaceVACS-SDK runtime library and if necessary the Intel IPP/MKL libraries

- other runtime libraries (e.q. if using BioAPI interface or .NET runtime library) Also you have to configure the shared library search path at the target machine. For a detailed description regarding the dependencies and the necessary runtime libraries, please refer to the section Run-Time requirements for FaceVACS-SDK applications

After you have configured your run-time environment, you could execute all provided examples as follows:

1. Open a command line interpreter (Windows: cmd or Linux: console).
2. change to a directory of your choice, e.g. the library directory of your FaceVACS-SDK
3. execute one of the example using the following command line: <YourInstallationfolder>/examples/cpp/x86_-[32/64]/<YourExample> Now the usage of this example is returned and necessary arguments are explained.
4. extend the commandline with necessary arguments and execute this example.

Here is a whole commandline for the enrollment example "enroll.exe": <YourInstallationfolder>/examples/cpp/x86-_32/enroll <YourInstallationfolder>/etc/frsdk.cfg result.fir sample.jpg

Please note:

- all examples expect at least the configuration file (default: frsdk.cfg)
- the description of the necessary arguments are also part of the source code of each example. Please refer to the function int usage() of every C++ and/or C# example source file, e.g. enroll.cc.

The FaceVACS SDK documentation contains a list of the examples and the formatted source code providing hyperlinks to relevant parts of the documentation. Some of the examples are described more detailed in the section Tutorial.

0.27 FaceVACS-SDK Reference - Overview

FaceVACS-SDK API reference documentation can be browsed using the following entry points (see also in header of each page):

```
- @htmlonly <A HREF="namespaces.html">Namespaces:</A>\endhtmlonly
```

Namespaces: refers to all subnamespaces of FRsdk

- Namespace Members: alphabetical list of all typedefs, functions and classes, which are not part of a subnamespace
- Classes Alphabetical: alphabetical list off all classes
- Compound all classes,unions and structs of FRsdk and subnamespaces
- Compound Members: all nested classes, members and type definitions, which are part of any class of FRsdk
- Files: all header files distributed with FaceVACS-SDK
- File Member: global type definitions, functions and preprocessor defines, which are not part of the former references

0.28 FaceVACS-SDK FAQ

- Problems with Getting Started
- Licensing Problems and Questions

- Compiling and Building Applications
- Executing Applications
- Biometric Performance, Features and Problems
- 3D Algorithms
- Programming Issues
- Performance (Computing Speed) Questions
- Current and future SDK Programming Features

0.28.1 Problems with Getting Started

- License mismatch errors
- Problems running example programs: ∗.dat files not found
- Problems running example programs: DLLs / shared libraries not found

Q:

I've transferred the license to the configuration file, but I still get error messages about 'License mismatch'.

A:

Please check

- that the hardware key displayed in the license dialog of the Configuration Editor is still the same you provided when requesting the license
- that the hardware key provided to obtain the license has actually been created with the license dialog of the Configuration Editor of the SDK version you are using
- that the SDK version you indicated when requesting the license is the one you are actually using
- that both the import library/shared library used to link your application and the DLL / shared library used by your application at run-time are of the same version as the configuration file passed to your program. Users frequently encounter the 'license mismatch' error if they have previous versions of FaceVACS S-DK installed and missed to update either their project (linker) settings or the run-time DLL/shared library search path of their applications.

Q:

When trying to run one of the example programs, I get an error message like "Face finder : LdDiscriminator: error opening template file '/lddisc.dat'." or "Error Processing : Comparison module : PCLD: File '/gpcld.dat' could not be opened for reading".

A:

This message indicates that some algorithmic template file could not be loaded. The locations of these template files are defined in the configuration file etc/frsdk.cfg, namely within lines containing the string 'DataDir'. On MS Windows systems these locations are properly set after installation. Most often the reason that this error occurs anyway is that another file (for example, the license file) is passed as the first command line argument to the example program. On Linux systems, however, the locations are not preconfigured and have to be entered into the configuration file as described in Installation Settings.

Q:

Example applications fail to start complaining about missing DLL's (Win32) or shared libraries (Linux). Messages are

Win32: "This application has failed to start because XXX.dll was not found. Reinstalling this application may fix this problem"

Linux: error while loading shared libraries: libfrsdk.so: cannot load shared object file: No such file or directory

A (Win32):

This error message indicates that the respective DLL is not in the DLL search path of your application. The DLL 'libfrsdk.dll' is installed to the subdirectories lib/msc_X.X-ipp_crtdll(-g). The DLL libiomp5md.dll is installed in the subdirectory "lib/share". msvc*.dll' are provided as redistributable packages from MS or they are located in bin/x86_[32/64]/msc_X.X-ipp_crtdll/vcredist-[x86/x64].exe

How to fix:

See subsection Run-Time Environment for FaceVACS-SDK Applications for information on how to properly arrange the run-time environment. Avoid copying any DLL's into the system32 directory. This can cause trouble in case of installing a different SDK version in parallel or afterwards.

A (Linux):

See subsection Run-Time Environment for FaceVACS-SDK Applications for information how to properly set the LD_LIBRARY_PATH variable to fix this problem.

0.28.2 Licensing Problems and Questions

- When does a license become invalid?
- How can I check license validity in advance to prevent getting license exception

Q:

How does license protection work and under what circumstances will licenses become invalid?

A:

Prior to run any FaceVACS-SDK application, a valid license has to be transferred to its configuration file. A license contains settings for

- a mandatory license string containing some ID of the vendor distributing the application
- an optional 'hardware key' of the system the application is supposed to run on
- an optional expiry date of the license
- enabling and disabling of particular algorithmic features of the SDK.

The validity of these settings is ensured by a digital signature embodied in the license key.

The aforementioned 'hardware key' is actually a host ID, which is obtained in different ways on Win32 and Linux systems:

On **Win32** systems, the host ID is derived from the computer SID, a unique identifier created upon operating system installation. If you for any reason reinstall your OS a new, i.e. different computer SID will be generated, hence the host ID changes, rendering your license invalid.

On **Linux** systems, the value returned by the gethostid() system call is used to get the host ID. See *hostid (1)* for a description of how to set a Linux host id. If you did not yet before set a persistent host id with the 'hostid' command on your Linux computer (which can be seen by checking whether /etc/hostid exists), a default host id is returned, which is subject to changes, e.g when the IP address changes. So it's a good idea to set a persistent hostid before creating the hardware key file required for FaceVACS-SDK license generation.

A license can be invalid or valid. A valid license can still restrain the set of usable algorithmic features.

A license will be invalid if

- a hardware key was licensed which does not match the hardware key of the system the application is run on
- an expiry date has been licensed which has passed
- either the license key or one or more of the protected license settings have been changed within the configuration file.

An algorithmic feature is disabled if it has not been explicitely enabled with the license.

Q:

How can I check license validity in advance to prevent getting runtime license exceptions?

A:

License invalidity as defined above can be detected by trying to instantiate a FRsdk::Configuration object within a try-catch block catching license exceptions. Doing this at program start allows for graceful handling of license invalidity.

Enabling or disabling of particular features can be checked by parsing the list of key-value pairs returned by FRsdk::Configuration::protectedItems(). The licensed features are enumerated with keys starting with FRSDK.-LicenseSettings.... Use only features whose enabling is confirmed by the corresponding LicenseSettings entry. See subsection FaceVACS-SDK Feature enabling/disabling, Limits for details.

0.28.3 Compiling and Building Applications

- C1083/LNK1181 errors when building Visual C++ project
- Complains on undefined references when linking C++ SDK program
- Can you provide template project files for MS Visual Studio?
- Linux gcc: Error 'Cannot exec cpp0'
- Subtile bugs using stl iostream libraries

Q:

When I try to link the SDK based program I get multiple complains about undefined references

A:

Make sure that all the linker knows about all libraries required and the directories where they can be found. Use examples/Makefile (Linux) or the Win32 project files for references how to do this or for Win32 see next question.

Q:

When I try to build my FaceVACS-SDK based Visual C++ project, I get error messages like *C1083: cannot open include file 'frsdk/init.h'* or *LNK1181: cannot open input file "frsdk.lib"*

A:

You have to add the FaceVACS-SDK include and library directories to your project settings. Read the User Guide section Build Environment for how to do this. Use the project files provided with the SDK examples for reference.

Q:

Can you provide template project files for MS Visual Studio?"

A:

There are project files for the sample programs in the examples subdirectory, which you can use as a starting point.

Q:

I try to compile the examples on Linux and get the error message: g++: installation problem, cannot exec 'cpp0':

A:

This will occur if you are asking your gcc by providing the "-V" switch to run compilation with a compiler version not installed on your system. It is not a built-in feature of gcc to emulate previous versions.

Q:

I get subtile bugs using iostream STL libraries

A:

There are old (pre-STL) and new versions of the iostream libraries. Unfortunately, their syntax is partially the same, so you can get subtle bugs when including wrong headers and/or not fully specifying the std namespace for the STL version. Generally, they must not be intermixed within a compilation unit.

To avoid this sort of problems, carefully check your include directives to not include

```
#include <fstream.h>
#include <iostream.h>
#include <strstrea.h>
```

etc.

Solution 1a : Instead, use the versions without suffix, where required:

```
#include <fstream>
#include <iostream>
#include <strstream>
```

Solution 1b : Fully specify the std namespace of the classes and constants, e.g.:

```
std::ifstream firIn( "C:\\temp\\enroll.FIR", std::ios::in | std::ios::binary);
```

or:

```
using namespace std;
```

0.28.4 Errors on Executing FaceVACS-SDK based Applications

- Runtime Error R6034
- System.IO.FileNotFoundException
- System.IO.FileLoadException
- 'Access Violation' error
- EndOfData exception reported by debugger
- Strange error messages referring to 'Waterfall Procedure'

Q:

When trying to start my .NET based application, I get a dialog indicating some "Runtime Error R6034".

A:

This indicates a missing manifest file for your application. Please provide a manifest file. For libraries related to Cognitec stuff, a manifest is embedded into the .NET DLLs as explained in u_netapps.

Q:

When trying to start my .NET based application, I get an "System.IO.FileNotFoundException" with the message "Could not load file or assembly 'libfrsdknet'", though libfrsdknet.dll is in the DLL search path.

A:

The .NET library DLL libfrsdk(d).dll has to reside in the same directory as the application binary, the usual DLL search algorithm does not apply.

Q:

When trying to start my .NET based application, I get an "System.IO.FileLoadException" with the message "A procedure imported by 'libfrsdknet, Version=5.0.2.0, Culture=neutral, PublicKeyToken=b9feb4b24e2565f5' could not be loaded".

A:

Explanation:

The reason for this is that not only 'libfrsdknet.dll', but 'libfrsdk.dll' is required as well for executing a .NET application built with FaceVACS-SDK. 'libfrsdknet.dll' is just a .NET wrapper, the implementations reside in 'libfrsdk.dll'.

Solution A:

Which libraries are required to run different types of applications, is explained in the User Guide, section Run-Time requirements for FaceVACS-SDK applications. To fix the problem, you might for example copy 'libfrsdk.dll' into your bin\Debug directory, too.

Solution B:

It seems that all required files are located in the working directory and/or in the Windows search path, but you still receive one of the error messages listed above. Please try the following:

1. check that you are using the desired version of the required dlls:
   - for your current FaceVACS-SDK version
   - for the compiler you are using
2. check if the following files are available

- msvcpXX.dll and msvcrXX.dll Please replace "XX" with the number of your compiler. For example msvcr100.dll if you use MS Visual Studio 2010 (VC++ 10).

In doubt use some tool like Dependency Walker (http://www.dependencywalker.com) to determine the required files and the path of the DLL actually used. See also the FaceVACS-SDK User Guide, section "Run-Time Environment for FaceVACS-SDK Applications", for a description of how Windows searches for DLL's.

Q:

When running our application using FaceVACS-SDK, we get access violations or when running the application in the debugger, user breakpoints with messages like "Invalid Address specified to RtlValidateHeap" and that like.

A:

Ensure, that all instances of the code share the same instance of the MS Visual C run-time library. Since the release and debug DLL's of the SDK are linked with the multithreaded DLL and the multithreaded debug DLL version of the MS Visual C run-time library, respectively, ensure that your project settings for the release and the debug version of your application are set accordingly as described in Build Environment.

If you are using MFC, note that there is a similar relation between the MFC settings and the C run-time library used. Selecting use of the static MFC library (in Projects->Settings->General->Microsoft Foundation Classes) will result in linking with the static C run-time library, independent off any settings applied in "Use Run-time libraries".

Be careful when changing MFC settings in Visual Studio since changing them can "automagically" cause the "Use Run-time library" setting to be changed. So, first change the MFC settings, then check or reset the correct "Use Run-time library" setting.

Moreover, you have to ensure, that the application uses the appropriate FaceVACS-SDK DLL. Remember the search order for DLL's as described in Run-Time Environment for FaceVACS-SDK Applications.

Q:

An exception ::Partitionizer::EndOfData is thrown when we attempt 1-to-N matching based on a population. We get this exception when calling either FRsdk::FacialMatchingEngine::compare() or FRsdk::FacialMatching-Engine::bestMatches() with the population as a parameter.

A:

In the internal population comparison framework there is a condition which depending on the execution context can be either regular or irregular. This is the reason why this condition is handled internally by exception throwing even if it finally does not constitute an error condition. Since this exception is caught and handled internally, it will never be delivered to user code, so it is entirely safe to ignore it. The debugger nevertheless observes and reports the occurence of this exception.

Q:

When starting the program, I get a strange error message referring to something like a 'Waterfall procedure':

A (Win32):

The message is like "No DLL was found in the waterfall procedure, failed to initialize".

Reason:

Missing ipp20 subdirectory in lib directory

How to fix:

Reinstall FaceVACS-SDK (be sure to save licensed frsdk.cfg file before) or copy the missing ipp20 subdirectory from elsewhere.

When deploying SDK applications be sure to copy the entire directory hierarchy of the library directories.

Another possible reason:

Another software package has installed IPP libraries of a different version

How to fix:

Try to put all DLL's relevant to your application along with the application binary into the same directory. In this case the appropriate IPP libraries will be first in the DLL search path.

A (Linux):

The message is like "No shared libraries was found in the Waterfall procedure".

Possible Reasons are that some of the IPP libraries are missing or that a wrong version of GCC is installled on your system.

How to fix:

Re-install the SDK or copy IPP libraries from an existing valid installation or install the version of the GCC recommended in the SDK documentation.

0.28.5 Current and Future SDK Programming Features

- Is there any SQL DB support in FaceVACS-SDK?
- Are there any widgets to display video images with SDK?
- What is the status of thread safety of the SDK?
- How can I access the native jpeg format in FRsdk::JpegImage?
- Are there any interfaces for image processing operations like scaling, cropping etc.?
- Can we use the FaceVACS-SDK from MS Visual Basic or Borland Delphi?

Q:

Is there any SQL DB support in FaceVACS-SDK? How can I load/store FIR's or images from/into a SQL database with the SDK?

A:

SQL support is not part of FaceVACS-SDK. Only some of the so called "evaluation tools" allow for using SQL databases, but there aren't any programming interfaces providing this functionality.

Q:

(How) can I display video images with the SDK?

A:

No. The FaceVACS-SDK provides core face recognition technology only. The "application framework", i.e. graphical user interfaces, interprocess communication and database access is not part of the Cognitec SDK product and has to be developed by the SDK customers according to their requirements.

However, there is some Win32 sample code available which can give users who are not familiar with image display some idea how to implement this. The sample code is provided as is and will be made available on request.

Q:

What is the status of thread safety of the FaceVACS-SDK?

A:

See FaceVACS-SDK and Multithreading .

Q:

How can I access the native jpeg format in FRsdk::JpegImage to effectively transfer images across processes?

A:

An access to the JPEG encoded byte stream is not available.

Q:

Are there any interfaces in the FaceVACS-SDK for image processing operations, for example scaling, cropping etc.?

A:

Though we use these operations internally, interfaces of these are not provided by the FaceVACS-SDK. There are public libraries or inexpensive, highly optimized vendor libraries available (e.g. with Intel IPP) which provide a lot of this stuff.

Q:

Can we use the FaceVACS-SDK from MS Visual Basic or Borland Delphi?

A:

Yes, the FaceVACS-SDK .NET provides an assembly which can be used from MS Visual Basic or Delphi 2005.

0.28.6 Biometric Performance, Features and Problems

- Independent vendor test reports
- Reasons for failures upon FIR creation from image
- Can we use images containing partial faces only?
- FIR self comparison yields score below 1.0
- Biometric performance for face in the crowd Applications
- Is it advantageous to use color instead of b/w images?

Q:

Can you provide any documentation of independent tests comparing Cognitec's performance with that of other vendors?

A:

The best documentation of this kind available is that of the Face Recognition Vendor Test (FRVT) run 2002 by US NIST. See `http://www.frvt.org` for more information and publicly available reports.

Q:

What are the reasons that FIR's cannot be created of some face images?

A:

Possible reasons:

1. Eye distances in the image are bigger/smaller than configured

There are algorithmic parameters indicating which minimum and maximum eye distances of faces are to be expected within the images. Reducing the range of eye distances searched (especially from below) can considerably increase performance, but too tight settings can faces prevent from being found, which can cause enrollments to fail. The values for the minimum and maximum eye distances have defaults of relative 0.1 and 0.4 pixels (relative means to image width), respectively. See User Guide, section FaceVACS-SDK application configuration for how to change these values. Note that the settings within the configuration file apply to the face finders internally used by Enrollment/Identification/Verification Processors only. The FRsdk::Face::Finder takes these settings at run-time arguments of its find() function.

For good biometric performance, the minimum eye distance in the face images should be about 64 pixel. Less pixel in eye distance may decreases biometric performance.

2. The images do not meet the requirements for face images

Face image requirements:

- The complete face area has to be contained within the image. Take at least double of the eye distance for the width and the height of the image, better extended by a margin.
- The face has to be entirely visible. Partial covering of the face can cause the face or eye finders to fail.
- Head tilt and pan should not exceed 15 degrees. This is the range the face and eye finders are trained for. Even if they frequently find face and eyes outside these ranges, too, comparison performance will start to degrade then.

- The face has to be upright within the image, i.e. the line connecting the eye positions has to be approximately horizontal.

Q:

Is it possible to use images with partially visible faces with the SDK ?

A:

Generally, this is not recommended. More specific, the answer is different for face/eye finders and comparison. Just providing the images as they are to automatic face and eye finding will most probably cause face and eye finding problems.

What you can do is to determine eye positions manually and providing images along with eye positions to the Enrollment::Processor::process() functions taking annotated images. Then FIR creation will be possible. But biometric performance still can be bad, depending on the type of image corruption.

Q:

When I compare a FIR with itself, I do not get a value of *1.0*. Rather, I get a value of *0.xy*.

A:

This is due to a special feature called isotropization, which allows for more uniform score values across different types of faces and face image qualities at the cost of unexpected self-scores below 1.0.

Q:

In the documentation it is mentioned that the lighting conditions for the SDK should be similar between enrollment and verification and that sunlight should be avoided. In this case how do we identify a person from the public crowd ? For instance in an outdoor stadium with a football match going on, how would we identify a person ?

A:

In circumstances where you can control the lighting conditions, you should try to meet these requirements. But we do not absolutely exclude sunlight for usage with face recognition, neither do we state that identification will not work with different lighting conditions between search and enrollment images. Of course with sunlight it is quite difficult to have similar lighting conditions between enrollment and identification. Moreover, sunlight tends to produce varying regions with both intense lighting and deep shadows, which is quite challenging for face recognition and will normally produce poor results.

So the identification task with arbitrary lighting conditions on the search image is a very hard one with the current state of technology. But there are nevertheless chances to get a person identified when the enrollment images come from a similar lighting situation and show the same pose of the person's face.

Q:

Is it advantageous to use color instead of b/w images?

A:

No. Internally, images are converted to gray scale, since the algorithms have been trained with gray scale images.

0.28.7 3D Algorithms

Q: Can I use the 3D algorithms to improve performance of access control applications?

A:

Using 3D information will certainly enhance the accuracy of face recognition, but with the sensors availabe today applicability is quite limited. Many sensors available require lengthy exposure times to produce shape data smooth enough to be used by the algorithms. Moreover, long exposures create additional shape artefacts. Sensors using principles not requiring long exposure times, like stereo cameras, often do not provide sufficiently smooth shape data. In addition, most sensors are quite expensive and, since they demand special lighting to be applied, the effort can be better spent into building customized lighting arrangements for 2D face recognition.

Bearing in mind that 3D algorithms will increase biometric performance of face recognition and expand its application fields, Cognitec decided to make the 3D algorithms available anyway to customers which want to investigate this technology in its early stage already.

Q: Which data is required by the 3D algorithms?

A:

The 3D versions of the current algorithm require pairs of intensity and shape images for input.

These image pairs have to meet certain alignment requirements. Specifically, they have

- to be taken in approximately the same instant
- have to have the same width and hight
- have to be aligned on a pixel basis, i.e. a pixel at location (i,k) of the intensity image has to represent the intensity value of the corresponding vertex found at location (i,k) of the shape image.

Q: Can I compare facial shape data with intensity image data?

A:

The current algorithm requires pairs of intensity and shape images as described above for **both** enrollment and verification/identification, i.e. you cannot compare shape images with intensity images only.

Q: Can you recommend some 3D sensors suitable for face recognition?

A:

There are several 3D sensors available most of which have the limitations described above. Currently Cognitec cannot recommend any 3D sensor suitable for real-world applications. Cognitec continues to evaluate sensors and will make recommendations if some sensors appropriate for face recognition applications could have been found.

Q: Is it possible to create shape information from 2 or more intensity images?

A:

In theory, this is actually possible to create a 3D model from 2 or more intensity images displaying multiple views of the face, but retrieving the depth map from one or more different 2D views is quite a tough problem, especially if a dense mapping is required. Some current solutions from vendors which are specialized to produce calibrated stereo cameras along with depth map evaluation software do not yet produce shape data of sufficient quality for shape based face recognition. However, since these products are continuously improved, suitable solutions might be available at some time.

0.28.8 Programming Issues

- How do I create a FIR out of an Image?
- Isn't using feedback classes for enrollment too cumbersome?
- Errors when reading FIR's

Q:

How do I create a FIR out of an image?

A:

Use one of the process() interfaces of the FRsdk::Enrollment::Processor. See examples/enroll.cpp for sample code.

Q:

The enrollment procedure using feedback classes seems quite cumbersome. Is there a straight way to create a FIR?

A:

That's the only way to do it. Using feedback classes ensures that you get sufficient diagnostic information in case of FIR creation failures. There is no performance decrease due to the feedback concept.

Q:

When I try to read a FIR previously written by a SDK based program I get exceptions with error messages like "incompatible data" or "creating FNPackage from memory...".

A:

There may be several reasons. The most common error occurs when one tries to read a FIR created by a previous version of FaceVACS-SDK using a different comparison algorithm. This will result in an error message *PCLD: loadFeatureSet: incompatible data: wrong version.*

Each time the comparison algorithm changes FIR's have to be recreated, since they are specific to the algorithm used. The 'Readme' accompanying the SDK will state whether the FIR's are compatible or not.

Other exception error messages like *creating FNPackage from memory: reading sizes of cores* indicate, that you mixed interfaces for platform dependent and platform independent storage and retrieval. Interfaces of these groups must not be confused:

Platform dependent storage and construction:

```
FRsdk::FIR::writeTo(Byte *& buf )) const
FIR FRsdk::FIRBuilder build(Byte *& buf);
```

Platform independent storage and construction:

```
FRsdk::FIR::serialize( std::ostream& ) const;
FRsdk::FIR::serialize( Byte* p) const;
FIR FRsdk::FIRBuilder build( std::istream&);
FIR FRsdk::FIRBuilder build( const Byte* p, unsigned int len);
```

0.28.9 Performance (Computing Speed) Questions

- Original 'enroll' sample program too slow?
- My own FaceVACS-SDK based program runs slow

Q:

I run your 'enroll' sample program and detected, that it took more than 3 sec to create a FIR from a JPEG image. Is FaceVACS really that slow?

A:

1. Generally, when complaining about performance issues, please never forget to indicate type and clock of processor and the size of images used to make your figures comparable. Bigger images will require more time to be processed, and faster processors it will take less time to do their job.

1. The seemingly slow performance of the enroll sample when invoked just one image results from the SDK initialization time, which is part of the 3 sec mentioned. To get realistic figures on FaceVACS performance modify the sample program to make 10 or more FIR creations and start time measurement after the Enrollment::Processor object was constructed. Normally, FIR creation on a P4 2GHz should take not more than 0.5 sec for an image of approx. 250,000 pixel.

1. A table with performance figures for main FaceVACS algorithms on different platforms is available on request.

Q:

In my program I measure just the time needed for enrollment, which is nevertheless considerably more than you describe.

A:

Please check the performance of your PC by running the original 'enroll' sample program with about 10 images. This should be done within approx. 10 sec or less on a P4 machine. If this is the case, please check whether you are perhaps profiling a debug build of your program (linked against libfrsdkd.lib). The debug version of the FaceVACS-SDK library is inherently slow.

0.29 whatsnew

What is new in FaceVACS-SDK

Version 8.9.5

- Fixed some bugs in A15 face comparison algorithms which may cause segfaults
- A15-1 face comparison algorithm data was made more compressible and therefore better suitable for mobile platforms.

Version 8.9.4

- Upgraded A15-0 face comparison algorithm to A15-1. A15-1 has a lower memory requirements than A15-0, but it has also a lower biometrical performance.
- Added support for Intel platforms in Android version of the SDK.
- Extended Android version of the SDK with a C++ programming API.
- Removed D1 face comparison algorithm, which was intended for a special project only.

Version 8.9.2

- fixed the spelling of the ComplianceThresholds class in Java bindings.
- extended the API of FullFrontalExtractor Java class.
- Dropped the support for Visual Studio 2005 and 2008

Version 8.9.1

- added conversion between SDK images and Android and Desktop Java image classes
- Extended the Java language binding to support complete C++ API
- Added support for Visual Studio 2012 and 2013

Version 8.9.0

- replaced comparison algorithms B7 with B8 and A14 with A15. The B8 comparison algorithm is now the default one.
- camera control is available for abstract capture devices
- on Windows platform an optional static libraries SDK is available
- Added gcc 4.6 to and removed gcc 4.1 from supported compilers list

Version 8.8.0

- added support for Android OS 2.3 and higher on ARM devices.
- no support for Java bindings on MacOS in this version.

Fixes:

- fix MSVC 8.0 (VS2005) runtime redistributable dependency

Version 8.7.0

- added a Java language binding.
- no support for Java bindings on MacOS in this version.

Version 8.6.0

- added B7 comparison algorithm which is now the default one.
- for Linux platforms upgraded the internally used Intel Performance Libraries IPP and MKL to versions 7.1 and 11.0.0 respectively.

Version 8.5.0

- replaced comparison algorithms B5 with B6 and B5L5 with B6L5
- removed support for Windows Mobile platform (EMSC 8.0)
- removed FrameGrabbers / UrlFrameGrabbers / UEyeCameras interfaces
- introduced new, consolidated capture device interface (CaptureDevices) and added additional device types (Files, HTTP)
- introduced new capture device control interface (CameraControllers)
- SDK examples were made more flexible for biometric evaluation usage

Version 8.4.1

- renaming of contribution program Advertisment to Anonymous Video Analytics (ava.exe)
- additional command line parameters for provided examples available

Fixes:

- xmlcompmatch: empty FIRs does no longer cancel processing
- contribution program ava.exe starts now also on Windows 7

Version 8.4

- on MS Windows platforms: support of MS Visual Studio 2010
- on Mac OS X: support of XCode 3.2.1 (gcc 4.0 and gcc4.2)
- replaced B4 with B5 and B4L5 with B5L5
- scaling is now an optional part of image extraction for Token Face and Full Frontal
- classes SampleQuality and SampeEvaluator are removed and also the related feedback member function of the biometric processors (Enrollment, Verification, Identification). The classes Portrait::Characteristics, Portrait::Analyzer, Portrait::FullFrontal::Test Portrait::Feature::Test are more flexible and could be more adapted to customer requirements if image quality has to be classified.
- support of UEye on 64 Bit platforms
- on Linux: removed dependencies to system png-libraries
- switched to Qt 4.7 for GUI applications and tools delivered with FaceVACS SDK
- improved sharpness test of portrait characteristics

- contribute extended demonstration of portrait characteristics: FvAdvertisement

Fixes:

- face finder takes care about search box in parameter list
- improved cooperation with thread

If you are upgrading from an older version see previous versions' changes

0.30 oldrevs

Changes in previous versions of FaceVACS-SDK

Version 8.3.2

- added age estimator to portrait characteristics

Version 8.3.1

- Support for gcc-4.0.1 on MacOS X (10.5, 10.6) on x86 platforms

Version 8.3.0

- support for MacOS 10.5, 10.6
- Support for OMAP3 (TI Multimedia Processors Family 3) on Linux
- improved IDS UEYE camera support: now uses driver version 3.50
- fixed TIFF support for linux gcc-4.3
- improved T8 face finder performance
- Configuration Editor: fixed license dialog refreshing

Version 8.2.1

- supports Windows 7 and Vista now
- new installer
- uninstall is now performed by using windows standard mechanism: "control panel"-> "program and feature"

Fixes:

- on Linux platforms hwkey do not depends any longer from libodbc.1.so

Version 8.2.0

- New matching algorithm A14 replaces A13

Version 8.1.0

- new support for Windows Ce 5.0 on x86 compatible processors like Geode GX
- using new IPP version and reduced shared library dependencies to IPP and MKL Libraries

Version 8.0.1

- support of IDS uEye cameras including dynamic control of gain and exposure time adapting to current lighting conditions (in Face Area)
- replaced 3D algorithm B4L5 with B5L5
- added some examples for uEye and life face tracking

Version 8.0.0

- new support for Windows Mobile 6.2 an ArmV4I
- new compiler support gcc 4.3 on Linux
- removed compiler support gcc 3.3.3 on Linux
- improvements and interface changes in chin, crown and ear position detection
- new Face finder T8
- new SDK interfaces to activate license
- replaced B4 with new and improved B5 comparison algorithms

Version 7.1.0

- new 3D algorithm L4C1
- more precise 2D comparison algorithm C1 compared to B3 and A13.
- enhanced ISO FullFrontal Compliance test (FRsdk::ISO_19794_5::FullFrontal::Compliance) in a way to separate the minimal requirements and the optional requirements (best practice)
- extended portrait characteristics (FRsdk::Portrait::Characteristics) and feature test (FRsdk::Portrait::Feature::Set) with two new age classifications for 26 years and 36 years:
  - isBelow26()
  - isBelow36()
- speed and performance improvements in calculation of portrait characteristics:
  - background uniformity
  - hot spots/reflections
  - grayscale density

Version 7.0.0

- New face finding algorithm T8 with significantly increased detection performance and speed
- Extended Portrait::Characteristics measurements and ISO 197940-5 full frontal tests (FRsdk::Portrait::Characteristics and FRsdk::ISO_19794_5::FullFrontal::Compliance):
  - Gaze away detection, see FRsdk::ISO_19794_5::FullFrontal::Compliance::eyesGazeFrontal
  - Detection of hot spots (reflections), see FRsdk::ISO_19794_5::FullFrontal::Compliance::noHotSpots
- FaceVACS-SDK now supports 64bit Windows on x86_64 platforms

Version 6.4.0

- Extended Portrait::Characteristics measurements:
  - added detection of none uniform background, see FRsdk::Portrait::Characteristics::background-Uniformity, FRsdk::ISO_19794_5::FullFrontal::Compliance::backgroundUniformity,
  - added red eye detection, see FRsdk::Portrait::Characteristics::redXEye, FRsdk::ISO_19794_5::Full-Frontal::Compliance::eyesNotRed
  - added tinted glasses detection, see FRsdk::Portrait::Characteristics::eyeXTinted FRsdk::Portrait::-Characteristics::Feature::Set::tintedGlasses
  - added etnicity detection, see FRsdk::Portrait::Characteristics::ethnicity, FRsdk::Portrait::Characteristics-::Feature::Set::ethnicity -improved uniform lighting detection
- revised FRsdk::Face::Tracker tracking interface
- incorporated version name into library name in order to get name compatibility if interfaces are compatible of different SDKs

Version 6.3.0

- Extended Portrait::Characteristics with detection of natural colors, and detection of gender
- added feedback to get extended image format details on image loading
- added face tracking facility
- added load function for convenient image loading.

Version 6.2.1

- Added support for loading bitmap images from input streams

Version 6.2.0

- The BioAPI interface now supports BioAPI version 2.0 (ISO 19784:1:2006)
- FaceVACS-SDK for Linux now supports x86_64 platforms

Version 6.1

- introduced A13 comparison algorithm, replaces A12
- added support for Windows Vista

Version 6.0

- New FaceVACS B3 matching algorithm for higher biometric performance
- Support for adjusting images produced by fish eye lenses
- Support for vignetting of images

Version 5.0

- New FaceVACS B2 and A12 matching algorithm for higher biometric performance
- Added support for compilers: GCC 4.0.2 and Visual Studio 2005
- New Portrait::Characteristics measures and tests
- more robust face and eye finder (version T7)
- Portrait characteristics: added mouth open detector and sharpness measure
- new persistence format of FIRs (B2: FIR version 5.3, A12: FIR version 5.4)

Version 4.1

- New FaceVACS B1 matching algorithm for higher biometric performance in addition to the A11 matching algorithm

Version 4.0

- FaceVACS-SDK supports 3D (shape) data
- Improved algorithms for face and eyes finding T5
- Extended Portrait::Characteristics measures
- ISO 19794-5 compliance test

Version 3.1

- added `BioAPI` compliant interface
- bug fixes, documentation improvements
- changed EyesFinder::Location - renamed left to first and right to second to avoid confusions dealing with the "real" face orientation, for details see FRsdk::Eyes::Location
- added support for construct a FRsdk::Configuration object from input stream - this allows for "built in" configuration/license
- exception handling: renamed LicenseViolation exception to FRsdk::LicenseSignatureMismatch, added FRsdk::FeatureDisabled and FRsdk::LimitExceeded exceptions for more fine-grained exception handling
- added support for PNG (Portable Network Graphics) image I/O (FRsdk::Png)

Version 3.0

- made most of FaveVACS-SDK interfaces MT-safe,
- improved documentation according to MT-safe aspect
- added support for new compiler for linux: gcc 3.3.3
- added function for storing jpeg images with limited size
- added support for reading and writing ISO 19794-5 compliant CBEFF compliant Token Face Images <frsdk/cbeff.h>
- redesign of some FaceVACS-SDK interfaces for improved flexibility
- added `.NET` interface to FaceVACS-SDK (WIN32 only)

0.31 Namespace Index

0.31.1 Namespace List

Here is a list of all namespaces with brief descriptions:

0.32 Hierarchical Index

0.32.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

0.33 Class Index

0.33.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

0.34 File Index

0.34.1 File List

Here is a list of all files with brief descriptions:

0.35 Namespace Documentation

0.35.1 FRsdk Namespace Reference

The global name space for the SDK.

Namespaces

- Bmp

  Bitmap (BMP) image implementation.
- Enrollment

  the namespace for the enrollment facility
- Eyes

  the name space for the eyes finding facility
- Face

  the name space for the face finding facility
- Identification

  the namespace for the identification facility
- ImageIO
- ISO_19794_5

  Support for image formats defined by ISO/IEC 19794-5:2005.
- Jpeg

  JPEG Image support.
- Jpeg2000

  JPEG 2000 Image support.
- Pgm

  PGM/PPM image format support.

- Png

  PNG (Portable Network Graphics) image format support.
- Portrait

  Portrait characteristics and feature tests.
- Tools

  Misc tools.
- Verification

  the namespace for the verification facility

Classes

- class VideoFormat

  Opaque type representing a video format (resolution, bits per pixel, etc).
- class CaptureDeviceBody

  Abstract capture device body.
- class CaptureDevice

  Capture Device handle (interface)
- class LicenseSignatureMismatch

  License signature mismatch.
- class FeatureDisabled

  An object of this type is thrown at any time if requesting or accessing a disabled FaceVACS-SDK feature.
- class LimitExceeded

  An object of this type is thrown at any time if a configured limit of FaceVACS-SDK is exceeded.
- class Configuration

  Configuration object of the FaceVACS SDK library.
- class CountedPtr

  This class template implements the reference counting idiom simulating name semantics (see Scott Meyer's Bible, James Coplien's book).
- class FIR

  FIR - Facial Identification Record.
- class FIRBuilder

  Building FIRs from serialized representations Use Enrollment::Processor to build FIRs from primary biometric data (face images).
- class Population

  An ordered (in the order of additions by add()) set of named FIR's which represents the population used for identifications.
- class ImageBody

  Abstract image body.
- class Image

  explicit template instantiation for win32
- class LenseDistortionCorrector

  A class providing radial lense distortion correction.
- class FacialMatchingEngine

  Low level facial comparison facility.
- class Position

  explicit template instantiation for win32
- class Box

  The class Box describes a rectangular box in discrete 2D coordinates defined by 2 opposite points, origin and end, where origin.x <= end.x and origin.y <= end.y.
- class Sample

  Data sample containing mandatory intensity image and optional shape data and eye positions.

- class Score

  This class represents a score for representing the comparison result between a FIR and the biometric evidence.
- class ScoreMappings

  FAR,FRR / Score mappings.
- struct Vertex
- class ShapeImageBody

  Abstract shape image body.
- class ShapeImage

  explicit template instantiation for win32
- class PointSetBody
- class PointSet
- struct Rgb

  Red, green and blue color model.

Typedefs

- typedef void ∗ RefCountHandle
- typedef std::pair< Image,
  Eyes::Location > AnnotatedImage

  An image with annotated eye positions.
- typedef std::list< AnnotatedImage > AnnotatedImageSet

  A set of annotated images.
- typedef std::list< Score > Scores

  an ordered collection of score values
- typedef std::pair< std::string,
  Score > Match

  a named score for a match of two FIR's
- typedef std::list< Match > Matches

  the container used for a set of Matches.
- typedef std::list< Image > ImageSet

  container used for collection of images within FaceVACS-SDK
- typedef std::list< Sample > SampleSet

  container used for collection of Samples within FaceVACS-SDK
- typedef unsigned char Byte

  A 8 bit byte representation.

Enumerations

- enum ImageRotation { ROTATECLOCKWISE, ROTATECOUNTERCLOCKWISE, ROTATEUPSIDEDOWN }
- enum SmoothingFunction { Fixed, Linear, Gaussian }

  There are threee ways to blend the margin to the target color: Gaussian blends intensisty of each color channel with a half gaussian function, Linear blend is propotional to the border distance and Fixed sets all pixel of margin to the border color.

Functions

- CaptureDevice createCaptureDevice (const Configuration &, const std::string &Name)

  factory function for creating capture devices configured under FRSDK.CaptureDevices.Name using builtin capture device types a Camera Controller will be used if configured under FRSDK.CameraControllers.Name
- CaptureDevice createCaptureDevice (const CountedPtr< CaptureDeviceBody > &, const Configuration &, const std::string &camControlName)

factory function for creating user defined capture devices using a Camera Controller configured under FRSDK.-CameraControllers.Name

- RefCountHandle newRefcount (int initialValue)
- void deleteRefcount (RefCountHandle)
- void incRefcount (RefCountHandle)
- bool decAndTestRefcount (RefCountHandle)
- Position faceToLeftEyePos (const Position &facePosition, float eyeDistance, float faceRollAngle)

  Estimates the left eye position from the face position, face roll angle and eye distance.
- Position faceToRightEyePos (const Position &facePosition, float eyeDistance, float faceRollAngle)

  Estimates the right eye position from the face position, face roll angle and eye distance.
- void faceToEyesPos (const Position &facePosition, float faceRollAngle, float eyeDistance, Position &leftEye-Position, Position &rightEyePosition)

  Estimates the both eye positions from the face position, face roll angle and eye distance.
- std::ostream & operator<< (std::ostream &o, const FIR &fir)

  output operator for FIR's
- FRsdk::Image rotateImage (const FRsdk::Image &, ImageRotation)
- CountedPtr< ShapeImageBody > createImageBody (const Vertex *, const bool *mask, unsigned int width, unsigned int height, bool takeOwnerShip)

  create a ShapeImageBody from vertex and mask data.
- ShapeImage loadShapeImage (const std::string &)

  Constructs a ShapeImage from the given file which must be in one of the supported formats.
- ShapeImage loadShapeImage (std::istream &is)

  Builds a ShapeImage from the given stream which must provide one of the supported formats.
- Sample loadShapeSample (const std::string &)

  Constructs a Shape sample from the given file which must be in one of the supported formats.
- FRsdk::Image vignetting (const FRsdk::Image &, const FRsdk::Rgb &, int margin=10, int radius=0, SmoothingFunction sf=Gaussian)

  In photographic areas vignetting is a intensity blending of round or eliptical margin regions.

0.35.1.1 Detailed Description

The global name space for the SDK.

0.35.1.2 Typedef Documentation

0.35.1.2.1 typedef std::pair<Image, Eyes::Location> FRsdk::AnnotatedImage

An image with annotated eye positions.

0.35.1.2.2 typedef std::list<AnnotatedImage> FRsdk::AnnotatedImageSet

A set of annotated images.

0.35.1.2.3 typedef unsigned char FRsdk::Byte

A 8 bit byte representation.

Examples:

imagebody.cc.

0.35.1.2.4 typedef std::list<Image> FRsdk::ImageSet

container used for collection of images within FaceVACS-SDK

0.35.1.2.5 typedef std::pair< std::string, Score> FRsdk::Match

a named score for a match of two FIR's

0.35.1.2.6 typedef std::list< Match> FRsdk::Matches

the container used for a set of Matches.

0.35.1.2.7 typedef void* FRsdk::RefCountHandle

0.35.1.2.8 typedef std::list<Sample> FRsdk::SampleSet

container used for collection of Samples within FaceVACS-SDK

0.35.1.2.9 typedef std::list<Score> FRsdk::Scores

an ordered collection of score values

0.35.1.3 Enumeration Type Documentation

0.35.1.3.1 enum FRsdk::ImageRotation

Enumerator

ROTATECLOCKWISE

ROTATECOUNTERCLOCKWISE

ROTATEUPSIDEDOWN

0.35.1.3.2 enum FRsdk::SmoothingFunction

There are threee ways to blend the margin to the target color: Gaussian blends intensisty of each color channel with a half gaussian function, Linear blend is propotional to the border distance and Fixed sets all pixel of margin to the border color.

Enumerator

Fixed

Linear

Gaussian

0.35.1.4 Function Documentation

0.35.1.4.1 CaptureDevice FRsdk::createCaptureDevice (const Configuration & , const std::string & *Name*)

factory function for creating capture devices configured under FRSDK.CaptureDevices.Name using builtin capture device types a Camera Controller will be used if configured under FRSDK.CameraControllers.Name

Examples:

capdev.cc, and tracklife.cc.

0.35.1.4.2 CaptureDevice FRsdk::createCaptureDevice (const CountedPtr< CaptureDeviceBody > & , const Configuration & , const std::string & *camControlName*)

factory function for creating user defined capture devices using a Camera Controller configured under FRSDK.-CameraControllers.Name

0.35.1.4.3 CountedPtr<ShapeImageBody> FRsdk::createImageBody (const Vertex ∗ , const bool ∗ *mask,* unsigned int *width,* unsigned int *height,* bool *takeOwnerShip*)

create a ShapeImageBody from vertex and mask data.

Vertex and mask data have to be arranged in width columns and height rows without padding. The takeOwnerShip flag indicates whether the ShapeImageBody handles memory deallocation or not. If set to 'false', the programmer has to ensure that memory regions passed keep valid until destruction of the ShapeImageBody.

0.35.1.4.4 bool FRsdk::decAndTestRefcount (RefCountHandle)

0.35.1.4.5 void FRsdk::deleteRefcount (RefCountHandle)

0.35.1.4.6 void FRsdk::faceToEyesPos (const Position & *facePosition,* float *faceRollAngle,* float *eyeDistance,* Position & *leftEyePosition,* Position & *rightEyePosition*)

Estimates the both eye positions from the face position, face roll angle and eye distance.

output right eye position

Parameters

facePosition	the face position
faceRollAngle	face roll angle in radians
eyeDistance	face eye distance
leftEyePosition	output left eye position

0.35.1.4.7 Position FRsdk::faceToLeftEyePos (const Position & *facePosition,* float *eyeDistance,* float *faceRollAngle*)

Estimates the left eye position from the face position, face roll angle and eye distance.

face roll angle in radians

Parameters

facePosition	the face position
eyeDistance	face eye distance

0.35.1.4.8 Position FRsdk::faceToRightEyePos (const Position & *facePosition,* float *eyeDistance,* float *faceRollAngle*)

Estimates the right eye position from the face position, face roll angle and eye distance.

face roll angle in radians

Parameters

facePosition	the face position
eyeDistance	face eye distance

0.35.1.4.9 void FRsdk::incRefcount (RefCountHandle)

Referenced by FRsdk::CountedPtr< FRsdk::CaptureDeviceBody >::CountedPtr(), and FRsdk::CountedPtr< F-Rsdk::CaptureDeviceBody >::operator=().

0.35.1.4.10 ShapeImage FRsdk::loadShapeImage (const std::string &)

Constructs a ShapeImage from the given file which must be in one of the supported formats.

0.35.1.4.11 ShapeImage FRsdk::loadShapeImage (std::istream & *is*)

Builds a ShapeImage from the given stream which must provide one of the supported formats.

0.35.1.4.12 Sample FRsdk::loadShapeSample (const std::string &)

Constructs a Shape sample from the given file which must be in one of the supported formats.

This interface is intended to be used for 3D formats where intensity image information is included

0.35.1.4.13 RefCountHandle FRsdk::newRefcount (int *initialValue*)

0.35.1.4.14 std::ostream& FRsdk::operator<< (std::ostream & *o*, const FIR & *fir*)

output operator for FIR's

Parameters

o	the stream to write to
fir	the FIR to write

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, eyesfind.cc, idialog.h, and vdialog.h.

0.35.1.4.15 FRsdk::Image FRsdk::rotateImage (const FRsdk::Image & , ImageRotation)

0.35.1.4.16 FRsdk::Image FRsdk::vignetting (const FRsdk::Image & , const FRsdk::Rgb & , int *margin* = `10`, int *radius* = `0`, SmoothingFunction *sf* = `Gaussian`)

In photographic areas vignetting is a intensity blending of round or eliptical margin regions.

In context of FRsdk it is a way to over blend margin to a defined color. optionally round edges might be created

Examples:

vignetting.cc.

0.35.2 FRsdk::Bmp Namespace Reference

Bitmap (BMP) image implementation.

Functions

- Image load (const BITMAPINFO ∗bi, const Byte ∗image, const std::string &name, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  constructs a image representation from the specified bitmap image in memory.
- Image load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  load the Bmp image from file; the image will get the name of the file
- Image load (std::istream &stream, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  load the Bmp image from a stream
- void save (const Image &, const std::string &filename)

  saves the image to a file with the given file name.
- unsigned int getBitmapInfoSize (const Image &img)

  get the total size of bitmap info; returns sizeof(BITMAPINFOHEADER)
- void writeBitmapInfo (const Image &img, BITMAPINFO ∗info)

  write bitmap info to buffer; buffer size has to be at least the size returned by getBitmapInfoSize().

0.35.2.1 Detailed Description

Bitmap (BMP) image implementation. FaceVACS-SDK supports the following bitmap formats:

- loading:
  - 8, 16, 24, 32 bit pixel size,
  - BI_BITFIELD and BI_RGB compression mode
  - saving:
  - 8 and 24 bit pixel size
  - BI_RGB compression mode

MT-safe (reentrant) It is safe to call the functions concurrently from different threads.

0.35.2.2 Function Documentation

0.35.2.2.1 unsigned int FRsdk::Bmp::getBitmapInfoSize (const Image & *img*)

get the total size of bitmap info; returns sizeof(BITMAPINFOHEADER)

- sizeof(color map)

0.35.2.2.2 Image FRsdk::Bmp::load (const BITMAPINFO ∗ *bi*, const Byte ∗ *image*, const std::string & *name*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

constructs a image representation from the specified bitmap image in memory.

The bi pointer points to a BITMAPINFOHEADER followed by an optional color table. The existence of a colortable depends on the image type. The Byte pointer has to point to bitmap data as described by the bitmap information. During construction of the BmpImage, BITMAPINFO and bitmap data is copied, so they can be safely discarded afterwards without corrupting the BmpImage. This function is available on Win32 platforms only.

Parameters

bi	points to the bitmap info header followed by color table
image	points to the bitmap data
name	give a name to the image

Examples:

acquisition.cc, and trackrec.cc.

0.35.2.2.3 Image FRsdk::Bmp::load (const std::string & *filename*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

load the Bmp image from file; the image will get the name of the file

0.35.2.2.4 Image FRsdk::Bmp::load (std::istream & *stream*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

load the Bmp image from a stream

0.35.2.2.5 void FRsdk::Bmp::save (const Image & , const std::string & *filename*)

saves the image to a file with the given file name.

Note that the bitmap format of the file stored can differ from that used upon construction.

0.35.2.2.6 void FRsdk::Bmp::writeBitmapInfo (const Image & *img*, BITMAPINFO * *info*)

write bitmap info to buffer; buffer size has to be at least the size returned by getBitmapInfoSize().

Note that the bitmap format stored in info is the format of the FRsdk::Image color representation that can differ from that used when constructing an image from bitmap data. To access bitmap data use img.colorRepresentation()

0.35.3 FRsdk::Enrollment Namespace Reference

the namespace for the enrollment facility

Classes

- class FeedbackBody

  Body class for Feedback.
- class Feedback

  explicit template instantiation for win32
- class Processor

  this class represents the interface to the enrollment process Calls of process() are serialized but using multiple Processors enrollements can be done in parallel (one processor per thread).

0.35.3.1 Detailed Description

the namespace for the enrollment facility Enrollment is the use case of constructing a FIR from a number of images of a human face. One or more samples (face images) of the same person are captured and processed into a FIR. The results of the enrollment are returned by a callback mechanism via the Feedback object.

The constraint to use images of one and the same person only is essentially. Due to the limited "capacity" of a FIR and the properties of the clustering algorithm used, poor biometric performance may be the result if combining images or merging FIR's from different persons is attempted.

0.35.4 FRsdk::Eyes Namespace Reference

the name space for the eyes finding facility

Classes

- struct Location

  The Eyes::Location describes a location in the image where eyes within a face have been found.
- class Finder

  Eyes::Finder (handle) This class represents a interface to the eye finding procedure.

Typedefs

- typedef std::list< Location > LocationSet

  common used set of Locations

0.35.4.1 Detailed Description

the name space for the eyes finding facility

0.35.4.2 Typedef Documentation

0.35.4.2.1 typedef std::list<Location> FRsdk::Eyes::LocationSet

common used set of Locations

0.35.5 FRsdk::Face Namespace Reference

the name space for the face finding facility

Classes

- struct Location

  The Face::Location describes a image location where a face was found.
- class Finder

  Face::Finder (handle) This class represents a interface to the face finding procedure.
- class Tracker

  The Face Tracker locates and tracks faces across a sequence of images in an efficient way by analyzing the spatial and temporal dependencies between faces in subsequent images.

Typedefs

- typedef std::list< Location > LocationSet

  common used set of Locations

0.35.5.1 Detailed Description

the name space for the face finding facility

0.35.5.2 Typedef Documentation

0.35.5.2.1 typedef std::list<Location> FRsdk::Face::LocationSet

common used set of Locations

0.35.6 FRsdk::Identification Namespace Reference

the namespace for the identification facility

Classes

- class FeedbackBody

  Body class for Feedback.
- class Feedback

  explicit template instantiation for win32
- class Processor

  this class represents the interface to the identification process.

0.35.6.1 Detailed Description

the namespace for the identification facility Identification is the use case of identification of human faces. One or more samples (face images) are captured and processed into a FIR and matched against a set of FIR's. The result of the identification is a match set containing the matches ordered by score. This set is returned by callback mechanism using the Feedback object.

0.35.7 FRsdk::ImageIO Namespace Reference

Classes

- class PropertiesFeedbackBody

  abstract PropertiesFeedback body
- class PropertiesFeedback

  explicit template instantiation for win32

Enumerations

- enum ColorMode {
  Unknown, RGB, RGBA, Alpha,
  Intensity, Palette, YCbCr, YUY2,
  YCCK, CMYK, YVYU, UYVY,
  RLE8, RLE4, BITFIELDS, YVU9,
  YV12, I420, IYUV, Y800,
  Y8, CIELAB, ICCLAB, ITULAB,
  LOGL, LOGLUV, MASK, SEPARATED }

Functions

- Image load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from the given file.
- Image load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from the given stream.

0.35.7.1 Enumeration Type Documentation

0.35.7.1.1 enum FRsdk::ImageIO::ColorMode

Enumerator

Unknown

RGB

RGBA

Alpha

Intensity

Palette

YCbCr

YUY2

YCCK

CMYK

YVYU

UYVY

RLE8

RLE4

BITFIELDS

YVU9

YV12

I420

IYUV

Y800

Y8

CIELAB

ICCLAB

ITULAB

LOGL

LOGLUV

MASK

SEPARATED

0.35.7.2 Function Documentation

0.35.7.2.1 Image FRsdk::ImageIO::load (const std::string & *filename*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

Constructs an image representation from the given file.

Supported formats are FRsdk::Pgm, FRsdk::Bmp, FRsdk::Png, FRsdk::Jpeg

Examples:

enroll.cc, verify.cc, and verifyan.cc.

0.35.7.2.2 Image FRsdk::ImageIO::load (std::istream & *is*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

Constructs an image representation from the given stream.

Supported formats are FRsdk::Pgm, FRsdk::Bmp, FRsdk::Png, FRsdk::Jpeg

0.35.8 FRsdk::ISO_19794_5 Namespace Reference

Support for image formats defined by ISO/IEC 19794-5:2005.

Namespaces

- FullFrontal

  Support for Full Frontal type as defined in ISO_19794_5 section 8.
- TokenFace

  Support for Token Face Image type as defined in ISO_19794_5 9.2.

0.35.8.1 Detailed Description

Support for image formats defined by ISO/IEC 19794-5:2005. This namespace provides support for the face image formats defined by ISO/IEC 19794-5:2005, Biometric Data Interchange Formats - Part 5: Face Image Data. In the documentation of the namespace members the term 'ISO_19794_5' refers to the standard document ISO/IEC 19794-5:2005 final draft.

Use ISO_19794_5::FullFrontal::Test to determine compliance of a portrait with the ISO 19794-5:2005 requirements for Full Frontal Images.

Use ISO_19794_5::TokenFace::extract or ISO_19794_5::TokenFace::extractMinimal to produce images meeting the requirements of the ISO 19794-5:2005 TokenFace Image type.

Use ISO_19794_5::TokenFace::read and ISO_19794_5::TokenFace::write for reading and writing Token Face Images from and to files.

0.35.9 FRsdk::ISO_19794_5::FullFrontal Namespace Reference

Support for Full Frontal type as defined in ISO_19794_5 section 8.

Classes

- class Creator

  Extract a Full Frontal Image from the source image.
- class Compliance

  Compliance assessment results.
- class Boundaries

  Boundaries are used from the FullFrontal::Test.
- class Test

  Compliance assessment.

0.35.9.1 Detailed Description

Support for Full Frontal type as defined in ISO_19794_5 section 8. Compliance with ISO 19794-5:2005 Full Frontal Image type.

According to ISO_19794_5 the Full Frontal image is used to store the extracted face information from any other image source.

This namespace provides a framework for testing compliance with the ISO 19794-5:2005 Full Frontal Image requirements including the Best Practice recommendations for Full Frontal Images.

0.35.10 FRsdk::ISO_19794_5::TokenFace Namespace Reference

Support for Token Face Image type as defined in ISO_19794_5 9.2.

Classes

- class Creator

  Extract a Token Face Image from the source image.

Functions

- AnnotatedImageSet read (std::istream &i)

    TokenFace::read and TokenFace::write provide support for the ISO_19794_5, Common Biometric Exchange Formats Framework (CBEFF), Facial Data Interchange Format.
- void write (std::ostream &o, const AnnotatedImageSet &)

    Use TokenFace::write for storing annotated faces to ISO_19794_5 CBEFF compliant Token Face Image type format to the stream.

0.35.10.1 Detailed Description

Support for Token Face Image type as defined in ISO_19794_5 9.2. According to ISO_19794_5 the Token Face Image is used to store the extracted face information from any other image source.

0.35.10.2 Function Documentation

0.35.10.2.1 AnnotatedImageSet FRsdk::ISO_19794_5::TokenFace::read (std::istream & *i*)

TokenFace::read and TokenFace::write provide support for the ISO_19794_5, Common Biometric Exchange Formats Framework (CBEFF), Facial Data Interchange Format.

The CBEFF Face Image Data format has the following structure (according to the ISO_19794_5):

```
-----------------------
| CBEFF Header        |
-----------------------
| Facial Record Header  |
-----------------------
| Facial Record Data 1  |
-----------------------
...
-----------------------
| Facial Record Data N  |
-----------------------
| CBEFF Signature       |
-----------------------
```

The I/O functions use the Minimum Simple Patron Format (byte oriented) as described in the ISO/IEC 19785-1 standard. The CBEFF Header contains just 4 Bytes (CBEFF_BDB_format_owner (2 Bytes), CBEFF_BDB_format_type (2 Bytes)). The CBEFF Signature is empty (0 Bytes).

Use TokenFace::read for reading annotated faces from ISO_19794_5 CBEFF formated buffer. The face annotations for both eyes are assigned as follows: the eye position with the lowest x-coordinate is assigned to Eyes::Location::first and the other to Eyes::Location::second.

TokenFace::read never throws an exception if no annotations are found. In that case the positions are initialized with (0,0).

0.35.10.2.2 void FRsdk::ISO_19794_5::TokenFace::write (std::ostream & *o*, const AnnotatedImageSet &)

Use TokenFace::write for storing annotated faces to ISO_19794_5 CBEFF compliant Token Face Image type format to the stream.

The function never checks if the given faces are compliant with the Token Face Image requirements - this is up to the caller of this function. For that purpose the FRsdk provides the functions FRsdk::ISO_19794_5::TokenFace::extract and FRsdk::ISO_19794_5::TokenFace::extractMinimal which are defined in the <frsdk/tokenface.h> header file.

Examples:

acquisition.cc.

0.35.11 FRsdk::Jpeg Namespace Reference

JPEG Image support.

Classes

- struct Properties

  contains properties of a JPEG image

Functions

- Image load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from the given file which must be in jpeg format.
- Image load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from the given stream which is in jpeg format.
- Image load (const char ∗buf, unsigned int size, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from memory which is in jpeg format.
- int save (const Image &img, std::ostream &, int quality=100)

  saves the image to the ostream using given jpeg quality (1 .
- int save (const Image &img, const std::string &filename, int quality=100)

  saves the image to a file given by name using given jpeg quality (1 .
- Properties saveWithSizeConstraint (const Image &img, std::ostream &, int maxSize)

  saves the image to the ostream which has a size constraint given by maxSize (Byte).

0.35.11.1 Detailed Description

JPEG Image support. JPEG (pronounced "jay-peg") is a standardized compression method for full-color and gray scale images. The FaceVACS-SDK jpeg supporting software is based in part on the work of the Independent JPEG Group.

FaceVACS-SDK supports the following jpeg formats:

- loading:
  - color and gray scale images of any compression quality
- saving:
  - color and gray scale images, with quality parameter.

(reentrant) MT-safe It is safe to call the member functions concurrently

from different threads.

0.35.11.2 Function Documentation

0.35.11.2.1 Image FRsdk::Jpeg::load (const std::string & *filename*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

Constructs an image representation from the given file which must be in jpeg format.

Examples:

acquisition.cc, cropfullfrontal.cc, eyesfind.cc, identify.cc, trackrec.cc, and vignetting.cc.

0.35.11.2.2 Image FRsdk::Jpeg::load (std::istream & *is*, const ImageIO::PropertiesFeedback & *fb* = ImageIO::PropertiesFeedback())

Constructs an image representation from the given stream which is in jpeg format.

0.35.11.2.3 Image FRsdk::Jpeg::load (const char ∗ *buf*, unsigned int *size*, const ImageIO::PropertiesFeedback & *fb* = ImageIO::PropertiesFeedback())

Constructs an image representation from memory which is in jpeg format.

0.35.11.2.4 int FRsdk::Jpeg::save (const Image & *img*, std::ostream & *,* int *quality* = 100)

saves the image to the ostream using given jpeg quality (1 .

. 100) note that the jpeg quality ave influences on the recognition rate.

The measured increases of the equal error rate is: quality = 100, no increase of equal error rate quality = 80, increases the equal error rate by 3.5% quality = 50, increases the equal error rate by 10.5%

Returns the number of bytes written to the stream

Examples:

cropfullfrontal.cc, tracklife.cc, and vignetting.cc.

0.35.11.2.5 int FRsdk::Jpeg::save (const Image & *img*, const std::string & *filename*, int *quality* = 100)

saves the image to a file given by name using given jpeg quality (1 .

. 100) note that the jpeg quality ave influences on the recognition rate. For a detailed description see the ostream save function.

Returns the number of bytes written to the file.

0.35.11.2.6 Properties FRsdk::Jpeg::saveWithSizeConstraint (const Image & *img*, std::ostream & *,* int *maxSize*)

saves the image to the ostream which has a size constraint given by maxSize (Byte).

Returns the achieved JPEG image Properties.

0.35.12 FRsdk::Jpeg2000 Namespace Reference

JPEG 2000 Image support.

Functions

- Image load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    Loads an image representation from the given file which must be in jpeg 2000 format.
- Image load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    Loads an image representation from the given stream which is expected to contain jpeg 2000 format.
- Image load (const char ∗buf, unsigned int size, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    Loads an image representation from memory which is expected to contain jpeg 2000 format.

0.35.12.1 Detailed Description

JPEG 2000 Image support. JPEG 2000 (pronounced "jay-peg") is a standardized compression method for full-color and gray scale images. The FaceVACS-SDK jpeg 2000 supporting software is based on JASPER.

FaceVACS-SDK supports the following jpeg formats:

- loading:
  - color and gray scale images of any compression quality
- saving:
  - writing of j2k is not supported.

(reentrant) MT-safe It is safe to call functions concurrently from different threads.

0.35.12.2 Function Documentation

0.35.12.2.1 Image FRsdk::Jpeg2000::load (const std::string & *filename*, const ImageIO::PropertiesFeedback & *fb* = ImageIO::PropertiesFeedback())

Loads an image representation from the given file which must be in jpeg 2000 format.

Examples:

acquisition.cc, and trackrec.cc.

0.35.12.2.2 Image FRsdk::Jpeg2000::load (std::istream & *is*, const ImageIO::PropertiesFeedback & *fb* = ImageIO::PropertiesFeedback())

Loads an image representation from the given stream which is expected to contain jpeg 2000 format.

0.35.12.2.3 Image FRsdk::Jpeg2000::load (const char * *buf*, unsigned int *size*, const ImageIO::PropertiesFeedback & *fb* = ImageIO::PropertiesFeedback())

Loads an image representation from memory which is expected to contain jpeg 2000 format.

0.35.13 FRsdk::Pgm Namespace Reference

PGM/PPM image format support.

Functions

- Image load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::Properties-Feedback())

  constructs an image representation from the specified pgm format file
- void save (const Image &img, const std::string &filename)

  Saves the image to filename; gray scale images will be stored in pgm-format, color images in ppm-format.

0.35.13.1 Detailed Description

PGM/PPM image format support. This functions provide support for PGM (portable graymap file format) and PPM (portable pixmap file format).

MT-safe (reentrant) It is safe to call the functions concurrently from different threads.

0.35.13.2 Function Documentation

0.35.13.2.1 Image FRsdk::Pgm::load (const std::string & *filename*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

constructs an image representation from the specified pgm format file

Examples:

acquisition.cc, and trackrec.cc.

0.35.13.2.2 void FRsdk::Pgm::save (const Image & *img*, const std::string & *filename*)

Saves the image to filename; gray scale images will be stored in pgm-format, color images in ppm-format.

0.35.14 FRsdk::Png Namespace Reference

PNG (Portable Network Graphics) image format support.

Functions

- Image load (const std::string &fileName, const ImageIO::PropertiesFeedback &fb=ImageIO::Properties-Feedback())

  Constructs an image representation from image file specified by name.
- Image load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  constructs an image representation from the specified PNG formated stream
- void save (const Image &img, std::ostream &os, int compressionLevel=6)

  Write image as PNG image to the given stream, the stream will not be closed.

0.35.14.1 Detailed Description

PNG (Portable Network Graphics) image format support. This functions provide support for PNG (portable networks graphics) image format.

MT-safe (reentrant) It is safe to call the functions concurrently from different threads.

0.35.14.2 Function Documentation

0.35.14.2.1 Image FRsdk::Png::load (const std::string & *fileName*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

Constructs an image representation from image file specified by name.

The file must be in PNG format.

Examples:

acquisition.cc, and trackrec.cc.

0.35.14.2.2 Image FRsdk::Png::load (std::istream & *is*, const ImageIO::PropertiesFeedback & *fb* = `ImageIO::PropertiesFeedback()`)

constructs an image representation from the specified PNG formated stream

0.35.14.2.3 void FRsdk::Png::save (const Image & *img*, std::ostream & *os*, int *compressionLevel* = 6)

Write image as PNG image to the given stream, the stream will not be closed.

The intention of the PNG image format is to be lossless. Compression is in the range from 0-9, 0 means no compression and fast reading/writing 9 means high compression and slower reading/writing. The default level is 6 which is a compromise in speed and compression.

Examples:

acquisition.cc.

0.35.15 FRsdk::Portrait Namespace Reference

Portrait characteristics and feature tests.

Namespaces

- Feature

  Test for features in the portrait.

Classes

- struct EthnicityMeasurements

  measurements for ethnicity detection, contains the probability that a person belongs to the ethnicity class
- class Characteristics

  Portrait Characteristics.
- class Analyzer

  Portrait Characteristics Analyzer, create Portrait characteristics from annotated images.

Functions

- Box earToEarChinCrownSurroundingBox (const Characteristics &)

  This function returns the smallest surrounding box of the face according to the chin, crown and ear positions estimated in portrait characteristics.

0.35.15.1 Detailed Description

Portrait characteristics and feature tests. This namespace provides support for measuring various face portrait characteristics and testing for certain features in the portrait.

Use Portrait::Analyzer to produce Portrait::Characteristics of a portrait. The result of the analysis can be used to determine compliance of a portrait with ISO 19794-5:2005.

Use Portrait::Feature::Test to test for portrait features.

0.35.15.2 Function Documentation

0.35.15.2.1 Box FRsdk::Portrait::earToEarChinCrownSurroundingBox (const Characteristics &)

This function returns the smallest surrounding box of the face according to the chin, crown and ear positions estimated in portrait characteristics.

It is a convinience function only.

Examples:

acquisition.cc.

0.35.16 FRsdk::Portrait::Feature Namespace Reference

Test for features in the portrait.

Classes

- class Set

  Feature assessment results.
- class Test

  Test for features in a portrait.

Enumerations

- enum Gender { male, female }

  Gender.
- enum Ethnicity { white, black, asian }

  Ethnicity.

0.35.16.1 Detailed Description

Test for features in the portrait. This namespace provides tests for features in a portrait. These features are not required by ISO 19794-5:2005, although they might be of interest for other reasons.

0.35.16.2 Enumeration Type Documentation

0.35.16.2.1 enum **FRsdk::Portrait::Feature::Ethnicity**

Ethnicity.

Enumerator

white

black

asian

0.35.16.2.2 enum **FRsdk::Portrait::Feature::Gender**

Gender.

Enumerator

male

female

0.35.17 FRsdk::Tools Namespace Reference

Misc tools.

Functions

Score file i/o functions

- bool storeScoreValueFile (const std::string &probeRecordId, unsigned int positionInProbe, const FRsdk::-Scores &scores, const std::string &destinationDirectory)

  store the scores into a file in destinationDirectory
- bool loadScoreValueFile (const std::string &probeRecordId, unsigned int positionInProbe, FRsdk::Scores &scores, const std::string &sourceDirectory)

  load the scores for a person ID file from sourceDirectory
- bool loadScoreValueFile (const std::string &filename, FRsdk::Scores &scores)

  load the scores from a file

Match list file i/o functions

- bool storeMatchListFile (const std::string &probeRecordId, unsigned int positionInProbe, const FRsdk::-Matches &matches, const std::string &destinationDirectory)

  store the matches into a file in destinationDirectory
- bool loadMatchListFile (const std::string &probeRecordId, unsigned int positionInProbe, FRsdk::Matches &matches, const std::string &sourceDirectory)

  load the matches for a person ID file from sourceDirectory
- bool loadMatchListFile (const std::string &filename, FRsdk::Matches &matches)

  load the matches from a file

0.35.17.1 Detailed Description

Misc tools.

0.35.17.2 Function Documentation

0.35.17.2.1 bool FRsdk::Tools::loadMatchListFile (const std::string & *probeRecordId,* unsigned int *positionInProbe,* FRsdk::Matches & *matches,* const std::string & *sourceDirectory*)

load the matches for a person ID file from sourceDirectory

Parameters

probeRecordId	person ID of the probe set element the file is generated for
positionInProbe	position of probeRecordId within probe set
matches	to write the read data to
sourceDirectory	directory where the file has to be written to

Return values

bool	true if successfully loaded, otherwise false

0.35.17.2.2 bool FRsdk::Tools::loadMatchListFile (const std::string & *filename,* FRsdk::Matches & *matches*)

load the matches from a file

Parameters

filename	the name of the file to read from
matches	to write the read data to

Return values

bool	true if successfully loaded, otherwise false

0.35.17.2.3 bool FRsdk::Tools::loadScoreValueFile (const std::string & *probeRecordId,* unsigned int *positionInProbe,* FRsdk::Scores & *scores,* const std::string & *sourceDirectory*)

load the scores for a person ID file from sourceDirectory

Parameters

probeRecordId	person ID of the probe set element the file is generated for
positionInProbe	position of probeRecordId within probe set
scores	to write the read data to
sourceDirectory	directory where the file has to be written to

Return values

bool	true if successfully loaded, otherwise false

0.35.17.2.4 bool FRsdk::Tools::loadScoreValueFile (const std::string & *filename,* FRsdk::Scores & *scores*)

load the scores from a file

Parameters

filename	the name of the file to load
scores	to write the read data to

Return values

bool	true if successfully loaded, otherwise false

0.35.17.2.5 bool FRsdk::Tools::storeMatchListFile (const std::string & *probeRecordId,* unsigned int *positionInProbe,* const FRsdk::Matches & *matches,* const std::string & *destinationDirectory*)

store the matches into a file in destinationDirectory

Parameters

probeRecordId	person ID of the probe set element the file is generated for
positionInProbe	position of probeRecordId within probe set
matches	list of matches achieved by probeRecordId against gallery
destination-Directory	directory where the file has to be written to

Return values

bool	true if successfully stored, otherwise false

0.35.17.2.6 bool FRsdk::Tools::storeScoreValueFile (const std::string & *probeRecordId,* unsigned int *positionInProbe,* const FRsdk::Scores & *scores,* const std::string & *destinationDirectory*)

store the scores into a file in destinationDirectory

Parameters

probeRecordId	person ID of the probe set element the file is generated for

positionInProbe	position of probeRecordId within probe set
scores	list of scores achieved by probeRecordId against gallery
destination-Directory	directory where the file has to be written to

Return values

bool	true if successfully stored, otherwise false

0.35.18 FRsdk::Verification Namespace Reference

the namespace for the verification facility

Classes

- class FeedbackBody

  Body class for Feedback.
- class Feedback

  explicit template instantiation for win32
- class Processor

  this class represents the interface to the verification process Calls of process() are serialized but using multiple Processors verifications can be done in parallel (one processor per thread).

0.35.18.1 Detailed Description

the namespace for the verification facility Verification is the use case of authenticating human faces. One or more samples are processed into a FIR and then matched against an input (reference) FIR. The results of the verification are returned via the Feedback object.

0.36 Class Documentation

0.36.1 FRsdk::Portrait::Analyzer Class Reference

Portrait Characteristics Analyzer, create Portrait characteristics from annotated images.

```
#include <portrait.h>
```

Public Member Functions

- Analyzer (const Configuration &)
- Analyzer (const Analyzer &)
- Analyzer & operator= (const Analyzer &)
- ∼Analyzer ()
- Characteristics analyze (const AnnotatedImage &a) const

  Depending on color information of image analyzeGrayScale() or analyzeColor() is called.
- Characteristics analyzeGrayScale (const AnnotatedImage &a) const

  Create portrait characteristics from grayscale image f.
- Characteristics analyzeColor (const AnnotatedImage &a) const

  Create portrait characteristics from color image f.

0.36.1.1 Detailed Description

Portrait Characteristics Analyzer, create Portrait characteristics from annotated images.

(serialized) MT-safe It is safe to call the member functions concurrently from different threads.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.1.2 Constructor & Destructor Documentation

0.36.1.2.1 FRsdk::Portrait::Analyzer::Analyzer (const Configuration &)

0.36.1.2.2 FRsdk::Portrait::Analyzer::Analyzer (const Analyzer &)

0.36.1.2.3 FRsdk::Portrait::Analyzer::∼Analyzer ( )

0.36.1.3 Member Function Documentation

0.36.1.3.1 Characteristics FRsdk::Portrait::Analyzer::analyze (const AnnotatedImage & *a*) const

Depending on color information of image analyzeGrayScale() or analyzeColor() is called.

0.36.1.3.2 Characteristics FRsdk::Portrait::Analyzer::analyzeColor (const AnnotatedImage & *a*) const

Create portrait characteristics from color image f.

In case the image is based on intensity values and is only transformated to a color images, some tests will lead to wrong results.

0.36.1.3.3 Characteristics FRsdk::Portrait::Analyzer::analyzeGrayScale (const AnnotatedImage & *a*) const

Create portrait characteristics from grayscale image f.

All tests using color information are skipped.

0.36.1.3.4 Analyzer& FRsdk::Portrait::Analyzer::operator= (const Analyzer &)

The documentation for this class was generated from the following file:

- frsdk/portrait.h

0.36.2 FRsdk::ISO_19794_5::FullFrontal::Boundaries Class Reference

Boundaries are used from the FullFrontal::Test.

```
#include <portraittests.h>
```

Public Member Functions

- Boundaries (const Boundaries &)
- Boundaries & operator= (const Boundaries &)
- ∼Boundaries ()
- float lowerBoundVerticalPosition () const

  lower and upper bound of vertical positioning ratio
- float upperBoundVerticalPosition () const

- float leftBoundCenteredFace () const

  left and right bound of horizontal positioning ratio
- float rightBoundCenteredFace () const
- float minWidthOfHeadRatio () const

  minimal and maximal ratio of width and length of head
- float minLengthOfHeadRatio () const
- float maxWidthOfHeadRatio () const
- float maxLengthOfHeadRatio () const
- float minWidthOfHeadRatioBestPractice () const
- float minLengthOfHeadRatioBestPractice () const
- float maxWidthOfHeadRatioBestPractice () const
- float maxLengthOfHeadRatioBestPractice () const
- float minHeadWidth () const

  According to ISO Standard paragraph 8.4.1 the minimal resolution is defined by minimal head width in pixel.
- float minHeadWidthBestPractice () const

  According to ISO Standard paragraph A 3.1.1 the minimal resolution is defined by minimal head width in pixel.
- float minWidthToHeightRatioBestPractice () const

  min and max of ratio width versus height of the image
- float maxWidthToHeightRatioBestPractice () const
- float lowExposerThreshold () const

  thresholds of under and over exposer
- float highExposerThreshold () const
- float minimalGrayScaleBounding () const

  minimal number of gray scales in the face region
- float minimalSkinColourRatio () const

  minimal percentage of natural skin colour in face region
- float maxFrontalPoseDeviation () const

  maximal deviation from frontal in radiant
- float maxFrontalPoseDeviationBestPractice () const
- float poseMaxRotation () const

  maximal pose rotation (roll) in radiant
- float poseMaxRotationBestPractice () const
- float maxDeviationFromLighting () const

  maximal deviation from uniform lighting
- float eyesOpenThreshold () const

  threshold determining glasses are to heavy
- float eyesGazeFrontalThreshold () const

  threshold determining eyes looking frontal to the camera
- float eyesNotRedThreshold () const

  threshold determining eyes are red or not
- float minimalSharpness () const

  threshold to devide unsharp from sharp images
- float closedMouthThreshold () const

  threshold to devide open from closed mouthes
- float hotSpotsThreshold () const

  threshold, maximal percentage of hot spot pixels in face region
- float wearsGlassesThreshold () const

  threshold determining if persons wears glasses or not
- float tintedGlassesThreshold () const

  threshold determining if persons wears tinted glasses

Friends

- class Test

0.36.2.1 Detailed Description

Boundaries are used from the FullFrontal::Test.

The concrete boundaries used can be fetched by calling the function FullFrontal::Test::boundaries(). It describes the boundaries (limits) which each of the tested characteristics must be within. Each item of the boundaries is read only and can be modified only by using the configuration editor.

Examples:

acquisition.cc.

0.36.2.2 Constructor & Destructor Documentation

0.36.2.2.1 FRsdk::ISO_19794_5::FullFrontal::Boundaries::Boundaries (const Boundaries &)

0.36.2.2.2 FRsdk::ISO_19794_5::FullFrontal::Boundaries::∼Boundaries ( )

0.36.2.3 Member Function Documentation

0.36.2.3.1 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::closedMouthThreshold ( ) const

threshold to devide open from closed mouthes

Examples:

acquisition.cc.

0.36.2.3.2 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::eyesGazeFrontalThreshold ( ) const

threshold determining eyes looking frontal to the camera

Examples:

acquisition.cc.

0.36.2.3.3 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::eyesNotRedThreshold ( ) const

threshold determining eyes are red or not

Examples:

acquisition.cc.

0.36.2.3.4 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::eyesOpenThreshold ( ) const

threshold determining glasses are to heavy

threshold determining eyes are open or not

Examples:

acquisition.cc.

0.36.2.3.5 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::highExposerThreshold ( ) const

Examples:

acquisition.cc.

0.36.2.3.6 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::hotSpotsThreshold ( ) const

threshold, maximal percentage of hot spot pixels in face region

Examples:

acquisition.cc.

0.36.2.3.7 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::leftBoundCenteredFace ( ) const

left and right bound of horizontal positioning ratio

Examples:

acquisition.cc.

0.36.2.3.8 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::lowerBoundVerticalPosition ( ) const

lower and upper bound of vertical positioning ratio

Examples:

acquisition.cc.

0.36.2.3.9 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::lowExposerThreshold ( ) const

thresholds of under and over exposer

Examples:

acquisition.cc.

0.36.2.3.10 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxDeviationFromLighting ( ) const

maximal deviation from uniform lighting

Examples:

acquisition.cc.

0.36.2.3.11 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxFrontalPoseDeviation ( ) const

maximal deviation from frontal in radiant

Examples:

acquisition.cc.

0.36.2.3.12 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxFrontalPoseDeviationBestPractice ( ) const

0.36.2.3.13 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxLengthOfHeadRatio ( ) const

Examples:

acquisition.cc.

0.36.2.3.14 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxLengthOfHeadRatioBestPractice ( ) const

0.36.2.3.15 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxWidthOfHeadRatio ( ) const

Examples:

acquisition.cc.

0.36.2.3.16 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxWidthOfHeadRatioBestPractice ( ) const

0.36.2.3.17 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::maxWidthToHeightRatioBestPractice ( ) const

0.36.2.3.18 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minHeadWidth ( ) const

According to ISO Standard paragraph 8.4.1 the minimal resolution is defined by minimal head width in pixel.

Examples:

acquisition.cc.

0.36.2.3.19 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minHeadWidthBestPractice ( ) const

According to ISO Standard paragraph A 3.1.1 the minimal resolution is defined by minimal head width in pixel.

0.36.2.3.20 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minimalGrayScaleBounding ( ) const

minimal number of gray scales in the face region

Examples:

acquisition.cc.

0.36.2.3.21 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minimalSharpness ( ) const

threshold to devide unsharp from sharp images

Examples:

acquisition.cc.

0.36.2.3.22 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minimalSkinColourRatio ( ) const

minimal percentage of natural skin colour in face region

0.36.2.3.23 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minLengthOfHeadRatio ( ) const

Examples:

acquisition.cc.

0.36.2.3.24 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minLengthOfHeadRatioBestPractice ( ) const

0.36.2.3.25 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minWidthOfHeadRatio ( ) const

minimal and maximal ratio of width and length of head

Examples:

acquisition.cc.

0.36.2.3.26 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minWidthOfHeadRatioBestPractice ( ) const

0.36.2.3.27 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::minWidthToHeightRatioBestPractice ( ) const

min and max of ratio width versus height of the image

0.36.2.3.28 Boundaries& FRsdk::ISO_19794_5::FullFrontal::Boundaries::operator= (const Boundaries &)

0.36.2.3.29 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::poseMaxRotation ( ) const

maximal pose rotation (roll) in radiant

Examples:

acquisition.cc.

0.36.2.3.30 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::poseMaxRotationBestPractice ( ) const

0.36.2.3.31 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::rightBoundCenteredFace ( ) const

Examples:

acquisition.cc.

0.36.2.3.32 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::tintedGlassesThreshold ( ) const

threshold determining if persons wears tinted glasses

Examples:

acquisition.cc.

0.36.2.3.33 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::upperBoundVerticalPosition ( ) const

Examples:

acquisition.cc.

0.36.2.3.34 float FRsdk::ISO_19794_5::FullFrontal::Boundaries::wearsGlassesThreshold ( ) const

threshold determining if persons wears glasses or not

Examples:

acquisition.cc.

0.36.2.4 Friends And Related Function Documentation

0.36.2.4.1 friend class Test [friend]

The documentation for this class was generated from the following file:

- frsdk/portraittests.h

0.36.3 FRsdk::Box Class Reference

The class Box describes a rectangular box in discrete 2D coordinates defined by 2 opposite points, origin and end, where origin.x <= end.x and origin.y <= end.y.

```
#include <position.h>
```

Public Member Functions

- Box (int x1, int y1, int x2, int y2)

  c'tor which constructs from discrete coordinates
- Box (const Box &rhs)

  copy c'tor
- int originx () const

  returns the x coordinate of origin
- int originy () const

  returns the y coordinate of origin
- int endx () const

  returns the x coordinate of end
- int endy () const

  returns the y coordinate of end

0.36.3.1 Detailed Description

The class Box describes a rectangular box in discrete 2D coordinates defined by 2 opposite points, origin and end, where origin.x <= end.x and origin.y <= end.y.

Note that origin belongs to the box and end does not (to support empty boxes). origin is represented by x1 and y1, end is represented by x2 and y2.

Examples:

acquisition.cc.

0.36.3.2 Constructor & Destructor Documentation

0.36.3.2.1 FRsdk::Box::Box (int *x1*, int *y1*, int *x2*, int *y2*)

c'tor which constructs from discrete coordinates

0.36.3.2.2 FRsdk::Box::Box (const Box & *rhs*) [inline]

copy c'tor

0.36.3.3 Member Function Documentation

0.36.3.3.1 int FRsdk::Box::endx ( ) const [inline]

returns the x coordinate of end

Examples:

acquisition.cc.

0.36.3.3.2 int FRsdk::Box::endy ( ) const [inline]

returns the y coordinate of end

Examples:

acquisition.cc.

0.36.3.3.3 int FRsdk::Box::originx ( ) const [inline]

returns the x coordinate of origin

Examples:

acquisition.cc.

0.36.3.3.4 int FRsdk::Box::originy ( ) const [inline]

returns the y coordinate of origin

Examples:

acquisition.cc.

The documentation for this class was generated from the following file:

- frsdk/position.h

0.36.4 FRsdk::CaptureDevice Class Reference

Capture Device handle (interface)

`#include <capdev.h>`

Public Member Functions

- CaptureDevice (const CountedPtr< CaptureDeviceBody > &i)
- Image capture () const

  capture an image, the return value is a valid Image.
- Image release () const

  release an still image/single shot, beside the continuouse capturing of images some concrete CaptureDevice supports releasing a snapshot.
- void control (const FRsdk::Image &img, const std::list< Face::Location > &faceLocations)

  gradually adapt capture parameters towards optimal representation of faces
- unsigned int gain () const

  return the current value of the gain
- virtual std::pair< unsigned int, unsigned int > gainRange () const

  return the gain range (min,max)
- virtual void setGain (unsigned int newGain)

  set the gain to newGain
- unsigned int exposure () const

  return the current value of the exposure time
- virtual std::pair< unsigned int, unsigned int > exposureRange () const

  return the exposure range (min,max)
- virtual void setExposure (unsigned int newExposure)

  set the exposure time to newExposure
- bool configure () const

  call up device configuration dialog provided by the device driver
- bool videoFormatDialog () const

  call up video format dialog provided by the device driver

- VideoFormat getVideoFormat () const

  retrieve current video format
- bool setVideoFormat (const VideoFormat &f)

  apply a video format.
- CaptureDeviceBody & body () const

  access to the body; enables clients to perform (dynamic) casting to the concrete body class.

0.36.4.1 Detailed Description

Capture Device handle (interface)

The interface contains members to access some values adjusted by control() and to get access to configuration dialogs of some special camera drivers. But the main goal is to capture images from a image stream and to make snap shots.

Examples:

capdev.cc, and tracklife.cc.

0.36.4.2 Constructor & Destructor Documentation

0.36.4.2.1 FRsdk::CaptureDevice::CaptureDevice (const CountedPtr< CaptureDeviceBody > & *i*) [inline]

0.36.4.3 Member Function Documentation

0.36.4.3.1 CaptureDeviceBody& FRsdk::CaptureDevice::body ( ) const [inline]

access to the body; enables clients to perform (dynamic) casting to the concrete body class.

0.36.4.3.2 Image FRsdk::CaptureDevice::capture ( ) const [inline]

capture an image, the return value is a valid Image.

Examples:

capdev.cc, and tracklife.cc.

0.36.4.3.3 bool FRsdk::CaptureDevice::configure ( ) const [inline]

call up device configuration dialog provided by the device driver

0.36.4.3.4 void FRsdk::CaptureDevice::control (const FRsdk::Image & *img*, const std::list< Face::Location > & *faceLocations*) [inline]

gradually adapt capture parameters towards optimal representation of faces

Examples:

tracklife.cc.

0.36.4.3.5 unsigned int FRsdk::CaptureDevice::exposure ( ) const [inline]

return the current value of the exposure time

Examples:

tracklife.cc.

0.36.4.3.6 virtual std::pair<unsigned int, unsigned int> FRsdk::CaptureDevice::exposureRange ( ) const [inline], [virtual]

return the exposure range (min,max)

0.36.4.3.7 unsigned int FRsdk::CaptureDevice::gain ( ) const [inline]

return the current value of the gain

Examples:

tracklife.cc.

0.36.4.3.8 virtual std::pair<unsigned int, unsigned int> FRsdk::CaptureDevice::gainRange ( ) const [inline], [virtual]

return the gain range (min,max)

0.36.4.3.9 VideoFormat FRsdk::CaptureDevice::getVideoFormat ( ) const [inline]

retrieve current video format

0.36.4.3.10 Image FRsdk::CaptureDevice::release ( ) const [inline]

release an still image/single shot, beside the continuouse capturing of images some concrete CaptureDevice supports releasing a snapshot.

Mostly it supports compared to capture() addional settings and triggering of flash lights or similar.

0.36.4.3.11 virtual void FRsdk::CaptureDevice::setExposure (unsigned int *newExposure*) [inline], [virtual]

set the exposure time to newExposure

0.36.4.3.12 virtual void FRsdk::CaptureDevice::setGain (unsigned int *newGain*) [inline], [virtual]

set the gain to newGain

0.36.4.3.13 bool FRsdk::CaptureDevice::setVideoFormat (const VideoFormat & *f*) [inline]

apply a video format.

Even if 'true' is returned, the device still might fail to switch to this format, keeping the current one instead.

0.36.4.3.14 bool FRsdk::CaptureDevice::videoFormatDialog ( ) const [inline]

call up video format dialog provided by the device driver

The documentation for this class was generated from the following file:

- frsdk/capdev.h

0.36.5 FRsdk::CaptureDeviceBody Class Reference

Abstract capture device body.

`#include <capdev.h>`

Public Member Functions

- virtual ~CaptureDeviceBody ()
- virtual Image capture () const =0

- virtual Image release () const
- virtual unsigned int gain () const
- virtual std::pair< unsigned int, unsigned int > gainRange () const
- virtual void setGain (unsigned int)
- virtual unsigned int exposure () const
- virtual std::pair< unsigned int, unsigned int > exposureRange () const
- virtual void setExposure (unsigned int)
- virtual void control (const FRsdk::Image &, const std::list< Face::Location > &)
- virtual bool configure () const
- virtual bool videoFormatDialog () const
- virtual VideoFormat getVideoFormat () const
- virtual bool setVideoFormat (const VideoFormat &)

0.36.5.1 Detailed Description

Abstract capture device body.

For a detailed description of the memeber function see documentation of CaptureDevice

0.36.5.2 Constructor & Destructor Documentation

0.36.5.2.1 virtual FRsdk::CaptureDeviceBody::∼CaptureDeviceBody ( ) [inline], [virtual]

0.36.5.3 Member Function Documentation

0.36.5.3.1 virtual Image FRsdk::CaptureDeviceBody::capture ( ) const [pure virtual]

0.36.5.3.2 virtual bool FRsdk::CaptureDeviceBody::configure ( ) const [inline], [virtual]

0.36.5.3.3 virtual void FRsdk::CaptureDeviceBody::control (const FRsdk::Image & , const std::list< Face::Location > &) [inline], [virtual]

0.36.5.3.4 virtual unsigned int FRsdk::CaptureDeviceBody::exposure ( ) const [inline], [virtual]

0.36.5.3.5 virtual std::pair<unsigned int, unsigned int> FRsdk::CaptureDeviceBody::exposureRange ( ) const [inline], [virtual]

0.36.5.3.6 virtual unsigned int FRsdk::CaptureDeviceBody::gain ( ) const [inline], [virtual]

0.36.5.3.7 virtual std::pair<unsigned int, unsigned int> FRsdk::CaptureDeviceBody::gainRange ( ) const [inline], [virtual]

0.36.5.3.8 virtual VideoFormat FRsdk::CaptureDeviceBody::getVideoFormat ( ) const [inline], [virtual]

0.36.5.3.9 virtual Image FRsdk::CaptureDeviceBody::release ( ) const [inline], [virtual]

0.36.5.3.10 virtual void FRsdk::CaptureDeviceBody::setExposure (unsigned *int*) [inline], [virtual]

0.36.5.3.11 virtual void FRsdk::CaptureDeviceBody::setGain (unsigned *int*) [inline], [virtual]

0.36.5.3.12 virtual bool FRsdk::CaptureDeviceBody::setVideoFormat (const VideoFormat &) [inline], [virtual]

0.36.5.3.13 virtual bool FRsdk::CaptureDeviceBody::videoFormatDialog ( ) const [inline], [virtual]

The documentation for this class was generated from the following file:

- frsdk/capdev.h

0.36.6 FRsdk::Portrait::Characteristics Class Reference

Portrait Characteristics.

`#include <portrait.h>`

Public Member Functions

- Characteristics (const Characteristics &)
- Characteristics & operator= (const Characteristics &)
- ∼Characteristics ()
- bool isColor () const

  returns true if characteristics are received by processing the image as a real color image.
- unsigned int width () const

  The width of the portrait image in pixels.
- unsigned int height () const

  The height of the portrait image in pixels.
- Position eye0 () const

  Coordinate of Feature Point 12.2 (Right eye center).
- Position eye1 () const

  Coordinate of Feature Point 12.1 (Left eye center).
- float eyeDistance () const

  Get the eye distance in pixels.
- Position faceCenter () const

  Coordinate of the center of the line connecting Feature Points 12.1 and 12.2 (Center of left and right eye) See ISO 19794-5:2005 section 5.6.4.
- unsigned int numberOfFaces () const

  Try to detect all faces within the annotated image, ignores given position.
- float glasses () const

  Returns a measure for the probability of the person in the portrait to wear glasses See ISO 19794-5:2005 appendix A.3.2.4.
- float eye0Open () const

  Returns the confidence for the person's eyes beeing open.
- float eye1Open () const
- float eye0GazeFrontal () const

  Returns the confidence for the person's eyes looking frontal to the camera.
- float eye1GazeFrontal () const
- float eye0Red () const

  Returns the redness of eyes pupils.
- float eye1Red () const
- float eye0Tinted () const

  Returns a value how tinted the areas around eyes are.
- float eye1Tinted () const
- float exposure () const

  Returns average gray value within facial region.
- unsigned int grayScaleDensity () const

  Gray scale density (number of different gray values) within facial region.
- float naturalSkinColour () const

  Returns the natural colours ratio (0.0 - 1.0) within face region.
- float hotSpots () const

Returns the hot spot amount (greater or equal to 0.0) within face region.

- float backgroundUniformity () const

  Background ist not normativ according to ISO 19794-5:2005 section 7.2.6, but according to appendix A 2.4.3 the background uniformity is tested by this function.

- float widthOfHead () const

  Horizontal distance between the points where the external ear connects the head in pixels.

- float lengthOfHead () const

  Vertical distance between base of the chin and the crown in pixels.

- float poseAngleRoll () const

  Returns the tangent of the Pose Angle - Roll.

- float chin () const

  Returns the estimated distance (in pixel) between base of chin line and the eyes connecting line.

- float crown () const

  Returns the estimated distance (in pixel) between crown line and the eyes connecting line.

- float ear0 () const

  Returns the estimated distance (in pixel) between the center of the face and the left bounding line of the face reagion marked by the coordinate point 10.10 (left ear to head connection, see ISO 19794-5:2005 section 5.6.3.).

- float ear1 () const

  Returns the estimated distance (in pixel) between the center of the face and the right bounding line of the face reagion marked by the coordinate point 10.9 (right ear to head connection, see ISO ISO 19794-5:2005 section 5.6.3.).

- float deviationFromFrontalPose () const

  Returns a measure for the deviation from frontal pose.

- float isMale () const

  Returns a measure for the probability that the image contains a portrait of a male person.

- unsigned int age () const

  returns the estimated age in years

- float deviationFromUniformLighting () const

  Returns a measure for the deviation from uniform lighting in the face area.

- float sharpness () const

  Returns a measure for focus and depth of field according to specification of ISO 19794-5:2005 section 7.3.3.

- float mouthClosed () const

  Returns the confidence for the person's mouth beeing closed.

- EthnicityMeasurements ethnicity () const

  returns the measurements for the ethnicity of the person

Friends

- struct Implementation
- class Analyzer

0.36.6.1 Detailed Description

Portrait Characteristics.

An instance of this class is produced by analyzing a face portrait using Portrait::Analyzer. It provides various measures important for determining compliance with ISO 19794-5:2005.

The following sketches describes some names used in documentation.

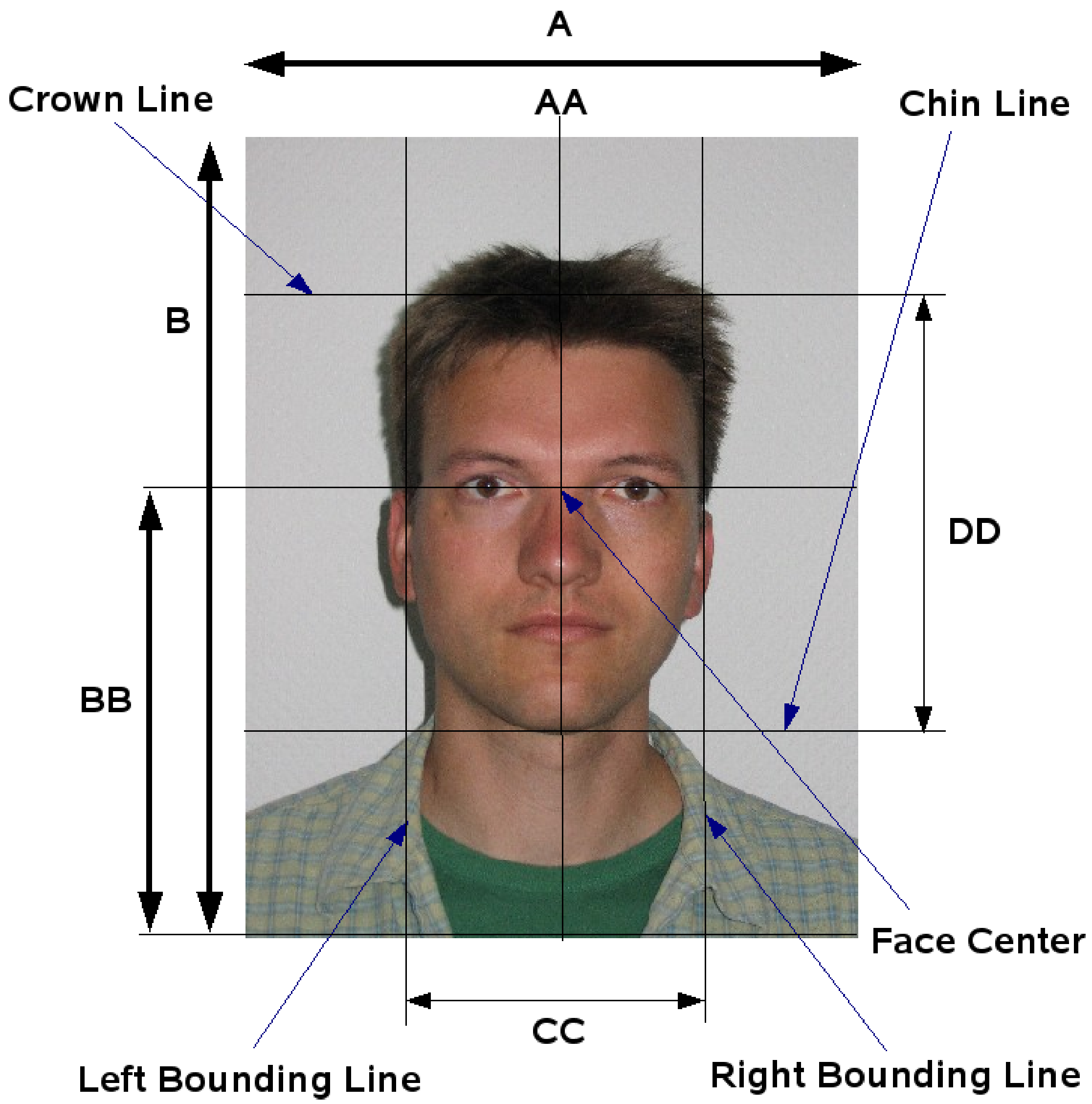


Figure 13: ISO face geometry

- **A** image width
- **AA** horizotal center of the face
- **B** image height
- **BB** distance of the center of the face from the bottom edge
- **CC** width of head
- **DD** length of head
- **Chin Line** line parallel to eyes connecting line
- **Crown Line** line parallel to eyes connecting line
- **Left Bounding Line** line perpendicular to eyes connecting line
- **Right Bounding Line** line perpendicular to eyes connecting line
- **Face Center** Center of the face, middle point of the eyes connecting line

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.2 Constructor & Destructor Documentation

0.36.6.2.1 FRsdk::Portrait::Characteristics::Characteristics (const Characteristics &)

0.36.6.2.2 FRsdk::Portrait::Characteristics::∼Characteristics ( )

0.36.6.3 Member Function Documentation

0.36.6.3.1 unsigned int FRsdk::Portrait::Characteristics::age ( ) const

returns the estimated age in years

Examples:

acquisition.cc.

0.36.6.3.2 float FRsdk::Portrait::Characteristics::backgroundUniformity ( ) const

Background ist not normativ according to ISO 19794-5:2005 section 7.2.6, but according to appendix A 2.4.3 the background uniformity is tested by this function.

Examples:

acquisition.cc.

0.36.6.3.3 float FRsdk::Portrait::Characteristics::chin ( ) const

Returns the estimated distance (in pixel) between base of chin line and the eyes connecting line.

Chin is definded as the central forward portion of the lower jaw (see ISO 19794-5:2005 section 4.1.) In context of face recognition it is a part to determine the face region. So, chin can be seen as a line representing the lower limit of the face region. The eyes connecting line is defined by the eye positions and the chin line is parallel to the eyes connecting line.

Examples:

acquisition.cc.

0.36.6.3.4 float FRsdk::Portrait::Characteristics::crown ( ) const

Returns the estimated distance (in pixel) between crown line and the eyes connecting line.

Crown is definded as the top head if it could be seen (see ISO 19794-5:2005 section 4.6.) In context of face recognition it is a part to determine the face region. So, crown can be seen as line representing the upper limit of the face region. The eyes connecting line is defined by the eye positions and the crown line is parallel to the eyes connecting line.

Examples:

acquisition.cc.

0.36.6.3.5 float FRsdk::Portrait::Characteristics::deviationFromFrontalPose ( ) const

Returns a measure for the deviation from frontal pose.

Higher values mean larger deviation from frontal pose (yaw and pitch). See ISO 19794-5:2005 section 7.2.2.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.6 float FRsdk::Portrait::Characteristics::deviationFromUniformLighting ( ) const

Returns a measure for the deviation from uniform lighting in the face area.

Higher absolute values mean higher deviation. The implementation uses a trained function that maps the face region of a face image to a score and that should return a lower score for face with uniform lighting than for a face with non- uniform lighting. See ISO 19794-5:2005 section 7.2.7.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.7 float FRsdk::Portrait::Characteristics::ear0 ( ) const

Returns the estimated distance (in pixel) between the center of the face and the left bounding line of the face reagion marked by the coordinate point 10.10 (left ear to head connection, see ISO 19794-5:2005 section 5.6.3.).

The left bounding line is perpendicular to the eyes connecting line.

Examples:

acquisition.cc.

0.36.6.3.8 float FRsdk::Portrait::Characteristics::ear1 ( ) const

Returns the estimated distance (in pixel) between the center of the face and the right bounding line of the face reagion marked by the coordinate point 10.9 (right ear to head connection, see ISO ISO 19794-5:2005 section 5.6.3.).

The right bounding line is perpendicular to the eyes connecting line.

Examples:

acquisition.cc.

0.36.6.3.9 EthnicityMeasurements FRsdk::Portrait::Characteristics::ethnicity ( ) const

returns the measurements for the ethnicity of the person

0.36.6.3.10 float FRsdk::Portrait::Characteristics::exposure ( ) const

Returns average gray value within facial region.

Examples:

acquisition.cc.

0.36.6.3.11 Position FRsdk::Portrait::Characteristics::eye0 ( ) const

Coordinate of Feature Point 12.2 (Right eye center).

See ISO 19794-5:2005 section 5.6.4.

Examples:

acquisition.cc.

0.36.6.3.12 float FRsdk::Portrait::Characteristics::eye0GazeFrontal ( ) const

Returns the confidence for the person's eyes looking frontal to the camera.

The higher the returned value is, the more frontal the gaze is. See ISO 19794-5:2005 section 7.2.3.

Examples:

acquisition.cc.

0.36.6.3.13 float FRsdk::Portrait::Characteristics::eye0Open ( ) const

Returns the confidence for the person's eyes beeing open.

Higher values mean a higher confidence. See ISO 19794-5:2005 section 7.2.3.

Examples:

acquisition.cc.

0.36.6.3.14 float FRsdk::Portrait::Characteristics::eye0Red ( ) const

Returns the redness of eyes pupils.

See ISO 19794-5:2005 section 7.3.4.

Examples:

acquisition.cc.

0.36.6.3.15 float FRsdk::Portrait::Characteristics::eye0Tinted ( ) const

Returns a value how tinted the areas around eyes are.

Higher Value means more tinted. It is only correctly intepretable when the glasses detection is positiv. See ISO 19794-5:2005 section 7.2.11.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.16 Position FRsdk::Portrait::Characteristics::eye1 ( ) const

Coordinate of Feature Point 12.1 (Left eye center).

See ISO 19794-5:2005 section 5.6.4.

Examples:

acquisition.cc.

0.36.6.3.17 float FRsdk::Portrait::Characteristics::eye1GazeFrontal ( ) const

Examples:

acquisition.cc.

0.36.6.3.18 float FRsdk::Portrait::Characteristics::eye1Open ( ) const

Examples:

acquisition.cc.

0.36.6.3.19 float FRsdk::Portrait::Characteristics::eye1Red ( ) const

Examples:

acquisition.cc.

0.36.6.3.20 float FRsdk::Portrait::Characteristics::eye1Tinted ( ) const

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.21 float FRsdk::Portrait::Characteristics::eyeDistance ( ) const

Get the eye distance in pixels.

Examples:

acquisition.cc.

0.36.6.3.22 Position FRsdk::Portrait::Characteristics::faceCenter ( ) const

Coordinate of the center of the line connecting Feature Points 12.1 and 12.2 (Center of left and right eye) See ISO 19794-5:2005 section 5.6.4.

Examples:

acquisition.cc.

0.36.6.3.23 float FRsdk::Portrait::Characteristics::glasses ( ) const

Returns a measure for the probability of the person in the portrait to wear glasses See ISO 19794-5:2005 appendix A.3.2.4.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.24 unsigned int FRsdk::Portrait::Characteristics::grayScaleDensity ( ) const

Gray scale density (number of different gray values) within facial region.

The facial region used is the area enclosed by the 2 semiellipses uniquely defined by crown, ear0, ear1 and chin, ear0, ear1, respectively. See ISO 19794-5:2005 section 7.4.2.1

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.25 unsigned int FRsdk::Portrait::Characteristics::height ( ) const

The height of the portrait image in pixels.

Examples:

acquisition.cc.

0.36.6.3.26 float FRsdk::Portrait::Characteristics::hotSpots ( ) const

Returns the hot spot amount (greater or equal to 0.0) within face region.

For details refer to ISO 19794-5:2005 sections 7.2.10 and 7.2.11. Setting the threshold to 0.5 rejects all images with hot spots. Setting the threshold to a value between 1 and 100 causes images with small reflections to be accepted. In images with visible hot spots the meassured value can reach up to 12000.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.27 bool FRsdk::Portrait::Characteristics::isColor ( ) const

returns true if characteristics are received by processing the image as a real color image.

Else the image is processed as a intensity based image and all test regarded to color information are skipped.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.28 float FRsdk::Portrait::Characteristics::isMale ( ) const

Returns a measure for the probability that the image contains a portrait of a male person.

Examples:

acquisition.cc.

0.36.6.3.29 float FRsdk::Portrait::Characteristics::lengthOfHead ( ) const

Vertical distance between base of the chin and the crown in pixels.

See ISO 19794-5:2005 section 8.3.5.

Examples:

acquisition.cc.

0.36.6.3.30 float FRsdk::Portrait::Characteristics::mouthClosed ( ) const

Returns the confidence for the person's mouth beeing closed.

Higher values mean a higher confidence. See ISO 19794-5:2005 section 7.2.3

Examples:

acquisition.cc.

0.36.6.3.31 float FRsdk::Portrait::Characteristics::naturalSkinColour ( ) const

Returns the natural colours ratio (0.0 - 1.0) within face region.

For details refer to ISO 19794-5:2005 section 7.3.4.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.32 unsigned int FRsdk::Portrait::Characteristics::numberOfFaces ( ) const

Try to detect all faces within the annotated image, ignores given position.

Returns number of found faces.

Examples:

acquisition.cc.

0.36.6.3.33 Characteristics& FRsdk::Portrait::Characteristics::operator= (const Characteristics &)

0.36.6.3.34 float FRsdk::Portrait::Characteristics::poseAngleRoll ( ) const

Returns the tangent of the Pose Angle - Roll.

This is the rotation about the horizontal axis from front to back. See ISO 19794-5:2005 sections 5.5.8.3 and 7.2.2.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.35 float FRsdk::Portrait::Characteristics::sharpness ( ) const

Returns a measure for focus and depth of field according to specification of ISO 19794-5:2005 section 7.3.3.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.6.3.36 unsigned int FRsdk::Portrait::Characteristics::width ( ) const

The width of the portrait image in pixels.

Examples:

acquisition.cc.

0.36.6.3.37 float FRsdk::Portrait::Characteristics::widthOfHead ( ) const

Horizontal distance between the points where the external ear connects the head in pixels.

See ISO 19794-5:2005 section 8.3.4.

Examples:

acquisition.cc.

0.36.6.4 Friends And Related Function Documentation

0.36.6.4.1 friend class Analyzer [friend]

0.36.6.4.2 friend struct Implementation [friend]

The documentation for this class was generated from the following file:

- frsdk/portrait.h

0.36.7 FRsdk::ISO_19794_5::FullFrontal::Compliance Class Reference

Compliance assessment results.

```
#include <portraittests.h>
```

Public Member Functions

- Compliance (const Compliance &)
- Compliance & operator= (const Compliance &)
- ~Compliance ()
- bool onlyOneFaceVisible () const

  Only one face has to be visible in the image according to ISO 19794-5:2005 section 7.2.4.
- bool goodVerticalFacePosition () const

  Test the vertical position of the face.
- bool horizontallyCenteredFace () const

  Test whether the face is centered in the image.
- bool widthOfHead () const

  Width of the head compared to image width.
- bool widthOfHeadBestPractice () const

  According to section A 3.2.2 best practice is a range of image width to face width ratio between 1.4 and 2.0.
- bool lengthOfHead () const

  Length of head is limited to the range of 60% to 90% of the image height.
- bool lengthOfHeadBestPractice () const

  Best practice reduces the range of face length to 70% to 80% of the image height.
- bool resolution () const

  Resolution of the full images shall be at least 180 pixels for the width of the head or 90 pixels from eye center to eye center (see ISO standard 8.4.1).
- bool resolutionBestPractice () const

  Best Practice recommendation are more strict.
- bool imageWidthToHeightBestPractice () const

  Paragraph A3.2.1 of ISO 19794_5 describes a best practice of ratio between image height and width.
- bool goodExposure () const

  'True' means there is no over or under exposure.
- bool goodGrayScaleProfile () const

  According to ISO 19794_5 sections 7.4.2.1 and 7.4.2.2 'True' will be returned only if the face area has a intensity resulotion of at least 7 bits (128 intensity values).
- bool hasNaturalSkinColour () const

  Natural colours in face region Returns true if the face region has natural colors, otherwise false.
- bool noHotSpots () const

  Hot Spots (bright areas of light reflected from the face).
- bool isBackgroundUniformBestPractice () const

  background uniformity.
- bool isFrontal () const

  The face is considered frontal if the rotation of the head is less than +/-5 degrees from frontal for yaw and pitch and if roll angle of head is less then +/-8 degrees.
- bool isFrontalBestPractice () const

  The face is considered frontal if the rotation of the head is less than +/-5 degrees from frontal in every direction (roll, pitch and yaw).
- bool isLightingUniform () const

  Returns true if lighting is equally distributed in the face area.
- bool eyesOpenBestPractice () const

  Returns true if the both eyes of the person are open.
- bool eyesGazeFrontalBestPractice () const

  Returns true if the person's eyes are looking frontal to the camera.
- bool eyesNotRedBestPractice () const

  returns true if both eyes pupils are not detected as red.
- bool noTintedGlasses () const

according to ISO 19794-5:2005 section 7.2.11 and best recommendations glasses should not be tinted.

- bool isSharp () const

  returns true if the face area (from chin to crown and from left to right ear) fits the focus and depth in field characteristics (see ISO 19794-5:2005 section 7.3.3).
- bool mouthClosedBestPractice () const

  returns true if mouth is closed according to ISO 19794-5:2005 section 7.2.3 and appendix A 2.2.1
- bool isCompliant () const

  Returns true if the images is compliant with the ISO 19794-5:2005 requirements only.
- bool isBestPractice () const

  The test contains is Compliant and additionally all checks according to best practice represented by function names of this class with 'BestPractice' in name.

Friends

- class Test

0.36.7.1 Detailed Description

Compliance assessment results.

Instances of this class represent the compliance of portraits with the Full Frontal Image requirements. They can be produced using instances of FullFrontal::Test.

Compliance to ISO 19794-5:2005 mean the image is compliant to the specification of the document wihtout recommendations described in annex. The Annex contains only recommendations of best practice.

So assessment results contains of two different compliance test. All tests based on best practice recommendation have a post fix 'BestPractice'. All other are based only on test for mandatory features.

So if only mandatory features should be tested only isCompliant() should be called. isBestPractice() shoud be called if mandatory features and best practice recommendations should be tested.

All tests and feature are according to ISO 19794-5:2005 and the Technical Corrigendum 3 from February 2008.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.2 Constructor & Destructor Documentation

0.36.7.2.1 FRsdk::ISO_19794_5::FullFrontal::Compliance::Compliance (const Compliance &)

0.36.7.2.2 FRsdk::ISO_19794_5::FullFrontal::Compliance::∼Compliance ( )

0.36.7.3 Member Function Documentation

0.36.7.3.1 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::eyesGazeFrontalBestPractice ( ) const

Returns true if the person's eyes are looking frontal to the camera.

See ISO 19794-5:2005 section 7.2.3 and appendix A 2.2.1.

Examples:

acquisition.cc.

0.36.7.3.2 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::eyesNotRedBestPractice ( ) const

returns true if both eyes pupils are not detected as red.

Eyes are checkend independendly and the result is a logical And of both single checks. See ISO 19794-5:2005 section 7.3.4 and appendix A 2.2.1.

Examples:

acquisition.cc.

0.36.7.3.3 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::eyesOpenBestPractice ( ) const

Returns true if the both eyes of the person are open.

Eyes are checkend independendly and the result is a logical And of both single checks. See ISO 19794-5:2005 section 7.2.3 and appendix A 2.2.1.

Examples:

acquisition.cc.

0.36.7.3.4 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::goodExposure ( ) const

'True' means there is no over or under exposure.

See ISO 19794_5 section 7.3.2.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.5 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::goodGrayScaleProfile ( ) const

According to ISO 19794_5 sections 7.4.2.1 and 7.4.2.2 'True' will be returned only if the face area has a intensity resulotion of at least 7 bits (128 intensity values).

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.6 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::goodVerticalFacePosition ( ) const

Test the vertical position of the face.

0.5 image height $<$ eyes line $<$ 0.7 image height. See ISO 19794-5:2005 section 8.3.3.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.7 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::hasNaturalSkinColour ( ) const

Natural colours in face region Returns true if the face region has natural colors, otherwise false.

See ISO 19794_5 section 7.3.4

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.8 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::horizontallyCenteredFace ( ) const

Test whether the face is centered in the image.

The center should be between 45% and 55% of the image width. See ISO 19794-5:2005 section 8.3.2

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.9 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::imageWidthToHeightBestPractice ( ) const

Paragraph A3.2.1 of ISO 19794_5 describes a best practice of ratio between image height and width.

It should be between 1.25 and 1.34.

Examples:

acquisition.cc.

0.36.7.3.10 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::isBackgroundUniformBestPractice ( ) const

background uniformity.

returns true if the background is uniform. See ISO 19794_5 appendix A 2.4.3

Examples:

acquisition.cc.

0.36.7.3.11 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::isBestPractice ( ) const

The test contains is Compliant and additionally all checks according to best practice represented by function names of this class with 'BestPractice' in name.

True is return incase of all checks are passed, else false is returned.

Examples:

acquisition.cc.

0.36.7.3.12 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::isCompliant ( ) const

Returns true if the images is compliant with the ISO 19794-5:2005 requirements only.

If it failes only member function without 'BestPractice' in name must be checked in order to get the reason why the test failes.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.13 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::isFrontal ( ) const

The face is considered frontal if the rotation of the head is less than +/-5 degrees from frontal for yaw and pitch and if roll angle of head is less then +/-8 degrees.

See ISO 19794-5:2005 section 7.2.2.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.14 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::isFrontalBestPractice ( ) const

The face is considered frontal if the rotation of the head is less than +/-5 degrees from frontal in every direction (roll, pitch and yaw).

See ISO 19794-5:2005 section 7.2.2 and appendix A 2.2.

Examples:

acquisition.cc.

0.36.7.3.15 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::isLightingUniform ( ) const

Returns true if lighting is equally distributed in the face area.

That means that there is no significant direction of the light from the point of view of the photographer. See ISO standard 7.2.7

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.16 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::isSharp ( ) const

returns true if the face area (from chin to crown and from left to right ear) fits the focus and depth in field characteristics (see ISO 19794-5:2005 section 7.3.3).

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.17 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::lengthOfHead ( ) const

Length of head is limited to the range of 60% to 90% of the image height.

See ISO 19794-5:2005 section 8.3.5.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.18 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::lengthOfHeadBestPractice ( ) const

Best practice reduces the range of face length to 70% to 80% of the image height.

See ISO 19794-5:2005 appendix A 3.2.3.

Examples:

acquisition.cc.

0.36.7.3.19 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::mouthClosedBestPractice ( ) const

returns true if mouth is closed according to ISO 19794-5:2005 section 7.2.3 and appendix A 2.2.1

Examples:

acquisition.cc.

0.36.7.3.20 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::noHotSpots ( ) const

Hot Spots (bright areas of light reflected from the face).

Refer to ISO 19794_5 section 7.2.10 and 7.2.11.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.21 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::noTintedGlasses ( ) const

according to ISO 19794-5:2005 section 7.2.11 and best recommendations glasses should not be tinted.

noTintedGlasses() returns true if the person either wears no glasses or glasses are not tinted.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.22 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::onlyOneFaceVisible ( ) const

Only one face has to be visible in the image according to ISO 19794-5:2005 section 7.2.4.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.23 Compliance& FRsdk::ISO_19794_5::FullFrontal::Compliance::operator= (const Compliance &)

0.36.7.3.24 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::resolution ( ) const

Resolution of the full images shall be at least 180 pixels for the width of the head or 90 pixels from eye center to eye center (see ISO standard 8.4.1).

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.25 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::resolutionBestPractice ( ) const

Best Practice recommendation are more strict.

A face should be 240 pixel in width (roughly 120 Pixel eye to eye distance, see ISO standard A3.1.1).

Examples:

acquisition.cc.

0.36.7.3.26 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::widthOfHead ( ) const

Width of the head compared to image width.

It should be between 50% and 75% of the image width. See ISO 19794-5:2005 section 8.3.4.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.7.3.27 bool FRsdk::ISO_19794_5::FullFrontal::Compliance::widthOfHeadBestPractice ( ) const

According to section A 3.2.2 best practice is a range of image width to face width ratio between 1.4 and 2.0.

Examples:

acquisition.cc.

0.36.7.4 Friends And Related Function Documentation

0.36.7.4.1 friend class Test [friend]

The documentation for this class was generated from the following file:

- frsdk/portraittests.h

0.36.8 FRsdk::Configuration Class Reference

Configuration object of the FaceVACS SDK library.

```
#include <config.h>
```

Public Member Functions

- Configuration (const std::string &configFilename)

  Construct Configuration object from given configuration file.
- Configuration (std::istream &is)

  Construct Configuration object from given stream.
- Configuration (const Configuration &)
- Configuration & operator= (const Configuration &)
- ~Configuration ()
- void computerId (std::ostream &) const

  License activation: save local computer id to ostream.
- void activateLicense (std::istream &istr)

  Activate a license using activation information.
- std::string licenseInformation () const

  returns a string which contains the license information
- std::string version () const

  returns a string which contains the version of the current used library
- std::string getValue (const std::string &key) const

  Configuration item value access, returns the string representation of the configuration item's value, the key has to be a valid configuration item name in dotted notation, e.g.
- void setValue (const std::string &key, const std::string &value)

  Set configuration item to given value, the key has to be a valid configuration item name, the value has to the string representation of the item's value, e.g.
- void resetToDefault (const std::string &key)

  Reset the configuration item to the default value.
- std::list< std::pair
  < std::string, std::string > > protectedItems () const

  returns a list of configuration items (key/value pairs) which are set via the license signature, first of pair is config key, second is the value.

Friends

- class Implementation

0.36.8.1 Detailed Description

Configuration object of the FaceVACS SDK library.

It is recommended that only one instance of a Configuration per Application is instatiated.

MT-unsafe:

concurrent creation of this class might fail on some plattforms due to some limitations of C++ compiler regarding function-local static variables. Also all modifying member functions are not thread safe.

Examples:

acquisition.cc, capdev.cc, cropfullfrontal.cc, enroll.cc, eyesfind.cc, facefind.cc, identify.cc, match.cc, tracklife.cc, trackrec.cc, verify.cc, verifyan.cc, and vignetting.cc.

0.36.8.2 Constructor & Destructor Documentation

0.36.8.2.1 FRsdk::Configuration::Configuration (const std::string & *configFilename*)

Construct Configuration object from given configuration file.

0.36.8.2.2 FRsdk::Configuration::Configuration (std::istream & *is*)

Construct Configuration object from given stream.

0.36.8.2.3 FRsdk::Configuration::Configuration (const Configuration &)

0.36.8.2.4 FRsdk::Configuration::∼Configuration ( )

0.36.8.3 Member Function Documentation

0.36.8.3.1 void FRsdk::Configuration::activateLicense (std::istream & *istr*)

Activate a license using activation information.

Reads license activation information from istream (e.g. from an activation file) and stores it into current configuration. In case of an invalid activation information the current configuration will be untouched and a LicenseSignatureMismatch exception is thrown.

0.36.8.3.2 void FRsdk::Configuration::computerId (std::ostream &) const

License activation: save local computer id to ostream.

0.36.8.3.3 std::string FRsdk::Configuration::getValue (const std::string & *key*) const

Configuration item value access, returns the string representation of the configuration item's value, the key has to be a valid configuration item name in dotted notation, e.g.

"FRSDK.ComparisonAlgorithm"

0.36.8.3.4 std::string FRsdk::Configuration::licenseInformation ( ) const

returns a string which contains the license information

0.36.8.3.5 Configuration& FRsdk::Configuration::operator= (const Configuration &)

0.36.8.3.6 std::list< std::pair< std::string, std::string> > FRsdk::Configuration::protectedItems ( ) const

returns a list of configuration items (key/value pairs) which are set via the license signature, first of pair is config key, second is the value.

Changing key or values from that list in the FaceVACS-SDK configuration file will cause throwing LicenseSignature-Mismatch exceptions. For details see FaceVACS-SDK Feature enabling/disabling, Limits.

0.36.8.3.7 void FRsdk::Configuration::resetToDefault (const std::string & *key*)

Reset the configuration item to the default value.

The key has to be a valid configuration item name. The default values can be inspected using the configuration editor

0.36.8.3.8 void FRsdk::Configuration::setValue (const std::string & *key,* const std::string & *value*)

Set configuration item to given value, the key has to be a valid configuration item name, the value has to the string representation of the item's value, e.g.

"B2ComparisonAlgorithm", "0.5" or "42". The changes will become persistent if the Configuration object was created from a configFilename. Otherwise the changes will apply to the current object only. Note that most classes use the configuration at object construction time only. In these cases later changes might have no effect.

0.36.8.3.9 std::string FRsdk::Configuration::version ( ) const

returns a string which contains the version of the current used library

0.36.8.4 Friends And Related Function Documentation

0.36.8.4.1 friend class Implementation `[friend]`

The documentation for this class was generated from the following file:

- frsdk/config.h

0.36.9 FRsdk::CountedPtr< T > Class Template Reference

This class template implements the reference counting idiom simulating name semantics (see Scott Meyer's Bible, James Coplien's book).

```
#include <cptr.h>
```

Public Member Functions

- CountedPtr (CountedObject *p=0)

  c'tor which takes the ownership of the object given pointer.
- CountedPtr (const CountedPtr &rhs)

  copy c'tor, increments the reference count
- template<class Derived >
  CountedPtr (const CountedPtr< Derived > &rhs)

  Templatized c'tor to support derived classes This feature is not available with MS Visual C++ 6.0 on Win32 platforms due to problems with explicit template instantiation.
- ~CountedPtr ()

  d'tor, which decrements the reference count to the counted object and deletes the object if the reference count is zero.
- CountedPtr< CountedObject > & operator= (const CountedPtr< CountedObject > &rhs)

assignment operator, which smartly decrements the reference count if the right hand side counted pointer holds a different object.

- bool operator== (const CountedPtr< CountedObject > &rhs) const

  comparison operator
- bool operator!= (const CountedPtr< CountedObject > &rhs) const

  comparison operator
- CountedObject & operator* () const

  access operator with reference semantics
- CountedObject * operator-> () const

  access operator with pointer semantics

Public Attributes

- CountedObject * co

  never touch this
- RefCountHandle refs

  never touch this

0.36.9.1 Detailed Description

template<class T>class FRsdk::CountedPtr< T >

This class template implements the reference counting idiom simulating name semantics (see Scott Meyer's Bible, James Coplien's book).

The public attributes of this class should be private but the supported compilers have problems with template friend declarations.

```
template <class Derived> friend class CountedPtr;
```

MT-safe (partly reentrant, partly serialized) It is safe to call

the member functions concurrently from different threads.

Examples:

acquisition.cc, edialog.h, and match.cc.

0.36.9.2 Constructor & Destructor Documentation

0.36.9.2.1 template<class T> FRsdk::CountedPtr< T >::CountedPtr (CountedObject * *p* = 0) [inline]

c'tor which takes the ownership of the object given pointer.

Note that only objects created dynamically must be passed to a CountedPtr's constructor.

0.36.9.2.2 template<class T> FRsdk::CountedPtr< T >::CountedPtr (const CountedPtr< T > & *rhs*) [inline]

copy c'tor, increments the reference count

0.36.9.2.3 template<class T> template<class Derived > FRsdk::CountedPtr< T >::CountedPtr (const CountedPtr< Derived > & *rhs*) [inline]

Templatized c'tor to support derived classes This feature is not available with MS Visual C++ 6.0 on Win32 platforms due to problems with explicit template instantiation.

0.36.9.2.4 template<class T> FRsdk::CountedPtr< T >::~CountedPtr ( ) [inline]

d'tor, which decrements the reference count to the counted object and deletes the object if the reference count is zero.

0.36.9.3 Member Function Documentation

0.36.9.3.1 template<class T> bool FRsdk::CountedPtr< T >::operator!= (const CountedPtr< CountedObject > & *rhs*) const [inline]

comparison operator

0.36.9.3.2 template<class T> CountedObject& FRsdk::CountedPtr< T >::operator* ( ) const [inline]

access operator with reference semantics

0.36.9.3.3 template<class T> CountedObject* FRsdk::CountedPtr< T >::operator-> ( ) const [inline]

access operator with pointer semantics

0.36.9.3.4 template<class T> CountedPtr<CountedObject>& FRsdk::CountedPtr< T >::operator= (const CountedPtr< CountedObject > & *rhs*) [inline]

assignment operator, which smartly decrements the reference count if the right hand side counted pointer holds a different object.

0.36.9.3.5 template<class T> bool FRsdk::CountedPtr< T >::operator== (const CountedPtr< CountedObject > & *rhs*) const [inline]

comparison operator

0.36.9.4 Member Data Documentation

0.36.9.4.1 template<class T> CountedObject* FRsdk::CountedPtr< T >::co

never touch this

Referenced by FRsdk::CountedPtr< FRsdk::CaptureDeviceBody >::operator*(), FRsdk::CountedPtr< FRsdk::-CaptureDeviceBody >::operator->(), FRsdk::CountedPtr< FRsdk::CaptureDeviceBody >::operator=(), and F-Rsdk::CountedPtr< FRsdk::CaptureDeviceBody >::operator==().

0.36.9.4.2 template<class T> RefCountHandle FRsdk::CountedPtr< T >::refs

never touch this

Referenced by FRsdk::CountedPtr< FRsdk::CaptureDeviceBody >::CountedPtr(), and FRsdk::CountedPtr< F-Rsdk::CaptureDeviceBody >::operator=().

The documentation for this class was generated from the following file:

- frsdk/cptr.h

0.36.10 FRsdk::ISO_19794_5::FullFrontal::Creator Class Reference

Extract a Full Frontal Image from the source image.

```
#include <fullfrontal.h>
```

Classes

- class PaddingRatioExceeded

  Exception of this type is thrown when the padding ratio of the to be extracted FullFrontal portrait exceeds the pre-configured threshold.

Public Member Functions

- Creator (const Configuration &)

  create an instance of class Creator
- Creator (const Creator &)
- Creator & operator= (const Creator &)
- ∼Creator ()
- AnnotatedImage extract (const AnnotatedImage &source)

  extract the Full Frontal Image from an annotated image.
- AnnotatedImage extract (const AnnotatedImage &source, float headLengthToImageHeightRatio, float verticalHeadToImagePositionRatio)

  extract the Full Frontal Image from an annotated image, creating the resulting image with the face length to image height ratio and relative vertical position ratio passed with the 2nd and 3rd argument.

0.36.10.1 Detailed Description

Extract a Full Frontal Image from the source image.

The extract functions meet the geometric requirements of ISO standard ISO_19794_5 section 8.3 and the geometric recommendations of section A.3.2.3 Table 17 (Summary of best practices for Full Frontal Images on travel documents). In addition, the face is rotated to have horizontally aligned eye positions.

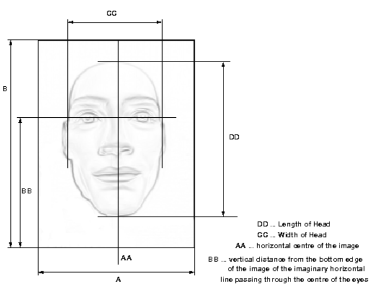


Figure 14: Geometric Characteristics of the Full Frontal Face Image

Image parameter	Required	Recommended
Vertical position of face	0.5∗B <= BB <= 0.7∗B	
Width of head	A >= 1.4∗CC	1.4∗CC <= A <= 2∗CC
Length of head	B >= 1.25∗DD	0.7∗B <= DD <= 0.8∗B
Height-to-width ratio of image		1.25 <= B/A <= 1.34

The resolution of the image (see section 8.4.1) is not modified by this function. This means that the input image is required to fit the Full Frontal Image resolution requirement (minimal resolution: 180 pixel head width or 90 pixel from eye center to eye center) in advance to be standard compliant. There is a configuration key setting the image height-to-width ratio of the target image. The default value is 45/35.

In some cases the source image is smaller than the required dimension of the target image. So some pixels of the resulting image have no corresponding pixel in source image and will be filled with a configurable padding color. Additionally the ratio of padded pixel to the full resulting image can be configured and will be tested during extraction. IF the ratio is exceeded an std::exception is thrown.

Examples:

cropfullfrontal.cc.

0.36.10.2 Constructor & Destructor Documentation

0.36.10.2.1 FRsdk::ISO_19794_5::FullFrontal::Creator::Creator (const Configuration &)

create an instance of class Creator

0.36.10.2.2 FRsdk::ISO_19794_5::FullFrontal::Creator::Creator (const Creator &)

0.36.10.2.3 FRsdk::ISO_19794_5::FullFrontal::Creator::∼Creator ( )

0.36.10.3 Member Function Documentation

0.36.10.3.1 AnnotatedImage FRsdk::ISO_19794_5::FullFrontal::Creator::extract (const AnnotatedImage & *source*)

extract the Full Frontal Image from an annotated image.

0.36.10.3.2 AnnotatedImage FRsdk::ISO_19794_5::FullFrontal::Creator::extract (const AnnotatedImage & *source,* float *headLengthToImageHeightRatio,* float *verticalHeadToImagePositionRatio*)

extract the Full Frontal Image from an annotated image, creating the resulting image with the face length to image height ratio and relative vertical position ratio passed with the 2nd and 3rd argument.

0.36.10.3.3 Creator& FRsdk::ISO_19794_5::FullFrontal::Creator::operator= (const Creator &)

The documentation for this class was generated from the following file:

- frsdk/fullfrontal.h

0.36.11 FRsdk::ISO_19794_5::TokenFace::Creator Class Reference

Extract a Token Face Image from the source image.

```
#include <tokenface.h>
```

Classes

- class PaddingRatioExceeded

    Exception of this type is thrown when the padding ratio of the to be extracted TokenFace exceeds the pre-configured threshold.

Public Member Functions

- Creator (const Configuration &)

    create an instance of class Creator
- Creator (const Creator &)
- Creator & operator= (const Creator &)
- ∼Creator ()
- AnnotatedImage extract (const AnnotatedImage &source)

    Extract a Token Face Image from the Full Frontal Image source meeting the geometric requirements ISO_19794_5 9.2.2 and 9.2.3.
- AnnotatedImage extractMinimal (const AnnotatedImage &source)

    Extract a Minimum width Token Face Image from the Full Frontal Image source meeting the geometric requirements ISO_19794_5 9.2.2, 9.2.3 and additionally 9.2.4.

0.36.11.1 Detailed Description

Extract a Token Face Image from the source image.

The extract functions meet the geometric requirements of ISO standard ISO_19794_5 section 9.2.3.

In some cases the source image is smaller than the required dimension of the target image (see Appendix A 4.-3). So some pixels of the resulting image have no corresponding pixel in source image and will be filled with a configurable padding color. Additionally the ratio of padded pixel to the full resulting image can be configured and will be tested during extraction. If the ratio is exceeded an std::exception is thrown.

Examples:

acquisition.cc.

0.36.11.2 Constructor & Destructor Documentation

0.36.11.2.1 FRsdk::ISO_19794_5::TokenFace::Creator::Creator (const **Configuration** &)

create an instance of class Creator

0.36.11.2.2 FRsdk::ISO_19794_5::TokenFace::Creator::Creator (const **Creator** &)

0.36.11.2.3 FRsdk::ISO_19794_5::TokenFace::Creator::~Creator ( )

0.36.11.3 Member Function Documentation

0.36.11.3.1 **AnnotatedImage** FRsdk::ISO_19794_5::TokenFace::Creator::extract (const **AnnotatedImage** & *source*)

Extract a Token Face Image from the Full Frontal Image source meeting the geometric requirements ISO_19794_5 9.2.2 and 9.2.3.

The original eye distance D is preserved. Token Face Image: Width: 4 ∗ D Width/Height: 0.75 Eyes : (0.375 ∗ Width - 1, 0.6 ∗ Width), (0.625 ∗ Width - 1, 0.6 ∗ Width) The caller is responsible for source being compliant with the Full Frontal Image Type requirements.

If during extraction of a token face the padding ratio of the resulting image exceeds the pre-configured threshold than a PaddingRatioExceeded exception is thrown.

MT-safe (reentrant) It is safe to call this function concurrently from different threads.

0.36.11.3.2 **AnnotatedImage** FRsdk::ISO_19794_5::TokenFace::Creator::extractMinimal (const **AnnotatedImage** & *source*)

Extract a Minimum width Token Face Image from the Full Frontal Image source meeting the geometric requirements ISO_19794_5 9.2.2, 9.2.3 and additionally 9.2.4.

Minimal Token Face Image: Width: 240 Height: 320 Eyes : (90.5, 144.5), (149.5, 144.5) The caller is responsible for source being compliant with the Full Frontal Image Type requirements.

If during extraction of a token face the padding ratio of the resulting image exceeds the pre-configured threshold than a PaddingRatioExceeded exception is thrown.

MT-safe (reentrant) It is safe to call this function concurrently from different threads.

0.36.11.3.3 **Creator**& FRsdk::ISO_19794_5::TokenFace::Creator::operator= (const **Creator** &)

The documentation for this class was generated from the following file:

- frsdk/tokenface.h

0.36.12 FRsdk::Portrait::EthnicityMeasurements Struct Reference

measurements for ethnicity detection, contains the probability that a person belongs to the ethnicity class

`#include <portrait.h>`

Public Member Functions

- EthnicityMeasurements (float w, float b, float a)

Public Attributes

- const float white
- const float black
- const float asian

0.36.12.1 Detailed Description

measurements for ethnicity detection, contains the probability that a person belongs to the ethnicity class

0.36.12.2 Constructor & Destructor Documentation

0.36.12.2.1 FRsdk::Portrait::EthnicityMeasurements::EthnicityMeasurements (float *w*, float *b*, float *a*) `[inline]`

0.36.12.3 Member Data Documentation

0.36.12.3.1 const float FRsdk::Portrait::EthnicityMeasurements::asian

0.36.12.3.2 const float FRsdk::Portrait::EthnicityMeasurements::black

0.36.12.3.3 const float FRsdk::Portrait::EthnicityMeasurements::white

The documentation for this struct was generated from the following file:

- frsdk/portrait.h

0.36.13 FRsdk::FacialMatchingEngine Class Reference

Low level facial comparison facility.

`#include <match.h>`

Public Member Functions

- FacialMatchingEngine (const Configuration &)
- FacialMatchingEngine (const FacialMatchingEngine &)
- FacialMatchingEngine & operator= (const FacialMatchingEngine &)
- ∼FacialMatchingEngine ()
- Score compare (const FIR &firA, const FIR &firB) const

  Calculate the score between firA and firB.
- CountedPtr< Scores > compare (const FIR &fir, const Population &population) const

Calculate the scores between fir and the FIR's in population (One-To-Many Matching).

- CountedPtr< Matches > bestMatches (const FIR &fir, const Population &population, const Score &threshold, unsigned int maxMatches) const

  Calculates best matches of the comparison between fir and the FIR's in the population.

0.36.13.1 Detailed Description

Low level facial comparison facility.

Examples:

match.cc.

0.36.13.2 Constructor & Destructor Documentation

0.36.13.2.1 FRsdk::FacialMatchingEngine::FacialMatchingEngine (const Configuration &)

0.36.13.2.2 FRsdk::FacialMatchingEngine::FacialMatchingEngine (const FacialMatchingEngine &)

0.36.13.2.3 FRsdk::FacialMatchingEngine::~FacialMatchingEngine ( )

0.36.13.3 Member Function Documentation

0.36.13.3.1 CountedPtr<Matches> FRsdk::FacialMatchingEngine::bestMatches (const FIR & *fir*, const Population & *population*, const Score & *threshold*, unsigned int *maxMatches*) const

Calculates best matches of the comparison between fir and the FIR's in the population.

Matches returned are sorted by score value in descending order. Size of the match list returned is controlled by both a score threshold and a maximum size.

MT-safe (reentrant) It is safe to call this function

concurrently from different threads. The population has to be fixed during function execution.

This function takes care about the configured number of threads of the comparison algorism. (FRSDK.Comparison-Algorithm.NumberOfThreads)

Parameters

threshold	threshold for match decision
maxMatches	the maximum size of the FRsdk::Matches to be returned in the feedback

0.36.13.3.2 Score FRsdk::FacialMatchingEngine::compare (const FIR & *firA*, const FIR & *firB*) const

Calculate the score between firA and firB.

MT-safe (reentrant) It is safe to call this function concurrently from different threads.

0.36.13.3.3 CountedPtr<Scores> FRsdk::FacialMatchingEngine::compare (const FIR & *fir*, const Population & *population*) const

Calculate the scores between fir and the FIR's in population (One-To-Many Matching).

Scores in the list returned have the same order as the FIR's within the Population.

MT-safe (reentrant) It is safe to call this function

concurrently from different threads. The population has to be fixed during function execution.

This function does not use the settings of the number of threads to be used (FRSDK.ComparisonAlgorithm.Number-OfThreads). This behavior is by design. The returned scores are ordered in the same way as the FIRs are added to the population.

0.36.13.3.4 FacialMatchingEngine& FRsdk::FacialMatchingEngine::operator= (const FacialMatchingEngine &)

The documentation for this class was generated from the following file:

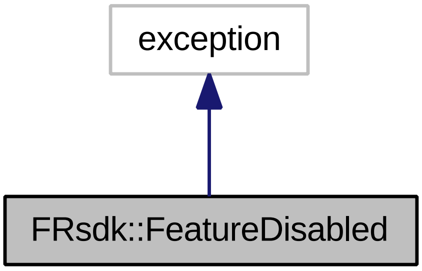
- frsdk/match.h

0.36.14 FRsdk::FeatureDisabled Class Reference

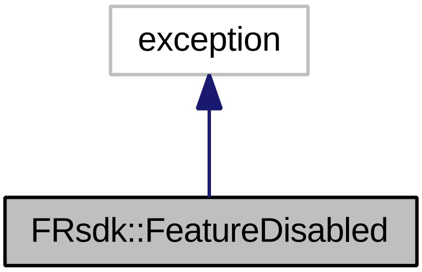
An object of this type is thrown at any time if requesting or accessing a disabled FaceVACS-SDK feature.

`#include <config.h>`

Inheritance diagram for FRsdk::FeatureDisabled:



Collaboration diagram for FRsdk::FeatureDisabled:



Public Member Functions

- FeatureDisabled (const char ∗msg_)
- ∼FeatureDisabled () throw ()
- virtual const char ∗ what () const throw ()

    returns a description of the feature which was disabled

0.36.14.1 Detailed Description

An object of this type is thrown at any time if requesting or accessing a disabled FaceVACS-SDK feature.

Examples:

acquisition.cc, cropfullfrontal.cc, enroll.cc, eyesfind.cc, identify.cc, match.cc, verify.cc, verifyan.cc, and vignetting.cc.

0.36.14.2 Constructor & Destructor Documentation

0.36.14.2.1 FRsdk::FeatureDisabled::FeatureDisabled (const char * *msg_*) [inline]

0.36.14.2.2 FRsdk::FeatureDisabled::∼FeatureDisabled ( ) throw) [inline]

0.36.14.3 Member Function Documentation

0.36.14.3.1 virtual const char∗ FRsdk::FeatureDisabled::what ( ) const throw) [inline], [virtual]

returns a description of the feature which was disabled

Examples:

acquisition.cc, cropfullfrontal.cc, enroll.cc, eyesfind.cc, identify.cc, match.cc, verify.cc, verifyan.cc, and vignetting.cc.

The documentation for this class was generated from the following file:

- frsdk/config.h

0.36.15 FRsdk::Enrollment::Feedback Class Reference

explicit template instantiation for win32

`#include <enroll.h>`

Public Member Functions

- Feedback (const CountedPtr< FeedbackBody > &i)
  body/handle supporting c'tor.
- void start () const
  Called at start of enrollment processing.
- void processingImage (const Image &img) const
  Called for each image when processed by the processor.
- void eyesFound (const Eyes::Location &l) const
  Called if eyes have been found in the current image; the location l indicates the position they have been found at.
- void eyesNotFound () const
  Called if no eyes have been found in the current image.
- void success (const FIR &fir) const
  Called if the enrollment was successful; passes the FIR created.
- void failure () const
  Called if the enrollment was not successful (due to failure conditions).
- void end () const
  Called at the end of the enrollment procedure.

0.36.15.1 Detailed Description

explicit template instantiation for win32

Feedback for the enrollment procedure.

The Feedback class is used for interaction with the environment during the processing of the enrollment. The member functions of the feedback will be called during the enrollment as the processing proceeds. This make it possible to watch the status of the enrollment for instance for gui feedback.

Examples:

enroll.cc.

0.36.15.2 Constructor & Destructor Documentation

0.36.15.2.1 FRsdk::Enrollment::Feedback::Feedback (const CountedPtr< FeedbackBody > & *i*) [inline]

body/handle supporting c'tor.

0.36.15.3 Member Function Documentation

0.36.15.3.1 void FRsdk::Enrollment::Feedback::end ( ) const [inline]

Called at the end of the enrollment procedure.

This function will be called in any case.

0.36.15.3.2 void FRsdk::Enrollment::Feedback::eyesFound (const Eyes::Location & *l*) const [inline]

Called if eyes have been found in the current image; the location l indicates the position they have been found at.

0.36.15.3.3 void FRsdk::Enrollment::Feedback::eyesNotFound ( ) const [inline]

Called if no eyes have been found in the current image.

This may happen if the image does not contain a face.

0.36.15.3.4 void FRsdk::Enrollment::Feedback::failure ( ) const [inline]

Called if the enrollment was not successful (due to failure conditions).

Possible failure conditions are:

- the enrollment time exceeds the maximum configured (default: 60 s) (Stream enrollment only)
- there are less "good" enrollment images than configured (default: 1)

0.36.15.3.5 void FRsdk::Enrollment::Feedback::processingImage (const Image & *img*) const [inline]

Called for each image when processed by the processor.

0.36.15.3.6 void FRsdk::Enrollment::Feedback::start ( ) const [inline]

Called at start of enrollment processing.

0.36.15.3.7 void FRsdk::Enrollment::Feedback::success (const FIR & *fir*) const [inline]

Called if the enrollment was successful; passes the FIR created.

The documentation for this class was generated from the following file:

- frsdk/enroll.h

0.36.16 FRsdk::Identification::Feedback Class Reference

explicit template instantiation for win32

`#include <ident.h>`

Public Member Functions

- Feedback (const CountedPtr< FeedbackBody > &i)

  body/handle supporting c'tor.
- void start () const

  Called at the start of identification processing.
- void processingImage (const Image &img) const

  Called for each image when processed by the processor.
- void eyesFound (const Eyes::Location &l) const

  Called if eyes have been found in the current image; the location l indicates the position they have been found at.
- void eyesNotFound () const

  Called if no eyes have been found in the current image; this may happen if the image does not contain a face.
- void matches (const FRsdk::Matches &m) const

  Called if at least one of the input images matches with the given FIR population.
- void end () const

  Called at the end of identification procedure.

0.36.16.1 Detailed Description

explicit template instantiation for win32

Feedback from the identification process

Abstract identification feedback is used to transfer information from the identification process to the client.

Examples:

identify.cc.

0.36.16.2 Constructor & Destructor Documentation

0.36.16.2.1 FRsdk::Identification::Feedback::Feedback (const CountedPtr< FeedbackBody > & *i*) [inline]

body/handle supporting c'tor.

0.36.16.3 Member Function Documentation

0.36.16.3.1 void FRsdk::Identification::Feedback::end ( ) const [inline]

Called at the end of identification procedure.

This function will be called in any case.

0.36.16.3.2 void FRsdk::Identification::Feedback::eyesFound (const Eyes::Location & *l*) const [inline]

Called if eyes have been found in the current image; the location l indicates the position they have been found at.

0.36.16.3.3 void FRsdk::Identification::Feedback::eyesNotFound ( ) const [inline]

Called if no eyes have been found in the current image; this may happen if the image does not contain a face.

0.36.16.3.4 void FRsdk::Identification::Feedback::matches (const FRsdk::Matches & *m*) const [inline]

Called if at least one of the input images matches with the given FIR population.

The best match is the first one and the worst the last one. The FIRs are referenced by names. The list of matches will be empty when life check is enabled but failed for processed images.

0.36.16.3.5 void FRsdk::Identification::Feedback::processingImage (const Image & *img*) const [inline]

Called for each image when processed by the processor.

0.36.16.3.6 void FRsdk::Identification::Feedback::start ( ) const [inline]

Called at the start of identification processing.

The documentation for this class was generated from the following file:

- frsdk/ident.h

0.36.17 FRsdk::Verification::Feedback Class Reference

explicit template instantiation for win32

`#include <verify.h>`

Public Member Functions

- Feedback (const CountedPtr< FeedbackBody > &i)

  body/handle supporting c'tor.
- void start () const

  Called at start of verification processing.
- void processingImage (const Image &img) const

  Called for each sample when processed by the processor.
- void eyesFound (const Eyes::Location &l) const

  Called if eyes have been found in the current sample; the location l indicates the position they have been found at.
- void eyesNotFound () const

  Called if no eyes could have been found in the current sample.
- void match (const Score &s) const

  Called if the current sample could be compared with the given FIR; s is the score obtained.
- void success () const

  Called if the verification was successful, i.e.
- void failure () const

  Called if verification failed.
- void end () const

  Called at the end of the verification procedure; this function will be called in either case.

0.36.17.1 Detailed Description

explicit template instantiation for win32

Feedback from the verification process

Abstract Verification feedback is used to transfer information from the verification process to the client.

Examples:

verify.cc, and verifyan.cc.

0.36.17.2 Constructor & Destructor Documentation

0.36.17.2.1 FRsdk::Verification::Feedback::Feedback (const CountedPtr< FeedbackBody > & *i*) [inline]

body/handle supporting c'tor.

0.36.17.3 Member Function Documentation

0.36.17.3.1 void FRsdk::Verification::Feedback::end ( ) const [inline]

Called at the end of the verification procedure; this function will be called in either case.

0.36.17.3.2 void FRsdk::Verification::Feedback::eyesFound (const Eyes::Location & *l*) const [inline]

Called if eyes have been found in the current sample; the location l indicates the position they have been found at.

0.36.17.3.3 void FRsdk::Verification::Feedback::eyesNotFound ( ) const [inline]

Called if no eyes could have been found in the current sample.

0.36.17.3.4 void FRsdk::Verification::Feedback::failure ( ) const [inline]

Called if verification failed.

A verification can fail for several reasons:

- all scores obtained from the probe samples were lower than the threshold
- None of the samples contained a (detectable) face
- sample quality was too low for all samples presented (stream verification only)
- timeout exceeded (stream verification only)
- an internal error prevented successful processing

0.36.17.3.5 void FRsdk::Verification::Feedback::match (const Score & *s*) const [inline]

Called if the current sample could be compared with the given FIR; s is the score obtained.

0.36.17.3.6 void FRsdk::Verification::Feedback::processingImage (const Image & *img*) const [inline]

Called for each sample when processed by the processor.

The intensity image part of the sample is provided to this function.

0.36.17.3.7 void FRsdk::Verification::Feedback::start ( ) const [inline]

Called at start of verification processing.

0.36.17.3.8 void FRsdk::Verification::Feedback::success ( ) const [inline]

Called if the verification was successful, i.e.

at least one of the input samples has got a match result above the given threshold.

The documentation for this class was generated from the following file:

- frsdk/verify.h

0.36.18 FRsdk::Enrollment::FeedbackBody Class Reference

Body class for Feedback.

`#include <enroll.h>`

Public Member Functions

- virtual ~FeedbackBody ()

d'tor

- virtual void start ()=0
- virtual void processingImage (const Image &img)=0
- virtual void eyesFound (const Eyes::Location &eyeLoc)=0
- virtual void eyesNotFound ()=0
- virtual void success (const FIR &)=0
- virtual void failure ()=0
- virtual void end ()=0

0.36.18.1 Detailed Description

Body class for Feedback.

Member documentation see Feedback.

Examples:

edialog.h.

0.36.18.2 Constructor & Destructor Documentation

0.36.18.2.1 virtual FRsdk::Enrollment::FeedbackBody::∼FeedbackBody ( ) [inline], [virtual]

d'tor

0.36.18.3 Member Function Documentation

0.36.18.3.1 virtual void FRsdk::Enrollment::FeedbackBody::end ( ) [pure virtual]

Examples:

edialog.h.

0.36.18.3.2 virtual void FRsdk::Enrollment::FeedbackBody::eyesFound (const Eyes::Location & *eyeLoc*) [pure virtual]

Examples:

edialog.h.

0.36.18.3.3 virtual void FRsdk::Enrollment::FeedbackBody::eyesNotFound ( ) [pure virtual]

Examples:

edialog.h.

0.36.18.3.4 virtual void FRsdk::Enrollment::FeedbackBody::failure ( ) [pure virtual]

Examples:

edialog.h.

0.36.18.3.5 virtual void FRsdk::Enrollment::FeedbackBody::processingImage (const Image & *img*) [pure virtual]

Examples:

edialog.h.

0.36.18.3.6 virtual void FRsdk::Enrollment::FeedbackBody::start ( ) [pure virtual]

Examples:

edialog.h.

0.36.18.3.7 virtual void FRsdk::Enrollment::FeedbackBody::success (const FIR &) [pure virtual]

Examples:

edialog.h.

The documentation for this class was generated from the following file:

- frsdk/enroll.h

0.36.19 FRsdk::Identification::FeedbackBody Class Reference

Body class for Feedback.

`#include <ident.h>`

Public Member Functions

- virtual ~FeedbackBody ()
- virtual void start ()=0
- virtual void processingImage (const Image &img)=0
- virtual void eyesFound (const Eyes::Location &eyeLoc)=0
- virtual void eyesNotFound ()=0
- virtual void matches (const FRsdk::Matches &matches)=0
- virtual void end ()=0

0.36.19.1 Detailed Description

Body class for Feedback.

Member documentation see Feedback.

Examples:

idialog.h.

0.36.19.2 Constructor & Destructor Documentation

0.36.19.2.1 virtual FRsdk::Identification::FeedbackBody::~FeedbackBody ( ) [inline], [virtual]

0.36.19.3 Member Function Documentation

0.36.19.3.1 virtual void FRsdk::Identification::FeedbackBody::end ( ) [pure virtual]

Examples:

idialog.h.

0.36.19.3.2 virtual void FRsdk::Identification::FeedbackBody::eyesFound (const **Eyes::Location** & *eyeLoc*) [pure virtual]

Examples:

idialog.h.

0.36.19.3.3 virtual void FRsdk::Identification::FeedbackBody::eyesNotFound ( ) [pure virtual]

Examples:

idialog.h.

0.36.19.3.4 virtual void FRsdk::Identification::FeedbackBody::matches (const **FRsdk::Matches** & *matches*) [pure virtual]

Examples:

idialog.h.

0.36.19.3.5 virtual void FRsdk::Identification::FeedbackBody::processingImage (const **Image** & *img*) [pure virtual]

Examples:

idialog.h.

0.36.19.3.6 virtual void FRsdk::Identification::FeedbackBody::start ( ) [pure virtual]

Examples:

idialog.h.

The documentation for this class was generated from the following file:

- frsdk/ident.h

0.36.20 FRsdk::Verification::FeedbackBody Class Reference

Body class for Feedback.

`#include <verify.h>`

Public Member Functions

- virtual ~FeedbackBody ()

  d'tor
- virtual void start ()=0
- virtual void processingImage (const Image &img)=0
- virtual void eyesFound (const Eyes::Location &eyeLoc)=0
- virtual void eyesNotFound ()=0
- virtual void match (const Score &)=0
- virtual void success ()=0
- virtual void failure ()=0
- virtual void end ()=0

0.36.20.1 Detailed Description

Body class for Feedback.

Member documentation see Feedback.

Examples:

vdialog.h.

0.36.20.2 Constructor & Destructor Documentation

0.36.20.2.1 virtual FRsdk::Verification::FeedbackBody::∼FeedbackBody ( ) [inline], [virtual]

d'tor

0.36.20.3 Member Function Documentation

0.36.20.3.1 virtual void FRsdk::Verification::FeedbackBody::end ( ) [pure virtual]

Examples:

vdialog.h.

0.36.20.3.2 virtual void FRsdk::Verification::FeedbackBody::eyesFound (const Eyes::Location & *eyeLoc*) [pure virtual]

Examples:

vdialog.h.

0.36.20.3.3 virtual void FRsdk::Verification::FeedbackBody::eyesNotFound ( ) [pure virtual]

Examples:

vdialog.h.

0.36.20.3.4 virtual void FRsdk::Verification::FeedbackBody::failure ( ) [pure virtual]

Examples:

vdialog.h.

0.36.20.3.5 virtual void FRsdk::Verification::FeedbackBody::match (const Score &) [pure virtual]

Examples:

vdialog.h.

0.36.20.3.6 virtual void FRsdk::Verification::FeedbackBody::processingImage (const Image & *img*) [pure virtual]

Examples:

vdialog.h.

0.36.20.3.7 virtual void FRsdk::Verification::FeedbackBody::start ( ) [pure virtual]

Examples:

vdialog.h.

0.36.20.3.8 virtual void FRsdk::Verification::FeedbackBody::success ( ) [pure virtual]

Examples:

vdialog.h.

The documentation for this class was generated from the following file:

- frsdk/verify.h

0.36.21 FRsdk::Face::Finder Class Reference

Face::Finder (handle) This class represents a interface to the face finding procedure.

`#include <face.h>`

Public Member Functions

- Finder (const Configuration &)

  Create an instance of class Finder.
- Finder (const Finder &)
- Finder & operator= (const Finder &)
- ∼Finder ()
- LocationSet find (const Image &, float minRelativeEyeDistance=0.1, float maxRelativeEyeDistance=0.4, int x1=INT_MIN/2, int y1=INT_MIN/2, int x2=INT_MAX/2-1, int y2=INT_MAX/2-1) const

  Returns a list of Face::Locations for the faces found.

0.36.21.1 Detailed Description

Face::Finder (handle) This class represents a interface to the face finding procedure.

Examples:

acquisition.cc, cropfullfrontal.cc, eyesfind.cc, facefind.cc, and verifyan.cc.

0.36.21.2 Constructor & Destructor Documentation

0.36.21.2.1 FRsdk::Face::Finder::Finder (const Configuration &)

Create an instance of class Finder.

0.36.21.2.2 FRsdk::Face::Finder::Finder (const Finder &)

0.36.21.2.3 FRsdk::Face::Finder::∼Finder ( )

0.36.21.3 Member Function Documentation

0.36.21.3.1 LocationSet FRsdk::Face::Finder::find (const Image & , float *minRelativeEyeDistance* = `0.1`, float *maxRelativeEyeDistance* = `0.4`, int *x1* = `INT_MIN/2`, int *y1* = `INT_MIN/2`, int *x2* = `INT_MAX/2-1`, int *y2* = `INT_MAX/2-1`) const

Returns a list of Face::Locations for the faces found.

Searching is focused to faces in the given eye distance range (relative to the image width) and within the given search box spanned by (x1, y1, x2, y2). The given search box is clipped by the boundaries of the image, so the default settings for the search box denote that the entire image has to be used as search area. (See your compiler's

<limits.h> for INT_MIN and INT_MAX definitions.). Also note that minRelativeEyeDistance and maxRelativeEyeDistance are hints for the finding engine. The search process can result in faces that are slightly smaller or bigger than suggested by these numbers.

Accuracy of face finding results is limited, i.e. both the location and the face width returned with the Locations can slightly deviate from actual values. The Face::Finder is optimized to collaborate with the Eyes::Finder, i.e. passing a Face::Location returned by the Face::Finder will enable the Eyes::Finder to find accurate eye positions.

MT-safe (reentrant) It is safe to call the find() method concurrently from different threads.

0.36.21.3.2 Finder& FRsdk::Face::Finder::operator= (const Finder &)

The documentation for this class was generated from the following file:

- frsdk/face.h

0.36.22 FRsdk::Eyes::Finder Class Reference

Eyes::Finder (handle) This class represents a interface to the eye finding procedure.

`#include <eyes.h>`

Public Member Functions

- Finder (const Configuration &)

  create an instance of class Finder
- Finder (const Finder &)
- Finder & operator= (const Finder &)
- ∼Finder ()
- LocationSet find (const Image &img, const Face::Location &l) const

  returns a list of eye locations found within a face at the given Face::Location.

0.36.22.1 Detailed Description

Eyes::Finder (handle) This class represents a interface to the eye finding procedure.

MT-safe (reentrant) It is safe to call the find method concurrently

from different threads.

Examples:

acquisition.cc, cropfullfrontal.cc, eyesfind.cc, and verifyan.cc.

0.36.22.2 Constructor & Destructor Documentation

0.36.22.2.1 FRsdk::Eyes::Finder::Finder (const Configuration &)

create an instance of class Finder

0.36.22.2.2 FRsdk::Eyes::Finder::Finder (const Finder &)

0.36.22.2.3 FRsdk::Eyes::Finder::∼Finder ( )

0.36.22.3 Member Function Documentation

0.36.22.3.1 LocationSet FRsdk::Eyes::Finder::find (const Image & *img*, const Face::Location & *l*) const

returns a list of eye locations found within a face at the given Face::Location.

Parameters

img	the image to find eyes in
l	the location of the face

0.36.22.3.2 Finder& FRsdk::Eyes::Finder::operator= (const Finder &)

The documentation for this class was generated from the following file:

- frsdk/eyes.h

0.36.23 FRsdk::FIR Class Reference

FIR - Facial Identification Record.

```
#include <fir.h>
```

Public Member Functions

- FIR (const FIR &)

  copy c'tor
- FIR & operator= (const FIR &)

  assignment operator
- bool operator== (const FIR &) const

  comparison operator
- ∼FIR ()

  d'tor
- unsigned int size () const

  returns the size of the FIR memory representation
- std::string version () const

  returns the version id string
- void serialize (Byte ∗buf) const

  write a serialized platform independent representation to a buffer which must be able to hold at least FIR::size() bytes.
- void serialize (std::ostream &o) const

  Write a serialized platform independent representation to an ostream; the format of this representation is equal to that of the memory representation obtained with serialize(Byte∗).
- void writeTo (Byte ∗&buf) const

  write a platform dependent representation to a buffer which must be able to hold at least FIR::size() bytes.

Friends

- class Implementation

0.36.23.1 Detailed Description

FIR - Facial Identification Record.

The FIR is a serializable byte stream which encapsulates the transformed representation of a face which is often called feature set. Use Enrollment::Processor to create FIRs from primary facial data (images) or FIRBuilder to (re)create FIRs from serialized representations.

Examples:

edialog.h, match.cc, verify.cc, and verifyan.cc.

0.36.23.2 Constructor & Destructor Documentation

0.36.23.2.1 FRsdk::FIR::FIR (const FIR &)

copy c'tor

0.36.23.2.2 FRsdk::FIR::∼FIR ( )

d'tor

0.36.23.3 Member Function Documentation

0.36.23.3.1 FIR& FRsdk::FIR::operator= (const FIR &)

assignment operator

0.36.23.3.2 bool FRsdk::FIR::operator== (const FIR &) const

comparison operator

0.36.23.3.3 void FRsdk::FIR::serialize (Byte ∗ *buf*) const

write a serialized platform independent representation to a buffer which must be able to hold at least FIR::size() bytes.

For reconstruction of a FIR stored in this way FIRBuilder::build(const Byte∗, unsigned int) is to be used.

0.36.23.3.4 void FRsdk::FIR::serialize (std::ostream & *o*) const

Write a serialized platform independent representation to an ostream; the format of this representation is equal to that of the memory representation obtained with serialize(Byte∗).

0.36.23.3.5 unsigned int FRsdk::FIR::size ( ) const

returns the size of the FIR memory representation

0.36.23.3.6 std::string FRsdk::FIR::version ( ) const

returns the version id string

0.36.23.3.7 void FRsdk::FIR::writeTo (Byte ∗& *buf*) const

write a platform dependent representation to a buffer which must be able to hold at least FIR::size() bytes.

After execution p points to the first byte after the FIR's data. For reconstruction of a FIR stored in this way FIR-Builder::build(Byte∗&) is to be used.

0.36.23.4 Friends And Related Function Documentation

0.36.23.4.1 friend class Implementation [friend]

The documentation for this class was generated from the following file:

- frsdk/fir.h

0.36.24 FRsdk::FIRBuilder Class Reference

Building FIRs from serialized representations Use Enrollment::Processor to build FIRs from primary biometric data (face images).

```
#include <fir.h>
```

Public Member Functions

- FIRBuilder (const Configuration &)
- FIRBuilder (const FIRBuilder &)
- FIRBuilder & operator= (const FIRBuilder &)
- ∼FIRBuilder ()
- FIR buildVoidFIR () const

  create a FIR that yields a score of 0.0 when compared with any other FIR built from the same FIR builder
- FIR build (const Byte ∗, unsigned int len) const

  create a FIR from platform independent representation created with FIR::serialize()
- FIR build (std::istream &) const

  create a FIR from stream containing a platform independent representation created with FIR::serialize()
- FIR build (Byte ∗&p) const

  create a FIR from platform dependent representation created using FIR::writeTo, starting from p, the memory must be valid during the lifetime of the FIR object, after the construction p points to the first byte after the FIR's data.

0.36.24.1 Detailed Description

Building FIRs from serialized representations Use Enrollment::Processor to build FIRs from primary biometric data (face images).

Examples:

identify.cc, match.cc, verify.cc, and verifyan.cc.

0.36.24.2 Constructor & Destructor Documentation

0.36.24.2.1 FRsdk::FIRBuilder::FIRBuilder (const **Configuration** &)

0.36.24.2.2 FRsdk::FIRBuilder::FIRBuilder (const **FIRBuilder** &)

0.36.24.2.3 FRsdk::FIRBuilder::∼FIRBuilder ( )

0.36.24.3 Member Function Documentation

0.36.24.3.1 FIR FRsdk::FIRBuilder::build (const **Byte** ∗ , unsigned int *len*) const

create a FIR from platform independent representation created with FIR::serialize()

0.36.24.3.2 FIR FRsdk::FIRBuilder::build (std::istream &) const

create a FIR from stream containing a platform independent representation created with FIR::serialize()

0.36.24.3.3 FIR FRsdk::FIRBuilder::build (**Byte** ∗& *p*) const

create a FIR from platform dependent representation created using FIR::writeTo, starting from p, the memory must be valid during the lifetime of the FIR object, after the construction p points to the first byte after the FIR's data.

0.36.24.3.4 FIR FRsdk::FIRBuilder::buildVoidFIR ( ) const

create a FIR that yields a score of 0.0 when compared with any other FIR built from the same FIR builder

0.36.24.3.5 FIRBuilder& FRsdk::FIRBuilder::operator= (const FIRBuilder &)

The documentation for this class was generated from the following file:

- frsdk/fir.h

0.36.25 FRsdk::Image Class Reference

explicit template instantiation for win32

`#include <image.h>`

Public Member Functions

- Image (const CountedPtr< ImageBody > &i)

  build an image handle from an abstract implementation
- bool isColor () const

  returns true if image is based on color data.
- unsigned int width () const

  returns the width of the image in pixels
- unsigned int height () const

  returns the height of the image n pixels
- const Byte ∗ grayScaleRepresentation () const

  Returns a pointer to an array of size width() ∗ height() ∗ sizeof(FRsdk::Byte) containing the gray scale representation of the image.
- const Rgb ∗ colorRepresentation () const

  Returns a pointer to an array of size width() ∗ height() ∗ sizeof(FRsdk::Rgb) containing the color representation of the image.
- std::string name () const

  returns the name of the image, or an empty string
- ImageBody & body () const

  Access to the body.

0.36.25.1 Detailed Description

explicit template instantiation for win32

Image handle (interface)

Examples:

acquisition.cc, capdev.cc, cropfullfrontal.cc, edialog.h, enroll.cc, eyesfind.cc, facefind.cc, idialog.h, tracklife.cc, trackrec.cc, vdialog.h, verify.cc, verifyan.cc, and vignetting.cc.

0.36.25.2 Constructor & Destructor Documentation

0.36.25.2.1 FRsdk::Image::Image (const CountedPtr< ImageBody > & *i*) [inline]

build an image handle from an abstract implementation

0.36.25.3 Member Function Documentation

0.36.25.3.1 ImageBody& FRsdk::Image::body ( ) const [inline]

Access to the body.

This enables clients to perform (dynamic) casting to the concrete body class.

0.36.25.3.2 const Rgb* FRsdk::Image::colorRepresentation ( ) const [inline]

Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Rgb) containing the color representation of the image.

The pointer has to remain valid during the whole lifetime of the ImageBody object. This function returns only valid data if isColor() returns true. Note that the order of colors per pixel is BGR (blue-green red) instead of RGB.

0.36.25.3.3 const Byte* FRsdk::Image::grayScaleRepresentation ( ) const [inline]

Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Byte) containing the gray scale representation of the image.

The pointer has to remain valid during the whole lifetime of the ImageBody object. This function returns a valid pointer only if isColor() returns false. See documentation of ImageBody::grayScaleRepresentation()

0.36.25.3.4 unsigned int FRsdk::Image::height ( ) const [inline]

returns the height of the image n pixels

Examples:

capdev.cc.

0.36.25.3.5 bool FRsdk::Image::isColor ( ) const [inline]

returns true if image is based on color data.

false will be returend if the images contains only intensity information

0.36.25.3.6 std::string FRsdk::Image::name ( ) const [inline]

returns the name of the image, or an empty string

Examples:

edialog.h, idialog.h, and vdialog.h.

0.36.25.3.7 unsigned int FRsdk::Image::width ( ) const [inline]

returns the width of the image in pixels

Examples:

capdev.cc.

The documentation for this class was generated from the following file:

- frsdk/image.h

0.36.26 FRsdk::ImageBody Class Reference

Abstract image body.

`#include <image.h>`

Public Member Functions

- virtual ~ImageBody ()
- virtual bool isColor () const =0

  returns true if image is based on color data and colorRepresentation() returns valid pointer.
- virtual unsigned int width () const =0

  returns the width of the image in pixels
- virtual unsigned int height () const =0

  returns the height of the image in pixels
- virtual const Byte * grayScaleRepresentation () const =0

  *Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Byte) containing the gray scale representation of the image.*
- virtual const Rgb * colorRepresentation () const =0

  *Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Rgb) containing the color representation of the image.*
- virtual std::string name () const =0

  returns the name of the image, or an empty string

0.36.26.1 Detailed Description

Abstract image body.

0.36.26.2 Constructor & Destructor Documentation

0.36.26.2.1 virtual FRsdk::ImageBody::~ImageBody ( ) [inline], [virtual]

0.36.26.3 Member Function Documentation

0.36.26.3.1 virtual const Rgb* FRsdk::ImageBody::colorRepresentation ( ) const [pure virtual]

Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Rgb) containing the color representation of the image.

The pointer has to remain valid during the whole lifetime of the ImageBody object. This function has to return only a valid pointer if isColor() returns true. Note that the order of colors per pixel is BGR (blue-green red) instead of RGB.

0.36.26.3.2 virtual const Byte* FRsdk::ImageBody::grayScaleRepresentation ( ) const [pure virtual]

Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Byte) containing the gray scale representation of the image.

The pointer has to remain valid during the whole lifetime of the ImageBody object. The function has to return a valid array only if isColor() returns false

0.36.26.3.3 virtual unsigned int FRsdk::ImageBody::height ( ) const [pure virtual]

returns the height of the image in pixels

0.36.26.3.4 virtual bool FRsdk::ImageBody::isColor ( ) const [pure virtual]

returns true if image is based on color data and colorRepresentation() returns valid pointer.

false will be returend if the images contains only intensity information and grayScaleRepresentation() returns a valid pointer.

0.36.26.3.5 virtual std::string FRsdk::ImageBody::name ( ) const [pure virtual]

returns the name of the image, or an empty string

0.36.26.3.6 virtual unsigned int FRsdk::ImageBody::width ( ) const [pure virtual]

returns the width of the image in pixels

The documentation for this class was generated from the following file:

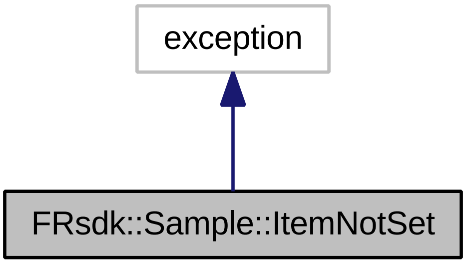
- frsdk/image.h

0.36.27 FRsdk::Sample::ItemNotSet Class Reference

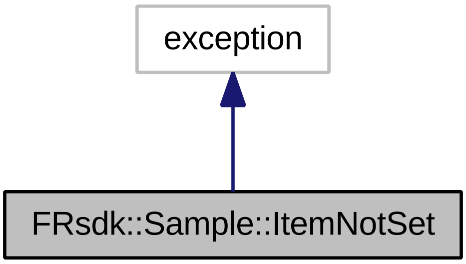
Specific exception type indicating access to optional data not present in Sample.

`#include <sample.h>`

Inheritance diagram for FRsdk::Sample::ItemNotSet:



Collaboration diagram for FRsdk::Sample::ItemNotSet:



Public Member Functions

- ItemNotSet (const std::string &id)
- ∼ItemNotSet () throw ()
- virtual const char ∗ what () const throw ()

0.36.27.1 Detailed Description

Specific exception type indicating access to optional data not present in Sample.

0.36.27.2 Constructor & Destructor Documentation

0.36.27.2.1 FRsdk::Sample::ItemNotSet::ItemNotSet (const std::string & *id*) [inline]

0.36.27.2.2 FRsdk::Sample::ItemNotSet::∼ItemNotSet ( ) throw) [inline]

0.36.27.3 Member Function Documentation

0.36.27.3.1 virtual const char* FRsdk::Sample::ItemNotSet::what ( ) const throw) [inline], [virtual]

The documentation for this class was generated from the following file:

- frsdk/sample.h

0.36.28 FRsdk::LenseDistortionCorrector Class Reference

A class providing radial lense distortion correction.

`#include <ldc.h>`

Public Member Functions

- LenseDistortionCorrector (const FRsdk::Configuration &, const float &k, unsigned int img_w, unsigned int img_h, int dcx=0, int dcy=0)

  Instantiate a LenseDistortionCorrector.
- FRsdk::Image undistort (const FRsdk::Image &img)
- LenseDistortionCorrector (const LenseDistortionCorrector &)
- LenseDistortionCorrector & operator= (const LenseDistortionCorrector &)
- ~LenseDistortionCorrector ()

0.36.28.1 Detailed Description

A class providing radial lense distortion correction.

MT-safe (reentrant) It is safe to call member functions of this class concurrently from different threads.

0.36.28.2 Constructor & Destructor Documentation

0.36.28.2.1 FRsdk::LenseDistortionCorrector::LenseDistortionCorrector (const FRsdk::Configuration & , const float & *k*, unsigned int *img_w*, unsigned int *img_h*, int *dcx* = 0, int *dcy* = 0)

Instantiate a LenseDistortionCorrector.

The distortion parameter k can be positive or negative, where positive values correct 'barrel' distortions, while negative values correct 'pillow' distortions. The minimum and maximum admissible values for k depend on the instantiation image size. Inappropriate values will cause an exception upon instantiation. With larger images the maximum appropriate (absolute) value of k decreases. Values outside of [-1,1] are always inappropriate.

The appropriate value for k is best determined visually by examining an image (e.g. displaying a rectangular grid) distorted by the lense and gradually changing k until the distortion is minimized. For this purpose FaceVACS SDK provides the 'ldcview' utility in the 'bin' subdir. Note that due to performance reasons the distortion model is an approximate one and will work properly with small values of k only.

Parameters

k	Parameter controlling undistortion.
img_w	width of images to be processed
img_h	height of images to be processed
dcx	Location of distortion center

dcy	(optical axis) relative to image center

0.36.28.2.2 FRsdk::LenseDistortionCorrector::LenseDistortionCorrector (const LenseDistortionCorrector &)

0.36.28.2.3 FRsdk::LenseDistortionCorrector::∼LenseDistortionCorrector ( )

0.36.28.3 Member Function Documentation

0.36.28.3.1 LenseDistortionCorrector& FRsdk::LenseDistortionCorrector::operator= (const LenseDistortionCorrector &)

0.36.28.3.2 FRsdk::Image FRsdk::LenseDistortionCorrector::undistort (const FRsdk::Image & *img*)

Parameters

img	The Image to be undistorted. Dimensions (w,h) of the image must match those used to instantiate this LenseDistortionCorrector, otherwise an exception will be thrown

The documentation for this class was generated from the following file:

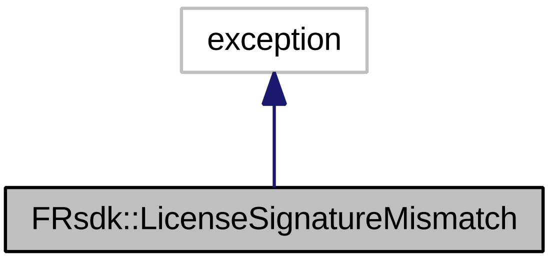
- frsdk/ldc.h

0.36.29 FRsdk::LicenseSignatureMismatch Class Reference

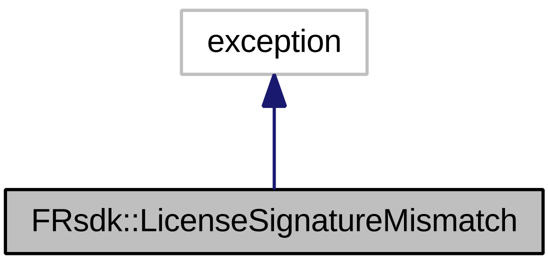
License signature mismatch.

```
#include <config.h>
```

Inheritance diagram for FRsdk::LicenseSignatureMismatch:



Collaboration diagram for FRsdk::LicenseSignatureMismatch:



Public Member Functions

- LicenseSignatureMismatch (const char ∗msg_)
- ∼LicenseSignatureMismatch () throw ()

- virtual const char ∗ what () const throw ()

    returns a description of the error

0.36.29.1 Detailed Description

License signature mismatch.

An object of this type is thrown at any time if the license activation information is missing, incomplete or the license is expired.

Examples:

acquisition.cc, cropfullfrontal.cc, enroll.cc, eyesfind.cc, identify.cc, match.cc, tracklife.cc, trackrec.cc, verify.cc, verifyan.cc, and vignetting.cc.

0.36.29.2 Constructor & Destructor Documentation

0.36.29.2.1 FRsdk::LicenseSignatureMismatch::LicenseSignatureMismatch (const char ∗ *msg_*) `[inline]`

0.36.29.2.2 FRsdk::LicenseSignatureMismatch::∼LicenseSignatureMismatch ( ) throw) `[inline]`

0.36.29.3 Member Function Documentation

0.36.29.3.1 virtual const char∗ FRsdk::LicenseSignatureMismatch::what ( ) const throw) `[inline]`, `[virtual]`

returns a description of the error

Examples:

acquisition.cc, cropfullfrontal.cc, enroll.cc, eyesfind.cc, identify.cc, match.cc, tracklife.cc, trackrec.cc, verify.cc, verifyan.cc, and vignetting.cc.

The documentation for this class was generated from the following file:

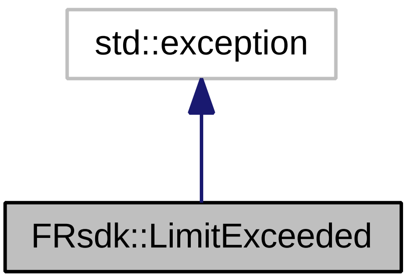
- frsdk/config.h

0.36.30 FRsdk::LimitExceeded Class Reference

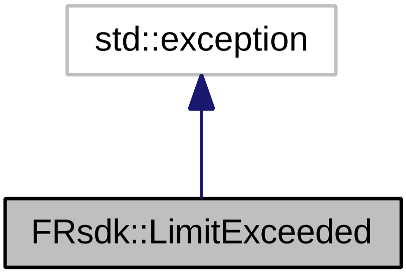
An object of this type is thrown at any time if a configured limit of FaceVACS-SDK is exceeded.

`#include <config.h>`

Inheritance diagram for FRsdk::LimitExceeded:



Collaboration diagram for FRsdk::LimitExceeded:



Public Member Functions

- LimitExceeded (const char ∗msg_)
- ∼LimitExceeded () throw ()

  returns a description of the limit which was exceeded
- virtual const char ∗ what () const throw ()

0.36.30.1 Detailed Description

An object of this type is thrown at any time if a configured limit of FaceVACS-SDK is exceeded.

0.36.30.2 Constructor & Destructor Documentation

0.36.30.2.1 FRsdk::LimitExceeded::LimitExceeded (const char ∗ *msg_*) [inline]

0.36.30.2.2 FRsdk::LimitExceeded::∼LimitExceeded ( ) throw) [inline]

returns a description of the limit which was exceeded

0.36.30.3 Member Function Documentation

0.36.30.3.1 virtual const char∗ FRsdk::LimitExceeded::what ( ) const throw) [inline], [virtual]

The documentation for this class was generated from the following file:

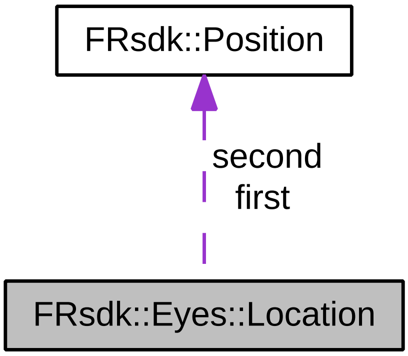
- frsdk/config.h

0.36.31 FRsdk::Eyes::Location Struct Reference

The Eyes::Location describes a location in the image where eyes within a face have been found.

`#include <eyes.h>`

Collaboration diagram for FRsdk::Eyes::Location:



Public Member Functions

- Location (const Position &f, const Position &s, float fc=0.0f, float sc=0.0f)

Public Attributes

- Position first

  position of first eye
- Position second

  position of second eye
- float firstConfidence

  confidence for first eye
- float secondConfidence

  confidence for second eye

0.36.31.1 Detailed Description

The Eyes::Location describes a location in the image where eyes within a face have been found.

The positions represent the center (located at half the distance between left and right eye corner) of the first and the second eye, respectively. First and second eye positions are defined relative to the image's coordinate system (and corresponds to the "usual" way images are displayed). The first eye is by definition the one with the lowest x-coodinate and the second the other. Whether the first eye (following this definition) is at the same time the person's actual left eye will depend on how the image is acquired and/or transformed. For common devices (cameras, scanners, image processing libraries, displays etc), the first eye definition in our context will correspond to the person's right eye. On the contrary, if one of these devices mirrors the image, then the first eye in this context will correspond to the person's left eye.

The usual range of the confidence is [0...6] - high values are good confidences, values near to 0 are bad confidences. Values in the range 2 ... 4 are usual.

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, idialog.h, and vdialog.h.

0.36.31.2 Constructor & Destructor Documentation

0.36.31.2.1 FRsdk::Eyes::Location::Location (const **Position** & *f*, const **Position** & *s*, float *fc* = `0.0f`, float *sc* = `0.0f`) `[inline]`

0.36.31.3 Member Data Documentation

0.36.31.3.1 Position FRsdk::Eyes::Location::first

position of first eye

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, idialog.h, and vdialog.h.

0.36.31.3.2 float FRsdk::Eyes::Location::firstConfidence

confidence for first eye

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, idialog.h, and vdialog.h.

0.36.31.3.3 Position FRsdk::Eyes::Location::second

position of second eye

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, idialog.h, and vdialog.h.

0.36.31.3.4 float FRsdk::Eyes::Location::secondConfidence

confidence for second eye

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, idialog.h, and vdialog.h.

The documentation for this struct was generated from the following file:

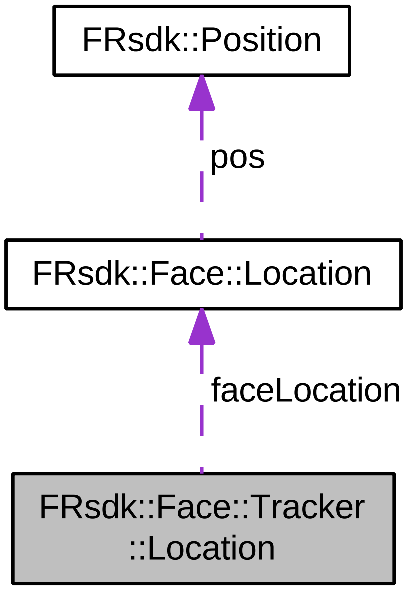
- frsdk/eyes.h

0.36.32 FRsdk::Face::Tracker::Location Struct Reference

The location of a face being tracked by the face tracker.

```
#include <tracker.h>
```

Collaboration diagram for FRsdk::Face::Tracker::Location:



Public Member Functions

- Location (const std::string &id_, const Face::Location &fl_)

Public Attributes

- const std::string id
- const Face::Location faceLocation

0.36.32.1 Detailed Description

The location of a face being tracked by the face tracker.

Location.id denotes the identifier assigned to a tracked face. A person's face tracked across multiple frames gets the same identifier. Location.eyesLocation denotes the positions of a tracked face's eyes in a frame.

Examples:

tracklife.cc, and trackrec.cc.

0.36.32.2 Constructor & Destructor Documentation

0.36.32.2.1 FRsdk::Face::Tracker::Location::Location (const std::string & *id_*, const Face::Location & *fl_*) [inline]

0.36.32.3 Member Data Documentation

0.36.32.3.1 const Face::Location FRsdk::Face::Tracker::Location::faceLocation

Examples:

tracklife.cc, and trackrec.cc.

0.36.32.3.2 const std::string FRsdk::Face::Tracker::Location::id

Examples:

tracklife.cc, and trackrec.cc.

The documentation for this struct was generated from the following file:

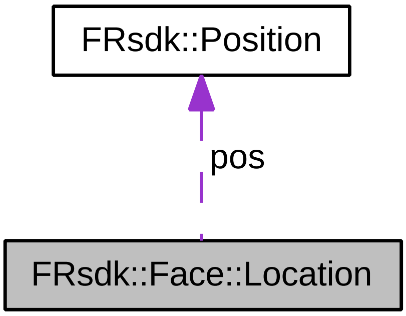
- frsdk/tracker.h

0.36.33 FRsdk::Face::Location Struct Reference

The Face::Location describes a image location where a face was found.

`#include <face.h>`

Collaboration diagram for FRsdk::Face::Location:



Public Member Functions

- Location (const Position &p, float w, float c=0.0f, float rotationAngle_=0.0f)

Public Attributes

- Position pos
  the face position
- float width
  the estimated eye distance of the face found
- float confidence
  confidence for the face
- float rotationAngle
  in-plane rotation angle (roll) of the face

0.36.33.1 Detailed Description

The Face::Location describes a image location where a face was found.

The Position pos is the middle point of a line crossing the eyes of that face. The width is the length of the line which connects the centers of the two eyes. The usual range of the confidence is [0...5] - high values are good confidences, values near to 0 are bad confidences. Values in the range 2 ... 4 are common. The in-plane rotation angle (roll) of the face is in radians. To get an horizontally aligned face (eyes) patch, rotate the patch by the rotationAngle in mathematically positive sense (with respect to the patch center).

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.33.2 Constructor & Destructor Documentation

0.36.33.2.1 FRsdk::Face::Location::Location (const Position & *p*, float *w*, float *c* = `0.0f`, float *rotationAngle_* = `0.0f`) `[inline]`

0.36.33.3 Member Data Documentation

0.36.33.3.1 float FRsdk::Face::Location::confidence

confidence for the face

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.33.3.2 Position FRsdk::Face::Location::pos

the face position

Examples:

acquisition.cc, cropfullfrontal.cc, tracklife.cc, and trackrec.cc.

0.36.33.3.3 float FRsdk::Face::Location::rotationAngle

in-plane rotation angle (roll) of the face

0.36.33.3.4 float FRsdk::Face::Location::width

the estimated eye distance of the face found

Examples:

acquisition.cc, and cropfullfrontal.cc.

The documentation for this struct was generated from the following file:

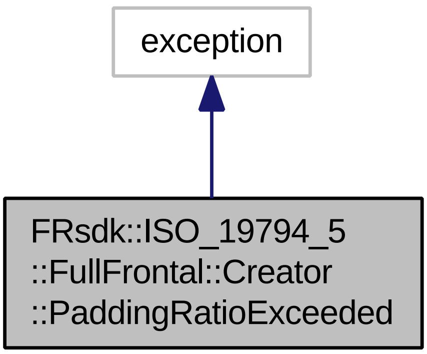
- frsdk/face.h

0.36.34 FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded Class Reference

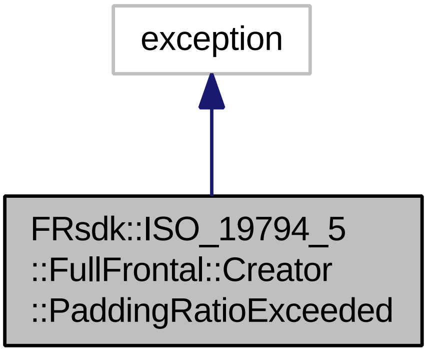
Exception of this type is thrown when the padding ratio of the to be extracted FullFrontal portrait exceeds the pre-configured threshold.

`#include <fullfrontal.h>`

Inheritance diagram for FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded:



Collaboration diagram for FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded:



Public Member Functions

- PaddingRatioExceeded (float paddingRatio) throw ()
- virtual ∼PaddingRatioExceeded () throw ()
- virtual const char ∗ what () const throw ()
- float paddingRatio () const

0.36.34.1 Detailed Description

Exception of this type is thrown when the padding ratio of the to be extracted FullFrontal portrait exceeds the pre-configured threshold.

0.36.34.2 Constructor & Destructor Documentation

0.36.34.2.1 FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded::PaddingRatioExceeded (float *paddingRatio*) throw) [inline], [explicit]

0.36.34.2.2 virtual FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded::∼PaddingRatioExceeded ( ) throw) [inline], [virtual]

0.36.34.3 Member Function Documentation

0.36.34.3.1 float FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded::paddingRatio ( ) const [inline]

0.36.34.3.2 virtual const char∗ FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded::what ( ) const throw) [inline], [virtual]

The documentation for this class was generated from the following file:

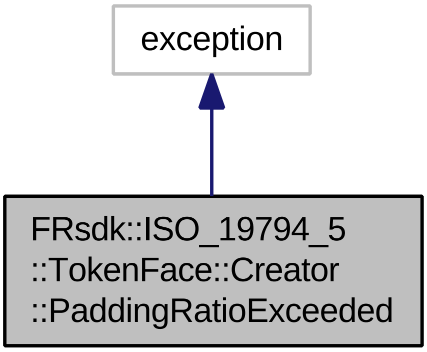
- frsdk/fullfrontal.h

0.36.35 FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded Class Reference

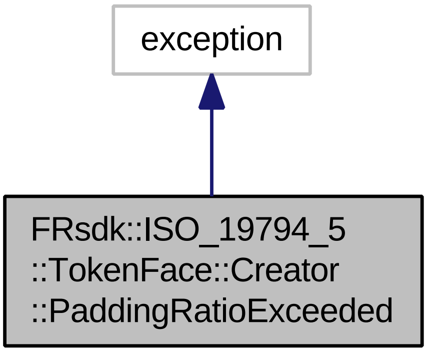
Exception of this type is thrown when the padding ratio of the to be extracted TokenFace exceeds the pre-configured threshold.

`#include <tokenface.h>`

Inheritance diagram for FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded:



Collaboration diagram for FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded:



Public Member Functions

- PaddingRatioExceeded (float paddingRatio) throw ()
- virtual ~PaddingRatioExceeded () throw ()
- virtual const char * what () const throw ()
- float paddingRatio () const

0.36.35.1 Detailed Description

Exception of this type is thrown when the padding ratio of the to be extracted TokenFace exceeds the pre-configured threshold.

0.36.35.2 Constructor & Destructor Documentation

0.36.35.2.1 FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded::PaddingRatioExceeded (float *paddingRatio*) throw) [inline], [explicit]

0.36.35.2.2 virtual FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded::~PaddingRatioExceeded ( ) throw) [inline], [virtual]

0.36.35.3 Member Function Documentation

0.36.35.3.1 float FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded::paddingRatio ( ) const [inline]

0.36.35.3.2 virtual const char* FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded::what ( ) const throw) [inline], [virtual]

The documentation for this class was generated from the following file:

- frsdk/tokenface.h

0.36.36 FRsdk::PointSet Class Reference

`#include <shapeimage.h>`

Public Member Functions

- PointSet (const CountedPtr< PointSetBody > &s)
- PointSetBody & body () const

0.36.36.1 Constructor & Destructor Documentation

0.36.36.1.1 FRsdk::PointSet::PointSet (const CountedPtr< PointSetBody > & *s*) [inline]

0.36.36.2 Member Function Documentation

0.36.36.2.1 PointSetBody& FRsdk::PointSet::body ( ) const [inline]

The documentation for this class was generated from the following file:

- frsdk/shapeimage.h

0.36.37 FRsdk::PointSetBody Class Reference

`#include <shapeimage.h>`

Public Member Functions

- virtual ~PointSetBody ()

0.36.37.1 Constructor & Destructor Documentation

0.36.37.1.1 virtual FRsdk::PointSetBody::~PointSetBody ( ) [inline], [virtual]

The documentation for this class was generated from the following file:

- frsdk/shapeimage.h

0.36.38 FRsdk::Population Class Reference

An ordered (in the order of additions by add()) set of named FIR's which represents the population used for identifications.

`#include <fir.h>`

Public Member Functions

- Population (const Configuration &)

    create a population object using the configuration
- Population (const Population &)

    copy c'tor
- Population & operator= (const Population &)

    assignment operator
- ∼Population ()

    d'tor
- void append (const FIR &fir, const std::string &name)

    append a named FIR to the population
- void remove (const std::string &name)

    remove a FIR by name from the population
- FIR get (const std::string &name) const

    get the mapped FIR by name

Friends

- class Implementation

0.36.38.1 Detailed Description

An ordered (in the order of additions by add()) set of named FIR's which represents the population used for identifications.

Examples:

identify.cc, and match.cc.

0.36.38.2 Constructor & Destructor Documentation

0.36.38.2.1 FRsdk::Population::Population (const Configuration &)

create a population object using the configuration

0.36.38.2.2 FRsdk::Population::Population (const Population &)

copy c'tor

0.36.38.2.3 FRsdk::Population::∼Population ( )

d'tor

0.36.38.3 Member Function Documentation

0.36.38.3.1 void FRsdk::Population::append (const FIR & *fir,* const std::string & *name*)

append a named FIR to the population

0.36.38.3.2 FIR FRsdk::Population::get (const std::string & *name*) const

get the mapped FIR by name

0.36.38.3.3 Population& FRsdk::Population::operator= (const Population &)

assignment operator

0.36.38.3.4 void FRsdk::Population::remove (const std::string & *name*)

remove a FIR by name from the population

0.36.38.4 Friends And Related Function Documentation

0.36.38.4.1 friend class Implementation [friend]

The documentation for this class was generated from the following file:

- frsdk/fir.h

0.36.39 FRsdk::Position Class Reference

explicit template instantiation for win32

`#include <position.h>`

Public Member Functions

- Position (const float &x, const float &y)

  c'tor which constructs from continuous coordinates
- Position (const Position &rhs)

  copy c'tor
- float x () const

  returns the x coordinate
- float y () const

  returns the y coordinate

0.36.39.1 Detailed Description

explicit template instantiation for win32

continuous two-dimensional image coordinates

A Position describes continuous two-dimensional image coordinates. In FRsdk context this is used to describe locations in digital images taken for face recognition.

The convention to convert between discrete (x') and continuous (x) coordinates is:

- x' = int(floor(x))
- x = x' + 0.5f

The coordinate system used for all images in FRsdk context has the origin (0,0) in the upper left, with the x-axis extending to the right and y-axis extending downwards. Position's x and y coordinates are relative to this origin.

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, eyesfind.cc, idialog.h, and vdialog.h.

0.36.39.2 Constructor & Destructor Documentation

0.36.39.2.1 FRsdk::Position::Position (const float & *x*, const float & *y*) [inline]

c'tor which constructs from continuous coordinates

0.36.39.2.2 FRsdk::Position::Position (const Position & *rhs*) [inline]

copy c'tor

0.36.39.3 Member Function Documentation

0.36.39.3.1 float FRsdk::Position::x ( ) const [inline]

returns the x coordinate

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, eyesfind.cc, idialog.h, tracklife.cc, trackrec.cc, and vdialog.h.

0.36.39.3.2 float FRsdk::Position::y ( ) const [inline]

returns the y coordinate

Examples:

acquisition.cc, cropfullfrontal.cc, edialog.h, eyesfind.cc, idialog.h, tracklife.cc, trackrec.cc, and vdialog.h.

The documentation for this class was generated from the following file:

- frsdk/position.h

0.36.40 FRsdk::Enrollment::Processor Class Reference

this class represents the interface to the enrollment process Calls of process() are serialized but using multiple Processors enrollements can be done in parallel (one processor per thread).

`#include <enroll.h>`

Public Member Functions

- Processor (const Configuration &)

  create an instance of class Processor
- Processor (const Processor &)
- Processor & operator= (const Processor &)
- ∼Processor ()
- void process (const CaptureDevice &capDev, const Feedback &fb)

  Stream enrollment.
- void process (const SampleSet::const_iterator &begin, const SampleSet::const_iterator &end, const Feedback &fb)

  Sample Set enrollment.
- FIR merge (const FIR &, const FIR &)

  Merging of 2 FIRs to create a new one, which combines biometric features of both.

0.36.40.1 Detailed Description

this class represents the interface to the enrollment process Calls of process() are serialized but using multiple Processors enrollements can be done in parallel (one processor per thread).

MT-safe (serialized) It is safe to call the member functions concurrently from different threads.

Examples:

enroll.cc.

0.36.40.2 Constructor & Destructor Documentation

0.36.40.2.1 FRsdk::Enrollment::Processor::Processor (const Configuration &)

create an instance of class Processor

0.36.40.2.2 FRsdk::Enrollment::Processor::Processor (const Processor &)

0.36.40.2.3 FRsdk::Enrollment::Processor::∼Processor ( )

0.36.40.3 Member Function Documentation

0.36.40.3.1 FIR FRsdk::Enrollment::Processor::merge (const FIR & , const FIR &)

Merging of 2 FIRs to create a new one, which combines biometric features of both.

When using this function, take into account that the "capacity" of an FIR is limited and that using an FIR merged from a set of FIR's can result in lower biometric performance than would result from combining results obtained from the single FIR's.

0.36.40.3.2 Processor& FRsdk::Enrollment::Processor::operator= (const Processor &)

0.36.40.3.3 void FRsdk::Enrollment::Processor::process (const CaptureDevice & *capDev*, const Feedback & *fb*)

Stream enrollment.

The processor tries to capture the required (configurable) number of images, analyse them and process a enrollment. If the processing time is over the timeout the enrollment fails. To locate the faces in the captured sample images the Face::Finder optimized for single face finding is used.

Parameters

capDev	the device for capture images
fb	the feedback observing processing and returning results

0.36.40.3.4 void FRsdk::Enrollment::Processor::process (const SampleSet::const_iterator & *begin*, const SampleSet::const_iterator & *end*, const Feedback & *fb*)

Sample Set enrollment.

The processor takes the required (configurable) number of Samples containing 2D images of a face and, optionally, its eye positions and shape image, and creates the FIR. Samples which cannot be processed are ignored. For samples not providing eye positions the Face::Finder is used to locate the eyes.

Parameters

begin	start iterator for the enrollment image set
end	iterator designating the position after the last image of the enrollment image set
fb	the feedback observing processing and returning results

The documentation for this class was generated from the following file:

- frsdk/enroll.h

0.36.41 FRsdk::Identification::Processor Class Reference

this class represents the interface to the identification process.

```
#include <ident.h>
```

Public Member Functions

- Processor (const Configuration &, const Population &)

  Create an instance of class Processor.
- Processor (const Processor &)
- Processor & operator= (const Processor &)
- ~Processor ()
- void process (const CaptureDevice &capDev, const Score &threshold, const Feedback &feedback, unsigned int maxMatches=1)

  Stream identification.
- void process (const SampleSet::const_iterator &begin, const SampleSet::const_iterator &end, const Score &threshold, const Feedback &feedback, unsigned int maxMatches=1)

  Identification with a set of Samples.

0.36.41.1 Detailed Description

this class represents the interface to the identification process.

Calls of process() are serialized but using multiple Processors identifications can be done in parallel (one processor per thread).

MT-safe (serialized) It is safe to call the member functions concurrently from different threads.

Examples:

identify.cc.

0.36.41.2 Constructor & Destructor Documentation

0.36.41.2.1 FRsdk::Identification::Processor::Processor (const Configuration & , const Population &)

Create an instance of class Processor.

Multiple instances can be created with different NamedFIRSet's.

0.36.41.2.2 FRsdk::Identification::Processor::Processor (const Processor &)

0.36.41.2.3 FRsdk::Identification::Processor::∼Processor ( )

0.36.41.3 Member Function Documentation

0.36.41.3.1 Processor& FRsdk::Identification::Processor::operator= (const Processor &)

0.36.41.3.2 void FRsdk::Identification::Processor::process (const CaptureDevice & *capDev*, const Score & *threshold*, const Feedback & *feedback*, unsigned int *maxMatches* = 1)

Stream identification.

The processor grabs continously images from the capture device and tries to identify the face in the image if one. The process cancels after a configurable timeout if no face was found or no person could be identified. To locate the faces in the captured sample images the Face::Finder is used.

Parameters

capDev	the device for capturing images
threshold	threshold for match decision
feedback	the feedback to observe processing and returning results to the application
maxMatches	the maximum size of the FRsdk::Matches to be returned in the feedback

0.36.41.3.3 void FRsdk::Identification::Processor::process (const SampleSet::const_iterator & *begin*, const SampleSet::const_iterator & *end*, const Score & *threshold*, const Feedback & *feedback*, unsigned int *maxMatches* = 1)

Identification with a set of Samples.

The processor iterates the sample set and tries to identify each sample. Once a sample could successfully be identified processing will be finished and returns the match list (via feedback). Not processed images are ignored. For samples not providing eye positions the Face::Finder is used to locate the eyes.

Parameters

begin	start iterator for the identification sample set
end	iterator designating the position after the last sample of the identification sample set
threshold	threshold for match decision
feedback	the feedback to observe processing and returning results to the application
maxMatches	the maximum size of the FRsdk::Matches to be returned in the feedback

The documentation for this class was generated from the following file:

- frsdk/ident.h

0.36.42 FRsdk::Verification::Processor Class Reference

this class represents the interface to the verification process Calls of process() are serialized but using multiple Processors verifications can be done in parallel (one processor per thread).

```
#include <verify.h>
```

Public Member Functions

- Processor (const Configuration &)

    create an instance of class Processor
- Processor (const Processor &)
- Processor & operator= (const Processor &)

- ∼Processor ()
- void process (const CaptureDevice &capDev, const FIR &fir, const Score &threshold, const Feedback &feedback)

    Stream verification.
- void process (const SampleSet::const_iterator &begin, const SampleSet::const_iterator &end, const FIR &fir, const Score &threshold, const Feedback &feedback)

    Verification from a set of samples.

0.36.42.1 Detailed Description

this class represents the interface to the verification process Calls of process() are serialized but using multiple Processors verifications can be done in parallel (one processor per thread).

MT-safe (serialized) It is safe to call the member functions concurrently from different threads.

Examples:

verify.cc, and verifyan.cc.

0.36.42.2 Constructor & Destructor Documentation

0.36.42.2.1 FRsdk::Verification::Processor::Processor (const Configuration &)

create an instance of class Processor

0.36.42.2.2 FRsdk::Verification::Processor::Processor (const Processor &)

0.36.42.2.3 FRsdk::Verification::Processor::∼Processor ( )

0.36.42.3 Member Function Documentation

0.36.42.3.1 Processor& FRsdk::Verification::Processor::operator= (const Processor &)

0.36.42.3.2 void FRsdk::Verification::Processor::process (const CaptureDevice & *capDev,* const FIR & *fir,* const Score & *threshold,* const Feedback & *feedback*)

Stream verification.

The processor continuously fetches images from the CaptureDevice and tries to verify the face in the image against the FIR passed upon Processor construction. Processing terminates

- once a sample could have been verified
- after a configurable timeout if none of the samples was successfully verified so far
- after a configurable number of samples has been processed

whatever happens first.

To locate the faces in the captured samples the Face::Finder is used.

Parameters

capDev	the device to capture images

fir	FIR to be used for verification
threshold	the threshold for verification success decision
feedback	the feedback for observing processing and returning results to the application

0.36.42.3.3 void FRsdk::Verification::Processor::process (const SampleSet::const_iterator & *begin*, const SampleSet::const_iterator & *end*, const FIR & *fir*, const Score & *threshold*, const Feedback & *feedback*)

Verification from a set of samples.

The processor iterates the given SampleSet and tries to verify each sample against the FIR passed upon Processor construction. Processing terminates either with the first sample which could successfully be verified or after the last sample has been processed. For samples not providing eye positions the Face::Finder is used to locate the eyes.

Parameters

begin	start iterator for the verification sample set
end	iterator designating the position after the last sample of the verification SampleSet
fir	FIR to be used for verification
threshold	the threshold for verification success decision
feedback	the feedback for observing processing and returning results to the application

The documentation for this class was generated from the following file:

- frsdk/verify.h

0.36.43 FRsdk::Jpeg::Properties Struct Reference

contains properties of a JPEG image

`#include <jpeg.h>`

Public Member Functions

- Properties (int fileSize_, int quality_)

Public Attributes

- int fileSize

  the file size
- int quality

  the JPEG quality

0.36.43.1 Detailed Description

contains properties of a JPEG image

0.36.43.2 Constructor & Destructor Documentation

0.36.43.2.1 FRsdk::Jpeg::Properties::Properties (int *fileSize_*, int *quality_*) [inline]

0.36.43.3 Member Data Documentation

0.36.43.3.1 int FRsdk::Jpeg::Properties::fileSize

the file size

0.36.43.3.2 int FRsdk::Jpeg::Properties::quality

the JPEG quality

The documentation for this struct was generated from the following file:

- frsdk/jpeg.h

0.36.44 FRsdk::ImageIO::PropertiesFeedback Class Reference

explicit template instantiation for win32

`#include <image.h>`

Public Member Functions

- PropertiesFeedback ()

  body/handle supporting c'tor.
- PropertiesFeedback (const CountedPtr< FRsdk::ImageIO::PropertiesFeedbackBody > &i)
- void compressionMode (const ImageIO::ColorMode &cm)

  the given Parameter tells about the original coding/compression of the image data.
- void pixelDepth (unsigned int pd)

  get the pixel depth in bit per pixel.

0.36.44.1 Detailed Description

explicit template instantiation for win32

Examples:

acquisition.cc.

0.36.44.2 Constructor & Destructor Documentation

0.36.44.2.1 FRsdk::ImageIO::PropertiesFeedback::PropertiesFeedback ( )

body/handle supporting c'tor.

0.36.44.2.2 FRsdk::ImageIO::PropertiesFeedback::PropertiesFeedback (const CountedPtr< FRsdk::ImageIO::PropertiesFeedbackBody > & *i*) [inline]

0.36.44.3 Member Function Documentation

0.36.44.3.1 void FRsdk::ImageIO::PropertiesFeedback::compressionMode (const ImageIO::ColorMode & *cm*) [inline]

the given Parameter tells about the original coding/compression of the image data.

0.36.44.3.2 void FRsdk::ImageIO::PropertiesFeedback::pixelDepth (unsigned int *pd*) [inline]

get the pixel depth in bit per pixel.

In color mode the different channels precisions will be accumulated.

The documentation for this class was generated from the following file:

- frsdk/image.h

0.36.45 FRsdk::ImageIO::PropertiesFeedbackBody Class Reference

abstract PropertiesFeedback body

`#include <image.h>`

Public Member Functions

- virtual ~PropertiesFeedbackBody ()
- virtual void compressionMode (const ColorMode &)=0

    the given Parameter tells about the original coding/compression of the image data.
- virtual void pixelDepth (unsigned int)=0

    get the pixel depth in bit per pixel.

0.36.45.1 Detailed Description

abstract PropertiesFeedback body

Examples:

acquisition.cc.

0.36.45.2 Constructor & Destructor Documentation

0.36.45.2.1 virtual FRsdk::ImageIO::PropertiesFeedbackBody::~PropertiesFeedbackBody ( ) [inline], [virtual]

0.36.45.3 Member Function Documentation

0.36.45.3.1 virtual void FRsdk::ImageIO::PropertiesFeedbackBody::compressionMode (const ColorMode &) [pure virtual]

the given Parameter tells about the original coding/compression of the image data.

0.36.45.3.2 virtual void FRsdk::ImageIO::PropertiesFeedbackBody::pixelDepth (unsigned *int*) [pure virtual]

get the pixel depth in bit per pixel.

In color mode the different channels precisions will be accumulated.

The documentation for this class was generated from the following file:

- frsdk/image.h

0.36.46 FRsdk::Rgb Struct Reference

Red, green and blue color model.

`#include <types.h>`

Public Member Functions

- Rgb (const Byte &r_, const Byte &g_, const Byte &b_)

    construction from red, green and blue
- Rgb ()

    default constructor

Public Attributes

- Byte b
  red
- Byte g
  green
- Byte r
  blue
- Byte a
  alpha

0.36.46.1 Detailed Description

Red, green and blue color model.

The three channels can have values from 0..255.

Examples:

vignetting.cc.

0.36.46.2 Constructor & Destructor Documentation

0.36.46.2.1 FRsdk::Rgb::Rgb (const Byte & *r_*, const Byte & *g_*, const Byte & *b_*) `[inline]`

construction from red, green and blue

0.36.46.2.2 FRsdk::Rgb::Rgb ( ) `[inline]`

default constructor

0.36.46.3 Member Data Documentation

0.36.46.3.1 Byte FRsdk::Rgb::a

alpha

0.36.46.3.2 Byte FRsdk::Rgb::b

red

0.36.46.3.3 Byte FRsdk::Rgb::g

green

0.36.46.3.4 Byte FRsdk::Rgb::r

blue

The documentation for this struct was generated from the following file:

- frsdk/types.h

0.36.47 FRsdk::Sample Class Reference

Data sample containing mandatory intensity image and optional shape data and eye positions.

`#include <sample.h>`

Classes

- class ItemNotSet

  Specific exception type indicating access to optional data not present in Sample.
- struct Vector3D

Public Member Functions

- Sample (const Image &intensityImage_)

  Construct from intensity image only.
- Sample (const Image &intensityImage_, const ShapeImage &shapeImage_)

  Construct from intensity image and shape image.
- Sample (const AnnotatedImage &a)

  Construct from intensity image and eyes annotation.
- void annotate (const Eyes::Location &eyes_)

  Add eyes annotation.
- bool annotated () const

  Ask for availabity of eyes annotation.
- const Eyes::Location & eyeLocations () const

  Retrieve eye annotations.
- const Image & intensityImage () const

  Retrieve intensity image.
- bool hasShapeImage () const

  Ask for availabity of shape image.
- const ShapeImage & shapeImage () const

  Retrieve shape image.
- Sample (const Image &intensityImage_, const PointSet &ps_)
- Sample (const Image &intensityImage_, const PointSet &ps_, const Vector3D &v0, const Vector3D &v1)
- bool hasPointSet () const
- const PointSet & pointSet () const
- void annotate3D (const Vector3D &v0, const Vector3D &v1)
- bool annotated3D () const
- const Vector3D & getEye03D () const
- const Vector3D & getEye13D () const

0.36.47.1 Detailed Description

Data sample containing mandatory intensity image and optional shape data and eye positions.

Examples:

enroll.cc, identify.cc, verify.cc, and verifyan.cc.

0.36.47.2 Constructor & Destructor Documentation

0.36.47.2.1 FRsdk::Sample::Sample (const Image & *intensityImage_*) [inline]

Construct from intensity image only.

0.36.47.2.2 FRsdk::Sample::Sample (const Image & *intensityImage_*, const ShapeImage & *shapeImage_*) [inline]

Construct from intensity image and shape image.

0.36.47.2.3 FRsdk::Sample::Sample (const AnnotatedImage & *a*) [inline]

Construct from intensity image and eyes annotation.

0.36.47.2.4 FRsdk::Sample::Sample (const Image & *intensityImage_*, const PointSet & *ps_*) [inline]

0.36.47.2.5 FRsdk::Sample::Sample (const Image & *intensityImage_*, const PointSet & *ps_*, const Vector3D & *v0*, const Vector3D & *v1*) [inline]

0.36.47.3 Member Function Documentation

0.36.47.3.1 void FRsdk::Sample::annotate (const Eyes::Location & *eyes_*) [inline]

Add eyes annotation.

0.36.47.3.2 void FRsdk::Sample::annotate3D (const Vector3D & *v0*, const Vector3D & *v1*) [inline]

0.36.47.3.3 bool FRsdk::Sample::annotated ( ) const [inline]

Ask for availabity of eyes annotation.

0.36.47.3.4 bool FRsdk::Sample::annotated3D ( ) const [inline]

0.36.47.3.5 const Eyes::Location& FRsdk::Sample::eyeLocations ( ) const [inline]

Retrieve eye annotations.

Will throw an 'ItemNotSet' exception, if eye annotations are not present

0.36.47.3.6 const Vector3D& FRsdk::Sample::getEye03D ( ) const [inline]

0.36.47.3.7 const Vector3D& FRsdk::Sample::getEye13D ( ) const [inline]

0.36.47.3.8 bool FRsdk::Sample::hasPointSet ( ) const [inline]

0.36.47.3.9 bool FRsdk::Sample::hasShapeImage ( ) const [inline]

Ask for availabity of shape image.

0.36.47.3.10 const Image& FRsdk::Sample::intensityImage ( ) const [inline]

Retrieve intensity image.

0.36.47.3.11 const PointSet& FRsdk::Sample::pointSet ( ) const [inline]

0.36.47.3.12 const ShapeImage& FRsdk::Sample::shapeImage ( ) const [inline]

Retrieve shape image.

Will throw an 'ItemNotSet' exception,if no shape image present

The documentation for this class was generated from the following file:

- frsdk/sample.h

0.36.48 FRsdk::Score Class Reference

This class represents a score for representing the comparison result between a FIR and the biometric evidence.

`#include <score.h>`

Public Member Functions

- Score (const float &)

    Construct a score value from a float.
- Score & operator= (const Score &s)
- operator float () const

    convert the score value to a float
- Score (const class RawScore &)

0.36.48.1 Detailed Description

This class represents a score for representing the comparison result between a FIR and the biometric evidence.

The range of scores returned by the algorithms is between 0.0f and 1.0f (except for the Composite Comparison Algorithm). Applications may use `requestFAR()` or `requestFRR()` to obtain score values corresponding to requested values of FAR or FRR. Score values can be used as thresholds for verifications and identifications and will be communicated as results of these operations.

Examples:

identify.cc, match.cc, vdialog.h, verify.cc, and verifyan.cc.

0.36.48.2 Constructor & Destructor Documentation

0.36.48.2.1 FRsdk::Score::Score (const float &)

Construct a score value from a float.

0.36.48.2.2 FRsdk::Score::Score (const class RawScore &)

0.36.48.3 Member Function Documentation

0.36.48.3.1 FRsdk::Score::operator float ( ) const

convert the score value to a float

0.36.48.3.2 Score& FRsdk::Score::operator= (const Score & *s*)

The documentation for this class was generated from the following file:

- frsdk/score.h

0.36.49 FRsdk::ScoreMappings Class Reference

FAR,FRR / Score mappings.

`#include <score.h>`

Public Member Functions

- ScoreMappings (const Configuration &)
- ScoreMappings (const ScoreMappings &)
- ScoreMappings & operator= (const ScoreMappings &)
- ∼ScoreMappings ()

- Score requestFAR (float requestedFAR)

Get a score value corresponding to the requested value of FAR.
- Score requestFRR (float requestedFRR)

  Get a score value corresponding to the requested value of FRR.
- float expectedFAR (const Score &threshold)

  Get a value for the False Acceptance Rate that would be achieved if a given score was used as the threshold.
- float expectedFRR (const Score &threshold)

  Get a value for the False Rejection Rate that would be achieved if a given score was used as the threshold.

0.36.49.1 Detailed Description

FAR,FRR / Score mappings.

Examples:

identify.cc, match.cc, verify.cc, and verifyan.cc.

0.36.49.2 Constructor & Destructor Documentation

0.36.49.2.1 FRsdk::ScoreMappings::ScoreMappings (const Configuration &)

0.36.49.2.2 FRsdk::ScoreMappings::ScoreMappings (const ScoreMappings &)

0.36.49.2.3 FRsdk::ScoreMappings::~ScoreMappings ( )

0.36.49.3 Member Function Documentation

0.36.49.3.1 float FRsdk::ScoreMappings::expectedFAR (const Score & *threshold*)

Get a value for the False Acceptance Rate that would be achieved if a given score was used as the threshold.

Parameters

threshold	score used as threshold

0.36.49.3.2 float FRsdk::ScoreMappings::expectedFRR (const Score & *threshold*)

Get a value for the False Rejection Rate that would be achieved if a given score was used as the threshold.

Parameters

threshold	score used as threshold

0.36.49.3.3 ScoreMappings& FRsdk::ScoreMappings::operator= (const ScoreMappings &)

0.36.49.3.4 Score FRsdk::ScoreMappings::requestFAR (float *requestedFAR*)

Get a score value corresponding to the requested value of FAR.

This value may be used as a threshold for verification and identification operations.

Parameters

requestedFAR	The maximum value for the rate of false acceptances the application wishes to allow.

0.36.49.3.5 Score FRsdk::ScoreMappings::requestFRR (float *requestedFRR*)

Get a score value corresponding to the requested value of FRR.

This value may be used as a threshold for verification and identification operations.

Parameters

requestedFRR	The maximum value for the rate of false rejections the application wishes to allow.

The documentation for this class was generated from the following file:

- frsdk/score.h

0.36.50 FRsdk::Portrait::Feature::Set Class Reference

Feature assessment results.

`#include <portraittests.h>`

Public Member Functions

- Set (const Set &)
- Set & operator= (const Set &)
- ∼Set ()
- bool wearsGlasses () const

  Returns true if the person wears eyeglasses.
- Gender gender () const

  Returns the gender of the person.
- Ethnicity ethnicity () const

  return the ethnicity of person

Friends

- class Test

0.36.50.1 Detailed Description

Feature assessment results.

An instance of this class represents portrait features detected by Portrait::Feature::Test.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.50.2 Constructor & Destructor Documentation

0.36.50.2.1 FRsdk::Portrait::Feature::Set::Set (const Set &)

0.36.50.2.2 FRsdk::Portrait::Feature::Set::∼Set ( )

0.36.50.3 Member Function Documentation

0.36.50.3.1 Ethnicity FRsdk::Portrait::Feature::Set::ethnicity ( ) const

return the ethnicity of person

Examples:

acquisition.cc.

0.36.50.3.2 Gender FRsdk::Portrait::Feature::Set::gender ( ) const

Returns the gender of the person.

0.36.50.3.3 Set& FRsdk::Portrait::Feature::Set::operator= (const Set &)

0.36.50.3.4 bool FRsdk::Portrait::Feature::Set::wearsGlasses ( ) const

Returns true if the person wears eyeglasses.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.50.4 Friends And Related Function Documentation

0.36.50.4.1 friend class Test [friend]

The documentation for this class was generated from the following file:

- frsdk/portraittests.h

0.36.51 FRsdk::ShapeImage Class Reference

explicit template instantiation for win32

`#include <shapeimage.h>`

Public Member Functions

- ShapeImage (const CountedPtr< ShapeImageBody > &s)

  build an image handle from an abstract implementation
- unsigned int width () const

  returns the width of the image in pixels
- unsigned int height () const

  returns the height of the image n pixels
- const Vertex * vertices () const

  *Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Vertex) containing the shape image vertices.*
- const bool * mask () const

  *Returns a pointer to an array of size width() * height() * sizeof(bool) containing a boolean mask indicating which elements of the vertex array contain valid vertices.*
- ShapeImageBody & body () const

  Access to the body.

0.36.51.1 Detailed Description

explicit template instantiation for win32

ShapeImage handle (interface)

0.36.51.2 Constructor & Destructor Documentation

0.36.51.2.1 FRsdk::ShapeImage::ShapeImage (const CountedPtr< ShapeImageBody > & *s*)

build an image handle from an abstract implementation

0.36.51.3 Member Function Documentation

0.36.51.3.1 ShapeImageBody& FRsdk::ShapeImage::body ( ) const [inline]

Access to the body.

This enables clients to perform (dynamic) casting to the concrete body class.

0.36.51.3.2 unsigned int FRsdk::ShapeImage::height ( ) const [inline]

returns the height of the image n pixels

0.36.51.3.3 const bool* FRsdk::ShapeImage::mask ( ) const [inline]

Returns a pointer to an array of size width() * height() * sizeof(bool) containing a boolean mask indicating which elements of the vertex array contain valid vertices.

The pointer has to remain valid during the whole lifetime of the ShapeImageBody object.

0.36.51.3.4 const Vertex* FRsdk::ShapeImage::vertices ( ) const [inline]

Returns a pointer to an array of size width() * height() * sizeof(FRsdk::Vertex) containing the shape image vertices.

The pointer has to remain valid during the whole lifetime of the ShapeImageBody object.

0.36.51.3.5 unsigned int FRsdk::ShapeImage::width ( ) const [inline]

returns the width of the image in pixels

The documentation for this class was generated from the following file:

- frsdk/shapeimage.h

0.36.52 FRsdk::ShapeImageBody Class Reference

Abstract shape image body.

`#include <shapeimage.h>`

Public Member Functions

- virtual ~ShapeImageBody ()
- virtual unsigned int width () const =0

  returns the width of the image in pixels
- virtual unsigned int height () const =0

  returns the height of the image in pixels
- virtual const Vertex * vertices () const =0

  *Returns a pointer to an array of size width() * height() * sizeof(Vertex) containing the vertex representation of the shape image.*
- virtual const bool * mask () const =0

  *Returns a pointer to an array of size width() * height() * sizeof(bool) containing a boolean mask indicating which elements of the vertex array contain valid vertices.*

0.36.52.1 Detailed Description

Abstract shape image body.

0.36.52.2 Constructor & Destructor Documentation

0.36.52.2.1 virtual FRsdk::ShapeImageBody::∼ShapeImageBody ( ) [inline], [virtual]

0.36.52.3 Member Function Documentation

0.36.52.3.1 virtual unsigned int FRsdk::ShapeImageBody::height ( ) const [pure virtual]

returns the height of the image in pixels

0.36.52.3.2 virtual const bool∗ FRsdk::ShapeImageBody::mask ( ) const [pure virtual]

Returns a pointer to an array of size width() ∗ height() ∗ sizeof(bool) containing a boolean mask indicating which elements of the vertex array contain valid vertices.

The pointer has to remain valid during the whole lifetime of the ShapeImageBody object.

0.36.52.3.3 virtual const Vertex∗ FRsdk::ShapeImageBody::vertices ( ) const [pure virtual]

Returns a pointer to an array of size width() ∗ height() ∗ sizeof(Vertex) containing the vertex representation of the shape image.

Vertices are arranged in consecutive order within lines from left to right, starting with the topmost line. The pointer has to remain valid during the whole lifetime of the ShapeImageBody object.

0.36.52.3.4 virtual unsigned int FRsdk::ShapeImageBody::width ( ) const [pure virtual]

returns the width of the image in pixels

The documentation for this class was generated from the following file:

- frsdk/shapeimage.h

0.36.53 FRsdk::ISO_19794_5::FullFrontal::Test Class Reference

Compliance assessment.

`#include <portraittests.h>`

Public Member Functions

- Test (const Configuration &)
- Test (const Test &)
- Test & operator= (const Test &)
- ∼Test ()
- Compliance assess (const Portrait::Characteristics &) const

    Assess Portrait::Characteristics of a portrait according to the ISO 19794-5:2005 requiremenents.
- Boundaries boundaries () const

    Get class containing the boundaries (limits) used to assess portrait charateristics.

0.36.53.1 Detailed Description

Compliance assessment.

An instance of this class can be used to assess Portrait::Characteristics of a portrait to determine the compliance with the ISO 19794-5:2005 Full Frontal Image requirements.

MT-safe (reentrant) It is safe to call the class member functions

concurrently from different threads.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.53.2 Constructor & Destructor Documentation

0.36.53.2.1 FRsdk::ISO_19794_5::FullFrontal::Test::Test (const Configuration &)

0.36.53.2.2 FRsdk::ISO_19794_5::FullFrontal::Test::Test (const Test &)

0.36.53.2.3 FRsdk::ISO_19794_5::FullFrontal::Test::~Test ( )

0.36.53.3 Member Function Documentation

0.36.53.3.1 Compliance FRsdk::ISO_19794_5::FullFrontal::Test::assess (const Portrait::Characteristics &) const

Assess Portrait::Characteristics of a portrait according to the ISO 19794-5:2005 requiremenents.

0.36.53.3.2 Boundaries FRsdk::ISO_19794_5::FullFrontal::Test::boundaries ( ) const

Get class containing the boundaries (limits) used to assess portrait charateristics.

These boundaries are read only and can be modified only by using the configuration editor.

0.36.53.3.3 Test& FRsdk::ISO_19794_5::FullFrontal::Test::operator= (const Test &)

The documentation for this class was generated from the following file:

- frsdk/portraittests.h

0.36.54 FRsdk::Portrait::Feature::Test Class Reference

Test for features in a portrait.

`#include <portraittests.h>`

Public Member Functions

- Test (const Configuration &)
- Test (const Test &)
- Test & operator= (const Test &)
- ~Test ()
- Set assess (const Portrait::Characteristics &) const

  Assess Portrait::Characteristics of a portrait to determine features.

0.36.54.1 Detailed Description

Test for features in a portrait.

Use an instance of this class to test Portrait::Characteristics for existence of portrait features.

Hint: Use Portrait::Analyzer to get Portrait::Characteristics.

MT-safe (reentrant) It is safe to call the class member functions

concurrently from different threads.

Examples:

acquisition.cc, and cropfullfrontal.cc.

0.36.54.2 Constructor & Destructor Documentation

0.36.54.2.1 FRsdk::Portrait::Feature::Test::Test (const Configuration &)

0.36.54.2.2 FRsdk::Portrait::Feature::Test::Test (const Test &)

0.36.54.2.3 FRsdk::Portrait::Feature::Test::∼Test ( )

0.36.54.3 Member Function Documentation

0.36.54.3.1 Set FRsdk::Portrait::Feature::Test::assess (const Portrait::Characteristics &) const

Assess Portrait::Characteristics of a portrait to determine features.

0.36.54.3.2 Test& FRsdk::Portrait::Feature::Test::operator= (const Test &)

The documentation for this class was generated from the following file:

- frsdk/portraittests.h

0.36.55 FRsdk::Face::Tracker Class Reference

The Face Tracker locates and tracks faces across a sequence of images in an efficient way by analyzing the spatial and temporal dependencies between faces in subsequent images.

```
#include <tracker.h>
```

Classes

- struct Location
  The location of a face being tracked by the face tracker.

Public Types

- typedef std::list< Location > Locations
  A collection of tracked faces.

Public Member Functions

- Tracker (const Configuration &)
  Create tracker from configuration.
- Tracker (const Tracker &)
- Tracker & operator= (const Tracker &)
- ∼Tracker ()
- Locations processImage (const FRsdk::Image &img, const unsigned int &captureTime)
  Processes an image (usually a frame from a video stream) and returns the tracked faces with their eyes positions in this image.
- Locations processFrame (const FRsdk::Image &img)
  Processes single image, captured from a stream with constant frame rate.

0.36.55.1 Detailed Description

The Face Tracker locates and tracks faces across a sequence of images in an efficient way by analyzing the spatial and temporal dependencies between faces in subsequent images.

Examples:

tracklife.cc, and trackrec.cc.

0.36.55.2 Member Typedef Documentation

0.36.55.2.1 typedef std::list<Location> FRsdk::Face::Tracker::Locations

A collection of tracked faces.

0.36.55.3 Constructor & Destructor Documentation

0.36.55.3.1 FRsdk::Face::Tracker::Tracker (const Configuration &)

Create tracker from configuration.

0.36.55.3.2 FRsdk::Face::Tracker::Tracker (const Tracker &)

0.36.55.3.3 FRsdk::Face::Tracker::∼Tracker ( )

0.36.55.4 Member Function Documentation

0.36.55.4.1 Tracker& FRsdk::Face::Tracker::operator= (const Tracker &)

0.36.55.4.2 Locations FRsdk::Face::Tracker::processFrame (const FRsdk::Image & *img*)

Processes single image, captured from a stream with constant frame rate.

The frame rate can be configured.

0.36.55.4.3 Locations FRsdk::Face::Tracker::processImage (const FRsdk::Image & *img*, const unsigned int & *captureTime*)

Processes an image (usually a frame from a video stream) and returns the tracked faces with their eyes positions in this image.

captureTime is the capture time of the given frame img in milliseconds relative to a user defined reference point in the past (e.g. capture time of the first frame).

The tracker prediction of the face search areas in subsequent frames depends on the strong monotonic time line. If the capture time of an image passed to processImage() is equal to or earlier than the capture time of the image passed before the tracker id assigned to the face locatios may be wrong.

The documentation for this class was generated from the following file:

- frsdk/tracker.h

0.36.56 FRsdk::Sample::Vector3D Struct Reference

```
#include <sample.h>
```

Public Member Functions

- Vector3D (const Vector3D &v)
- Vector3D (const double &x_, const double &y_, const double &z_)

Public Attributes

- double x

- double y
- double z

0.36.56.1 Constructor & Destructor Documentation

0.36.56.1.1 FRsdk::Sample::Vector3D::Vector3D (const Vector3D & *v*) [inline]

0.36.56.1.2 FRsdk::Sample::Vector3D::Vector3D (const double & *x_*, const double & *y_*, const double & *z_*) [inline]

0.36.56.2 Member Data Documentation

0.36.56.2.1 double FRsdk::Sample::Vector3D::x

0.36.56.2.2 double FRsdk::Sample::Vector3D::y

0.36.56.2.3 double FRsdk::Sample::Vector3D::z

The documentation for this struct was generated from the following file:

- frsdk/sample.h

0.36.57 FRsdk::Vertex Struct Reference

```
#include <shapeimage.h>
```

Public Attributes

- double x
- double y
- double z

0.36.57.1 Member Data Documentation

0.36.57.1.1 double FRsdk::Vertex::x

0.36.57.1.2 double FRsdk::Vertex::y

0.36.57.1.3 double FRsdk::Vertex::z

The documentation for this struct was generated from the following file:

- frsdk/shapeimage.h

0.36.58 FRsdk::VideoFormat Class Reference

Opaque type representing a video format (resolution, bits per pixel, etc).

```
#include <capdev.h>
```

Public Member Functions

- VideoFormat (std::istream &i)

    create from istream
- void write (std::ostream &o) const

write to ostream

- VideoFormat (const VideoFormat &)
- VideoFormat & operator= (const VideoFormat &)
- ∼VideoFormat ()

Friends

- class Body
- class BuiltInCaptureDeviceImpl

0.36.58.1 Detailed Description

Opaque type representing a video format (resolution, bits per pixel, etc).

Please note: Since video format settings are highly device dependent, a video format created from one video device by getVideoFormat() might be inappropriate for another one. Even a video format created from a given device might be inappropriate to the same device under changed conditions (e.g. a changed frame rate).

0.36.58.2 Constructor & Destructor Documentation

0.36.58.2.1 FRsdk::VideoFormat::VideoFormat (std::istream & *i*)

create from istream

0.36.58.2.2 FRsdk::VideoFormat::VideoFormat (const VideoFormat &)

0.36.58.2.3 FRsdk::VideoFormat::∼VideoFormat ( )

0.36.58.3 Member Function Documentation

0.36.58.3.1 VideoFormat& FRsdk::VideoFormat::operator= (const VideoFormat &)

0.36.58.3.2 void FRsdk::VideoFormat::write (std::ostream & *o*) const

write to ostream

0.36.58.4 Friends And Related Function Documentation

0.36.58.4.1 friend class Body `[friend]`

0.36.58.4.2 friend class BuiltInCaptureDeviceImpl `[friend]`

The documentation for this class was generated from the following file:

- frsdk/capdev.h

0.37 File Documentation

0.37.1 doc/contribution.doc File Reference

0.37.2 doc/examplesintro.doc File Reference

0.37.3 doc/mainpage.doc File Reference

0.37.4 doc/oldrevs.doc File Reference

0.37.5 doc/redistributablepackaging.doc File Reference

0.37.6 doc/references.doc File Reference

0.37.7 doc/sdkfaq.doc File Reference

0.37.8 doc/tutorial.doc File Reference

0.37.9 doc/userguide.doc File Reference

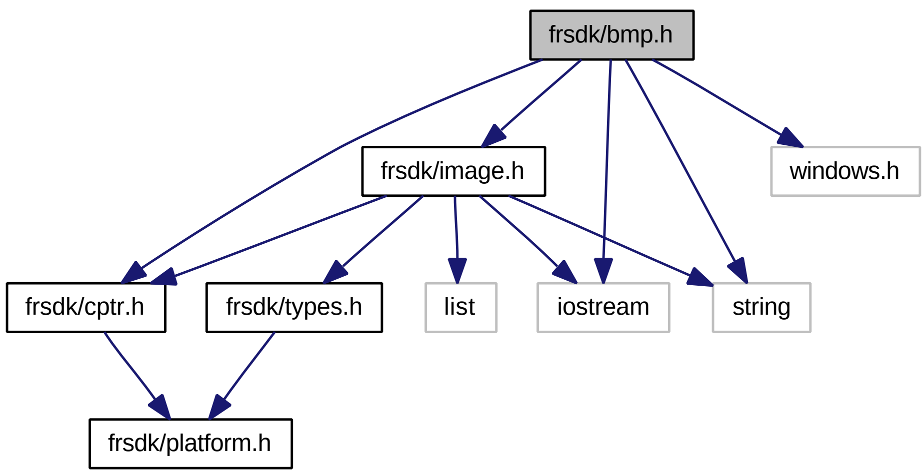
0.37.10 doc/whatsnew.doc File Reference

0.37.11 frsdk/bmp.h File Reference

Bitmap image format support.

```
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <string>
#include <iostream>
#include <windows.h>
```

Include dependency graph for bmp.h:



Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::Bmp

  Bitmap (BMP) image implementation.

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::Bmp

  Bitmap (BMP) image implementation.

Typedefs

- typedef struct tagBITMAPINFO BITMAPINFO

Functions

- Image FRsdk::Bmp::load (const BITMAPINFO ∗bi, const Byte ∗image, const std::string &name, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    constructs a image representation from the specified bitmap image in memory.
- Image FRsdk::Bmp::load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    load the Bmp image from file; the image will get the name of the file
- Image FRsdk::Bmp::load (std::istream &stream, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    load the Bmp image from a stream
- void FRsdk::Bmp::save (const Image &, const std::string &filename)

    saves the image to a file with the given file name.
- unsigned int FRsdk::Bmp::getBitmapInfoSize (const Image &img)

    get the total size of bitmap info; returns sizeof(BITMAPINFOHEADER)
- void FRsdk::Bmp::writeBitmapInfo (const Image &img, BITMAPINFO ∗info)

    write bitmap info to buffer; buffer size has to be at least the size returned by getBitmapInfoSize().

0.37.11.1 Detailed Description

Bitmap image format support.

0.37.11.2 Typedef Documentation

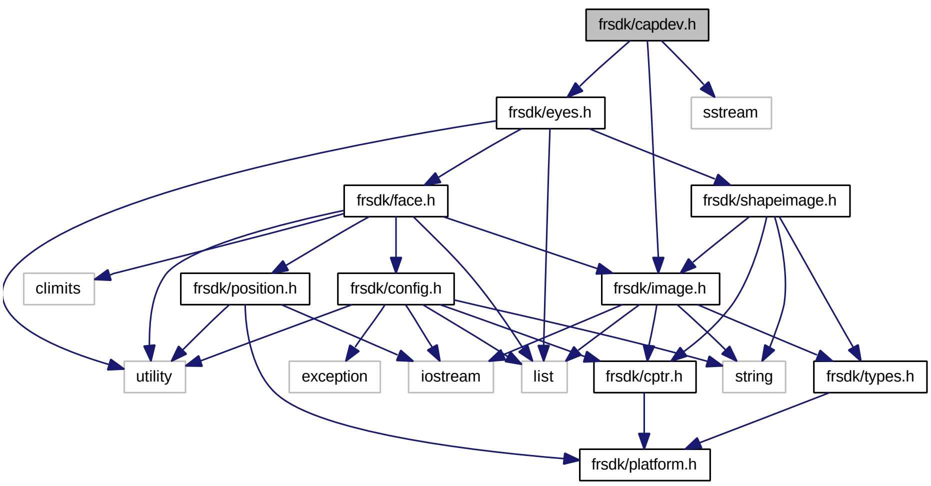
0.37.11.2.1 typedef struct tagBITMAPINFO **BITMAPINFO**

0.37.12 frsdk/capdev.h File Reference

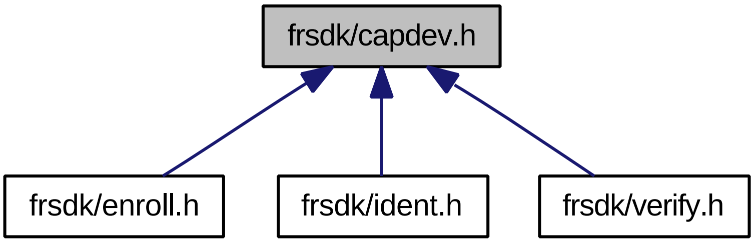
capture device abstraction

```
#include <frsdk/image.h>
#include <frsdk/eyes.h>
#include <sstream>
```

Include dependency graph for capdev.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::VideoFormat

    Opaque type representing a video format (resolution, bits per pixel, etc).
- class FRsdk::CaptureDeviceBody

    Abstract capture device body.
- class FRsdk::CaptureDevice

    Capture Device handle (interface)

Namespaces

- FRsdk

    The global name space for the SDK.

Constant Groups

- FRsdk

    The global name space for the SDK.

Functions

- CaptureDevice FRsdk::createCaptureDevice (const Configuration &, const std::string &Name)

    factory function for creating capture devices configured under FRSDK.CaptureDevices.Name using builtin capture device types a Camera Controller will be used if configured under FRSDK.CameraControllers.Name
- CaptureDevice FRsdk::createCaptureDevice (const CountedPtr< CaptureDeviceBody > &, const Configuration &, const std::string &camControlName)

    factory function for creating user defined capture devices using a Camera Controller configured under FRSDK.-CameraControllers.Name

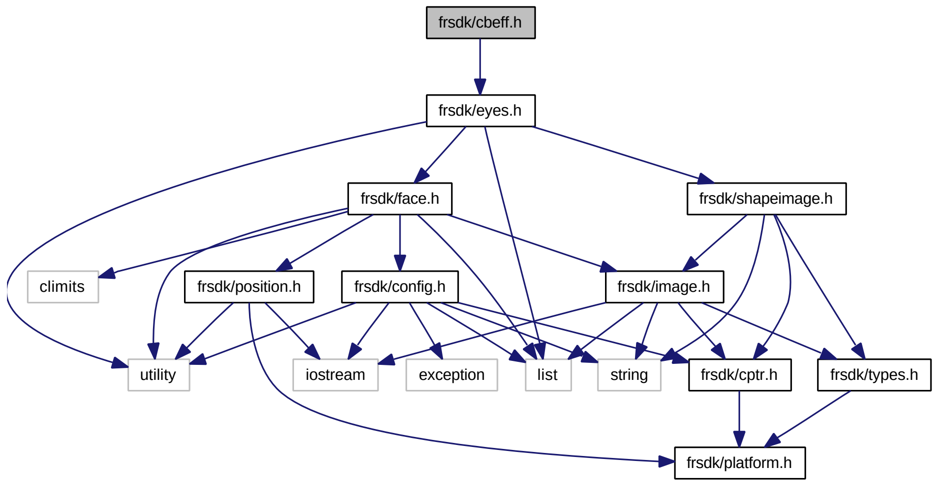
0.37.12.1 Detailed Description

capture device abstraction

0.37.13 frsdk/cbeff.h File Reference

Support for ISO/IEC 19794-5 CBEFF compliant Face Image I/O.

`#include <frsdk/eyes.h>`
Include dependency graph for cbeff.h:



Namespaces

- FRsdk

    The global name space for the SDK.
- FRsdk::ISO_19794_5

    Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::TokenFace

    Support for Token Face Image type as defined in ISO_19794_5 9.2.

Constant Groups

- FRsdk

    The global name space for the SDK.
- FRsdk::ISO_19794_5

    Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::TokenFace

    Support for Token Face Image type as defined in ISO_19794_5 9.2.

Functions

- AnnotatedImageSet FRsdk::ISO_19794_5::TokenFace::read (std::istream &i)

    TokenFace::read and TokenFace::write provide support for the ISO_19794_5, Common Biometric Exchange Formats Framework (CBEFF), Facial Data Interchange Format.
- void FRsdk::ISO_19794_5::TokenFace::write (std::ostream &o, const AnnotatedImageSet &)

    Use TokenFace::write for storing annotated faces to ISO_19794_5 CBEFF compliant Token Face Image type format to the stream.

0.37.13.1 Detailed Description

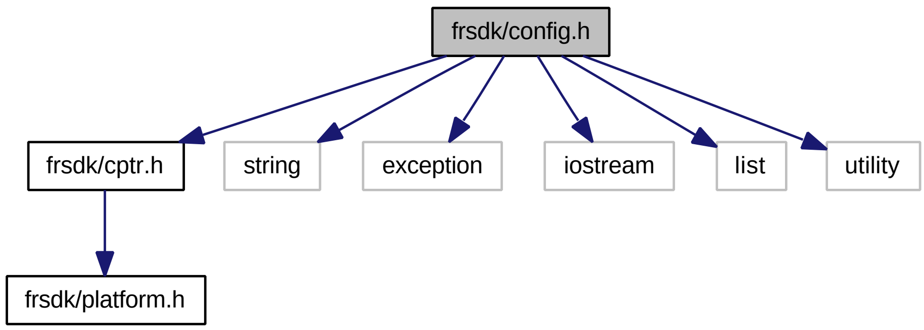
Support for ISO/IEC 19794-5 CBEFF compliant Face Image I/O.

0.37.14 frsdk/config.h File Reference

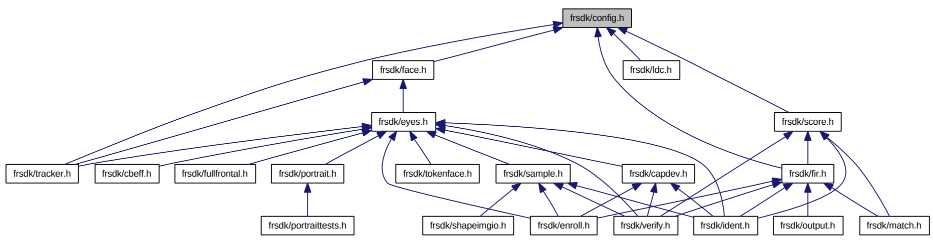
FaceVACS-SDK configuration support.

```
#include <frsdk/cptr.h>
#include <string>
#include <exception>
#include <iostream>
#include <list>
#include <utility>
```

Include dependency graph for config.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::LicenseSignatureMismatch

  License signature mismatch.
- class FRsdk::FeatureDisabled

  An object of this type is thrown at any time if requesting or accessing a disabled FaceVACS-SDK feature.
- class FRsdk::LimitExceeded

  An object of this type is thrown at any time if a configured limit of FaceVACS-SDK is exceeded.
- class FRsdk::Configuration

  Configuration object of the FaceVACS SDK library.

Namespaces

- FRsdk

  The global name space for the SDK.

Constant Groups

- FRsdk

  The global name space for the SDK.

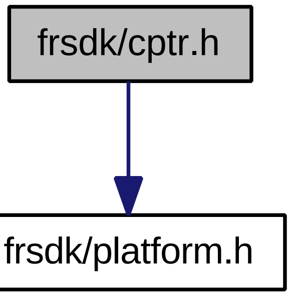
0.37.14.1 Detailed Description

FaceVACS-SDK configuration support.

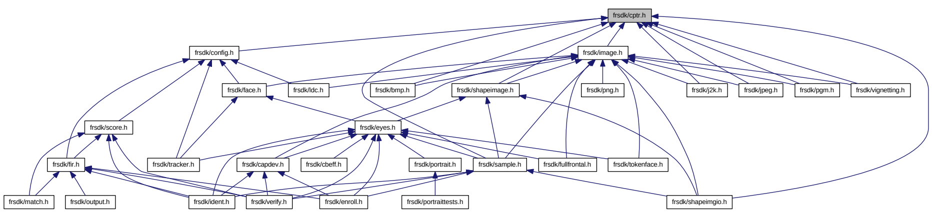
0.37.15 frsdk/cptr.h File Reference

counted pointer abstraction

`#include <frsdk/platform.h>`
Include dependency graph for cptr.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::CountedPtr< T >

  This class template implements the reference counting idiom simulating name semantics (see Scott Meyer's Bible, James Coplien's book).

- class FRsdk::CountedPtr< T >

  This class template implements the reference counting idiom simulating name semantics (see Scott Meyer's Bible, James Coplien's book).

Namespaces

- FRsdk

  The global name space for the SDK.

Constant Groups

- FRsdk

  The global name space for the SDK.

Typedefs

- typedef void * FRsdk::RefCountHandle

Functions

- RefCountHandle FRsdk::newRefcount (int initialValue)
- void FRsdk::deleteRefcount (RefCountHandle)
- void FRsdk::incRefcount (RefCountHandle)
- bool FRsdk::decAndTestRefcount (RefCountHandle)

0.37.15.1 Detailed Description

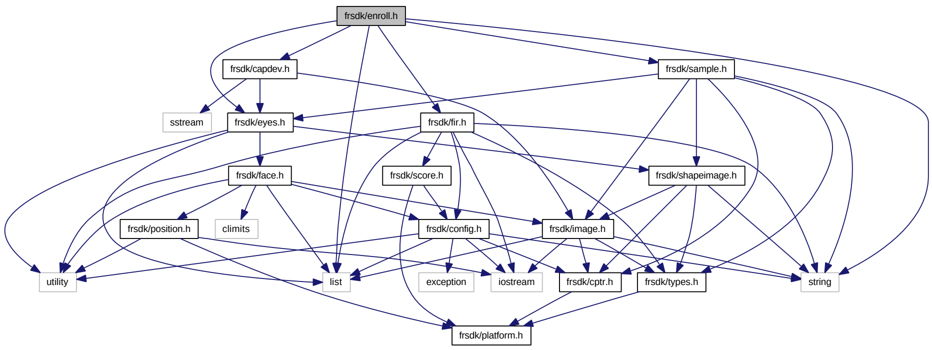
counted pointer abstraction

0.37.16 frsdk/enroll.h File Reference

enrollment use case support

```
#include <frsdk/capdev.h>
#include <frsdk/fir.h>
#include <frsdk/eyes.h>
#include <frsdk/sample.h>
#include <list>
#include <string>
```

Include dependency graph for enroll.h:



Classes

- class FRsdk::Enrollment::FeedbackBody

  Body class for Feedback.
- class FRsdk::Enrollment::Feedback

  explicit template instantiation for win32
- class FRsdk::Enrollment::Processor

  this class represents the interface to the enrollment process Calls of process() are serialized but using multiple Processors enrollements can be done in parallel (one processor per thread).

Namespaces

- FRsdk

  The global name space for the SDK.

- FRsdk::Enrollment

    the namespace for the enrollment facility

Constant Groups

- FRsdk

    The global name space for the SDK.
- FRsdk::Enrollment

    the namespace for the enrollment facility

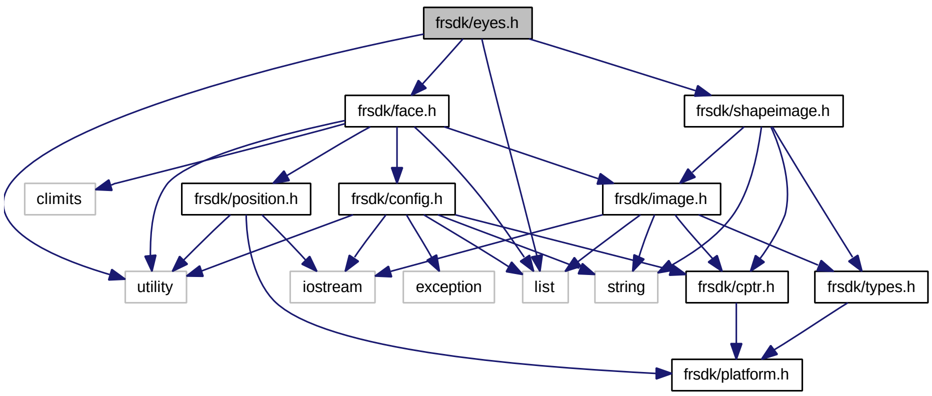
0.37.16.1 Detailed Description

enrollment use case support

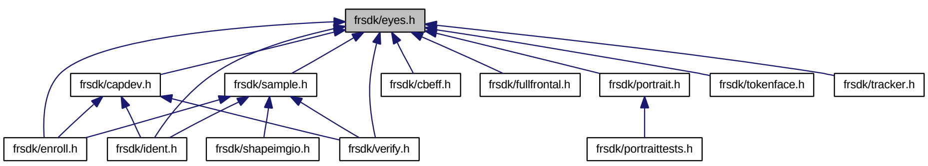
0.37.17 frsdk/eyes.h File Reference

eyes finding use case support

```
#include <frsdk/face.h>
#include <frsdk/shapeimage.h>
#include <list>
#include <utility>
```
Include dependency graph for eyes.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FRsdk::Eyes::Location

    The Eyes::Location describes a location in the image where eyes within a face have been found.
- class FRsdk::Eyes::Finder

    Eyes::Finder (handle) This class represents a interface to the eye finding procedure.

Namespaces

- FRsdk

    The global name space for the SDK.
- FRsdk::Eyes

    the name space for the eyes finding facility

Constant Groups

- FRsdk

    The global name space for the SDK.
- FRsdk::Eyes

    the name space for the eyes finding facility

Typedefs

- typedef std::list< Location > FRsdk::Eyes::LocationSet

    common used set of Locations
- typedef std::pair< Image,
  Eyes::Location > FRsdk::AnnotatedImage

    An image with annotated eye positions.
- typedef std::list< AnnotatedImage > FRsdk::AnnotatedImageSet

    A set of annotated images.

0.37.17.1 Detailed Description

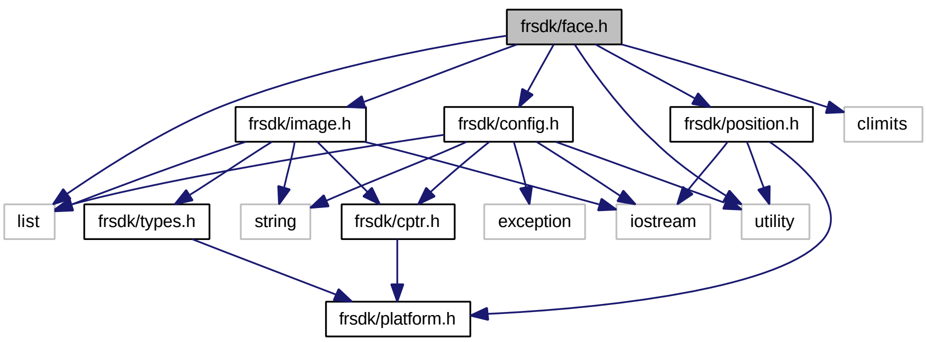
eyes finding use case support

0.37.18 frsdk/face.h File Reference

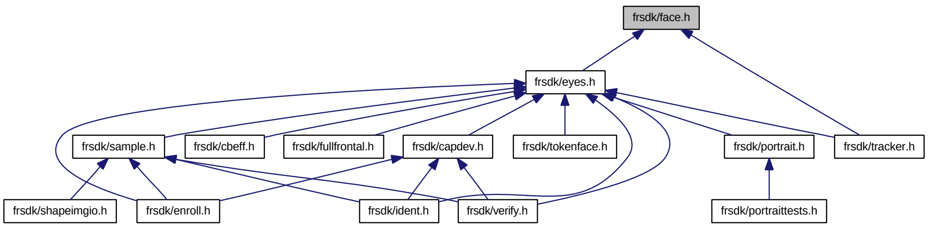
face finding use case support

```
#include <list>
#include <utility>
#include <climits>
#include <frsdk/config.h>
#include <frsdk/image.h>
#include <frsdk/position.h>
```

Include dependency graph for face.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FRsdk::Face::Location

    The Face::Location describes a image location where a face was found.

- class FRsdk::Face::Finder

    Face::Finder (handle) This class represents a interface to the face finding procedure.

Namespaces

- FRsdk

    The global name space for the SDK.

- FRsdk::Face

    the name space for the face finding facility

Constant Groups

- FRsdk

    The global name space for the SDK.

- FRsdk::Face

    the name space for the face finding facility

Typedefs

- typedef std::list< Location > FRsdk::Face::LocationSet

    common used set of Locations

Functions

- Position FRsdk::faceToLeftEyePos (const Position &facePosition, float eyeDistance, float faceRollAngle)

    Estimates the left eye position from the face position, face roll angle and eye distance.

- Position FRsdk::faceToRightEyePos (const Position &facePosition, float eyeDistance, float faceRollAngle)

    Estimates the right eye position from the face position, face roll angle and eye distance.

- void FRsdk::faceToEyesPos (const Position &facePosition, float faceRollAngle, float eyeDistance, Position &leftEyePosition, Position &rightEyePosition)

    Estimates the both eye positions from the face position, face roll angle and eye distance.

0.37.18.1 Detailed Description

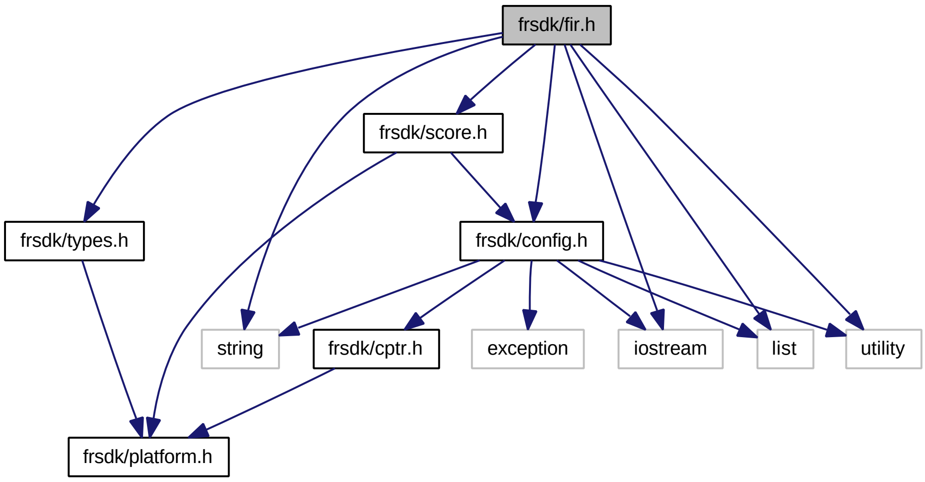
face finding use case support

0.37.19 frsdk/fir.h File Reference

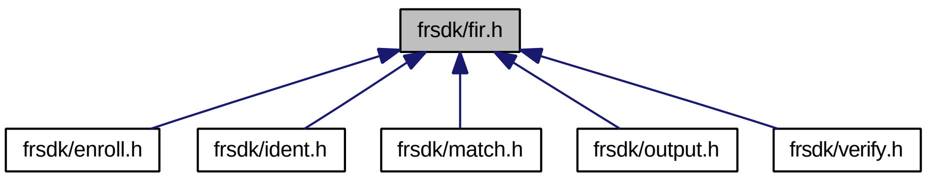
FIR (facial identification record) abstraction.

```
#include <frsdk/config.h>
#include <frsdk/score.h>
#include <frsdk/types.h>
#include <iostream>
#include <string>
#include <list>
#include <utility>
```

Include dependency graph for fir.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::FIR

  FIR - Facial Identification Record.
- class FRsdk::FIRBuilder

  Building FIRs from serialized representations Use Enrollment::Processor to build FIRs from primary biometric data (face images).
- class FRsdk::Population

  An ordered (in the order of additions by add()) set of named FIR's which represents the population used for identifications.

Namespaces

- FRsdk

  The global name space for the SDK.

Constant Groups

- FRsdk

    The global name space for the SDK.

Typedefs

- typedef std::list< Score > FRsdk::Scores

    an ordered collection of score values
- typedef std::pair< std::string,
  Score > FRsdk::Match

    a named score for a match of two FIR's
- typedef std::list< Match > FRsdk::Matches

    the container used for a set of Matches.

Functions

- std::ostream & FRsdk::operator<< (std::ostream &o, const FIR &fir)

    output operator for FIR's

0.37.19.1 Detailed Description

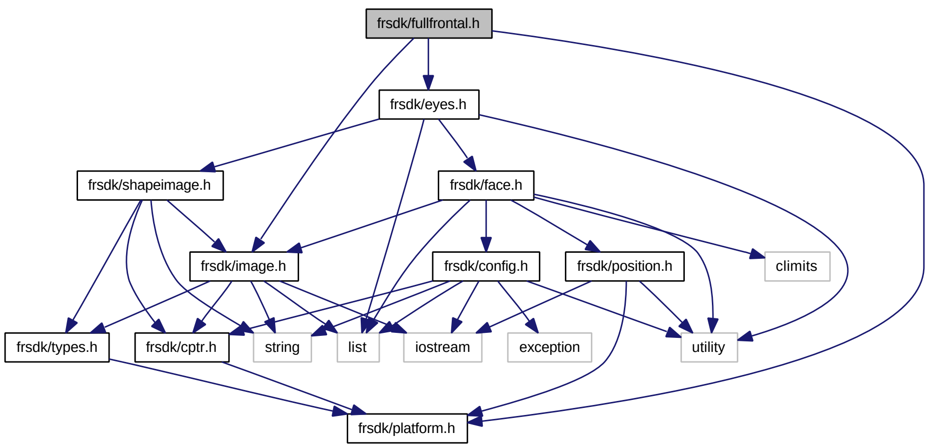
FIR (facial identification record) abstraction.

0.37.20 frsdk/fullfrontal.h File Reference

ISO/IEC 19794-5 Full Frontal Image extraction.

```
#include <frsdk/platform.h>
#include <frsdk/image.h>
#include <frsdk/eyes.h>
```

Include dependency graph for fullfrontal.h:



Classes

- class FRsdk::ISO_19794_5::FullFrontal::Creator

    Extract a Full Frontal Image from the source image.
- class FRsdk::ISO_19794_5::FullFrontal::Creator::PaddingRatioExceeded

Exception of this type is thrown when the padding ratio of the to be extracted FullFrontal portrait exceeds the pre-configured threshold.

Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::ISO_19794_5

  Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::FullFrontal

  Support for Full Frontal type as defined in ISO_19794_5 section 8.

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::ISO_19794_5

  Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::FullFrontal

  Support for Full Frontal type as defined in ISO_19794_5 section 8.

0.37.20.1 Detailed Description

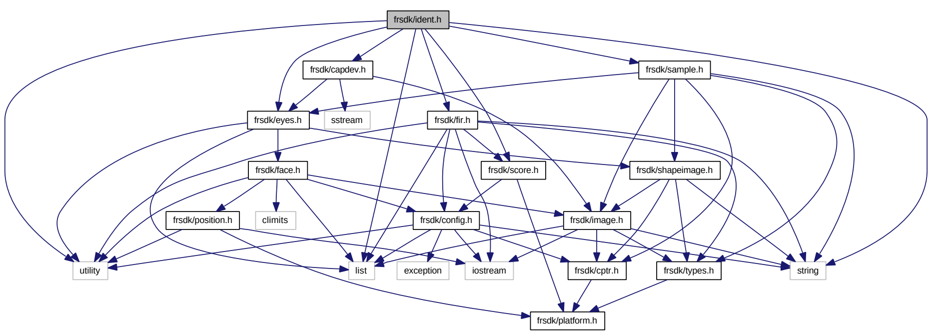
ISO/IEC 19794-5 Full Frontal Image extraction.

0.37.21 frsdk/ident.h File Reference

support the identification use case

```
#include <frsdk/capdev.h>
#include <frsdk/fir.h>
#include <frsdk/score.h>
#include <frsdk/eyes.h>
#include <frsdk/sample.h>
#include <list>
#include <string>
#include <utility>
```

Include dependency graph for ident.h:



Classes

- class FRsdk::Identification::FeedbackBody

  Body class for Feedback.
- class FRsdk::Identification::Feedback

  explicit template instantiation for win32
- class FRsdk::Identification::Processor

  this class represents the interface to the identification process.

Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::Identification

  the namespace for the identification facility

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::Identification

  the namespace for the identification facility

0.37.21.1 Detailed Description

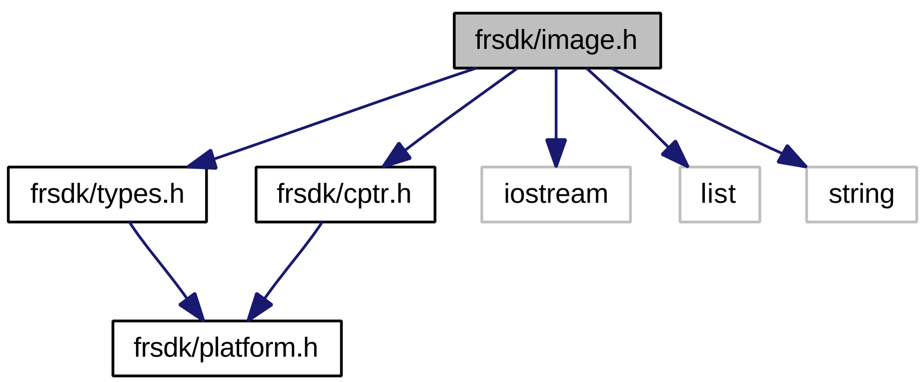
support the identification use case

0.37.22 frsdk/image.h File Reference

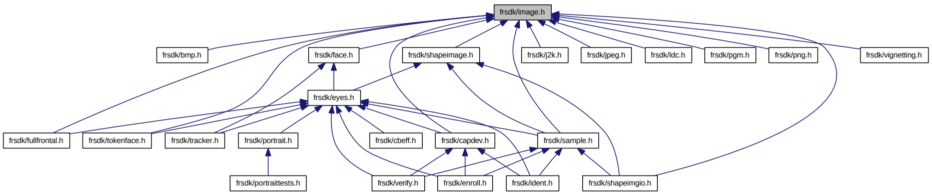
the image abstraction

```
#include <frsdk/types.h>
#include <frsdk/cptr.h>
#include <iostream>
#include <list>
#include <string>
```

Include dependency graph for image.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::ImageBody

  Abstract image body.
- class FRsdk::Image

  explicit template instantiation for win32
- class FRsdk::ImageIO::PropertiesFeedbackBody

  abstract PropertiesFeedback body
- class FRsdk::ImageIO::PropertiesFeedback

  explicit template instantiation for win32

Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::ImageIO

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::ImageIO

Typedefs

- typedef std::list< Image > FRsdk::ImageSet

  container used for collection of images within FaceVACS-SDK

Enumerations

- enum FRsdk::ImageIO::ColorMode {
  FRsdk::ImageIO::Unknown, FRsdk::ImageIO::RGB, FRsdk::ImageIO::RGBA, FRsdk::ImageIO::Alpha, FRsdk::ImageIO::Intensity, FRsdk::ImageIO::Palette, FRsdk::ImageIO::YCbCr, FRsdk::ImageIO::YUY2, FRsdk::ImageIO::YCCK, FRsdk::ImageIO::CMYK, FRsdk::ImageIO::YVYU, FRsdk::ImageIO::UYVY, FRsdk::ImageIO::RLE8, FRsdk::ImageIO::RLE4, FRsdk::ImageIO::BITFIELDS, FRsdk::ImageIO::YVU9, FRsdk::ImageIO::YV12, FRsdk::ImageIO::I420, FRsdk::ImageIO::IYUV, FRsdk::ImageIO::Y800, FRsdk::ImageIO::Y8, FRsdk::ImageIO::CIELAB, FRsdk::ImageIO::ICCLAB, FRsdk::ImageIO::ITULAB, FRsdk::ImageIO::LOGL, FRsdk::ImageIO::LOGLUV, FRsdk::ImageIO::MASK, FRsdk::ImageIO::SEPARATED }
- enum FRsdk::ImageRotation { FRsdk::ROTATECLOCKWISE, FRsdk::ROTATECOUNTERCLOCKWISE, FRsdk::ROTATEUPSIDEDOWN }

Functions

- Image FRsdk::ImageIO::load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from the given file.
- Image FRsdk::ImageIO::load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from the given stream.
- FRsdk::Image FRsdk::rotateImage (const FRsdk::Image &, ImageRotation)

0.37.22.1 Detailed Description

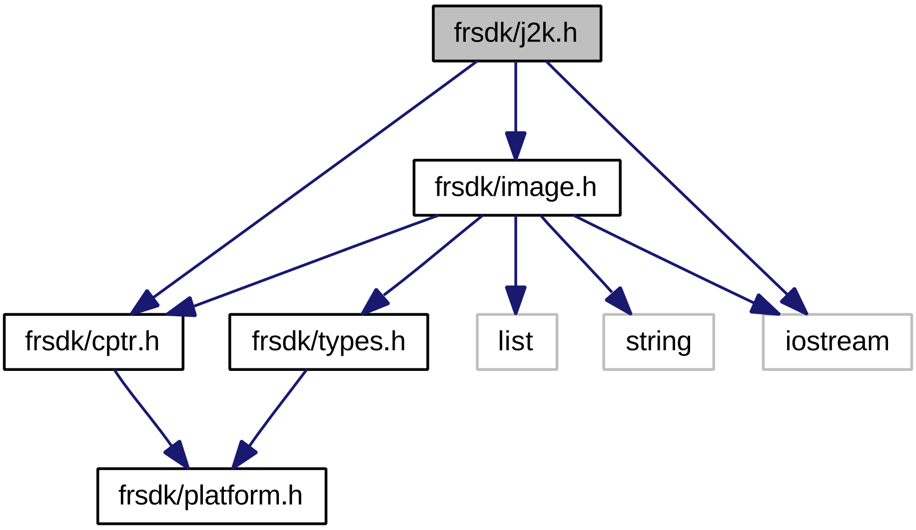
the image abstraction

0.37.23 frsdk/j2k.h File Reference

JPEG 2000 image format support.

```
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <iostream>
```

Include dependency graph for j2k.h:



Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::Jpeg2000

  JPEG 2000 Image support.

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::Jpeg2000

  JPEG 2000 Image support.

Functions

- Image FRsdk::Jpeg2000::load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    Loads an image representation from the given file which must be in jpeg 2000 format.
- Image FRsdk::Jpeg2000::load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    Loads an image representation from the given stream which is expected to contain jpeg 2000 format.
- Image FRsdk::Jpeg2000::load (const char ∗buf, unsigned int size, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

    Loads an image representation from memory which is expected to contain jpeg 2000 format.

0.37.23.1 Detailed Description

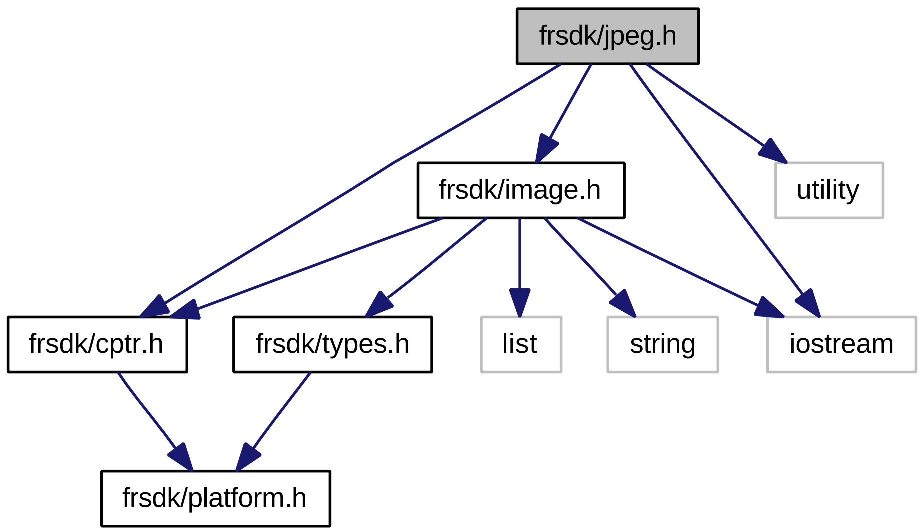
JPEG 2000 image format support.

0.37.24 frsdk/jpeg.h File Reference

JPEG (Joint Photographic Experts Group) image format support.

```
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <iostream>
#include <utility>
```

Include dependency graph for jpeg.h:



Classes

- struct FRsdk::Jpeg::Properties

    contains properties of a JPEG image

Namespaces

- FRsdk

    The global name space for the SDK.

- FRsdk::Jpeg

  JPEG Image support.

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::Jpeg

  JPEG Image support.

Functions

- Image FRsdk::Jpeg::load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::-PropertiesFeedback())

  Constructs an image representation from the given file which must be in jpeg format.
- Image FRsdk::Jpeg::load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::Properties-Feedback())

  Constructs an image representation from the given stream which is in jpeg format.
- Image FRsdk::Jpeg::load (const char ∗buf, unsigned int size, const ImageIO::PropertiesFeedback &fb=ImageIO::PropertiesFeedback())

  Constructs an image representation from memory which is in jpeg format.
- int FRsdk::Jpeg::save (const Image &img, std::ostream &, int quality=100)

  saves the image to the ostream using given jpeg quality (1 .
- int FRsdk::Jpeg::save (const Image &img, const std::string &filename, int quality=100)

  saves the image to a file given by name using given jpeg quality (1 .
- Properties FRsdk::Jpeg::saveWithSizeConstraint (const Image &img, std::ostream &, int maxSize)

  saves the image to the ostream which has a size constraint given by maxSize (Byte).

0.37.24.1 Detailed Description

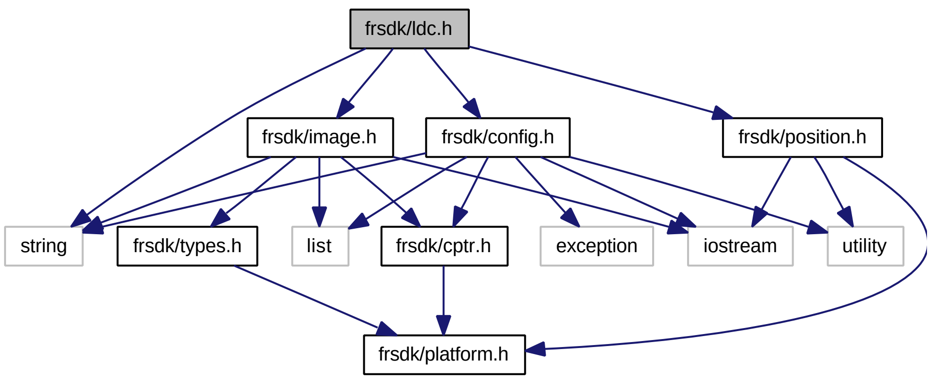
JPEG (Joint Photographic Experts Group) image format support.

0.37.25 frsdk/ldc.h File Reference

Radial Lense Correction.

```
#include <frsdk/image.h>
#include <frsdk/position.h>
#include <frsdk/config.h>
#include <string>
```

Include dependency graph for ldc.h:



Classes

- class FRsdk::LenseDistortionCorrector

  A class providing radial lense distortion correction.

Namespaces

- FRsdk

  The global name space for the SDK.

Constant Groups

- FRsdk

  The global name space for the SDK.

0.37.25.1 Detailed Description

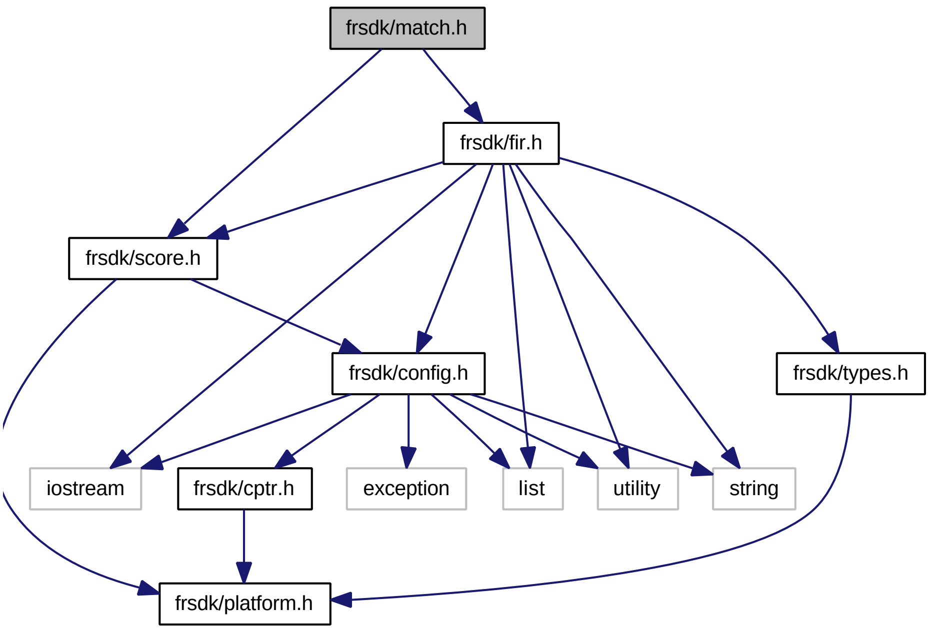
Radial Lense Correction.

0.37.26 frsdk/match.h File Reference

Low level match facilities.

```
#include <frsdk/fir.h>
#include <frsdk/score.h>
```

Include dependency graph for match.h:



Classes

- class FRsdk::FacialMatchingEngine

  Low level facial comparison facility.

Namespaces

- FRsdk

    The global name space for the SDK.

Constant Groups

- FRsdk

    The global name space for the SDK.

0.37.26.1 Detailed Description

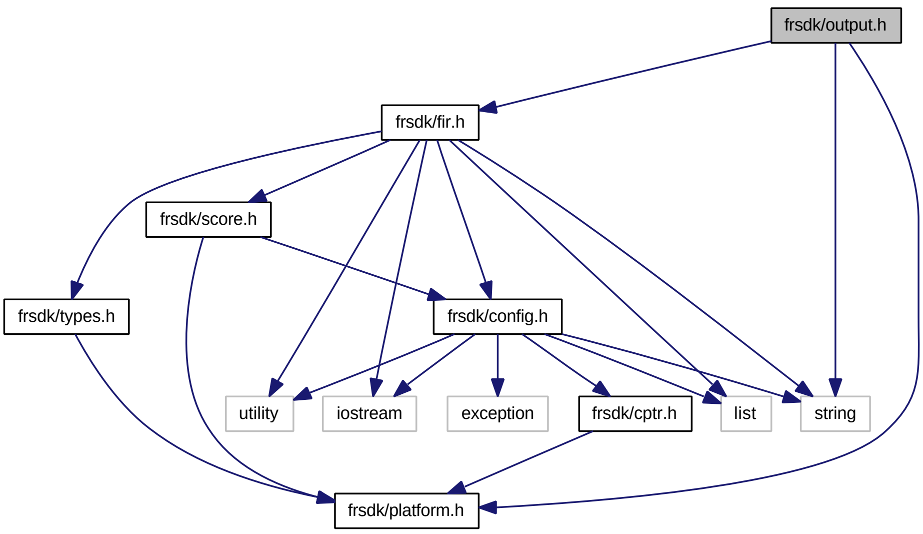
Low level match facilities. The file contains interfaces to calculate low level comparisions between FIR's and FIR's or Populations.

0.37.27 frsdk/output.h File Reference

file input output for matchlist and score files

```
#include <string>
#include <frsdk/platform.h>
#include <frsdk/fir.h>
```

Include dependency graph for output.h:



Namespaces

- FRsdk

    The global name space for the SDK.

- FRsdk::Tools

    Misc tools.

Constant Groups

- FRsdk

    The global name space for the SDK.

- FRsdk::Tools

  Misc tools.

Functions

- void setMagic (const std::string &magic)
- const char * getMagic ()

Score file i/o functions

- bool FRsdk::Tools::storeScoreValueFile (const std::string &probeRecordId, unsigned int positionInProbe, const FRsdk::Scores &scores, const std::string &destinationDirectory)

  store the scores into a file in destinationDirectory
- bool FRsdk::Tools::loadScoreValueFile (const std::string &probeRecordId, unsigned int positionInProbe, FRsdk::Scores &scores, const std::string &sourceDirectory)

  load the scores for a person ID file from sourceDirectory
- bool FRsdk::Tools::loadScoreValueFile (const std::string &filename, FRsdk::Scores &scores)

  load the scores from a file

Match list file i/o functions

- bool FRsdk::Tools::storeMatchListFile (const std::string &probeRecordId, unsigned int positionInProbe, const FRsdk::Matches &matches, const std::string &destinationDirectory)

  store the matches into a file in destinationDirectory
- bool FRsdk::Tools::loadMatchListFile (const std::string &probeRecordId, unsigned int positionInProbe, F-Rsdk::Matches &matches, const std::string &sourceDirectory)

  load the matches for a person ID file from sourceDirectory
- bool FRsdk::Tools::loadMatchListFile (const std::string &filename, FRsdk::Matches &matches)

  load the matches from a file

0.37.27.1 Detailed Description

file input output for matchlist and score files The file contains interfaces to store and load match lists and score matrices.

0.37.27.2 Function Documentation

0.37.27.2.1 const char* getMagic ( )

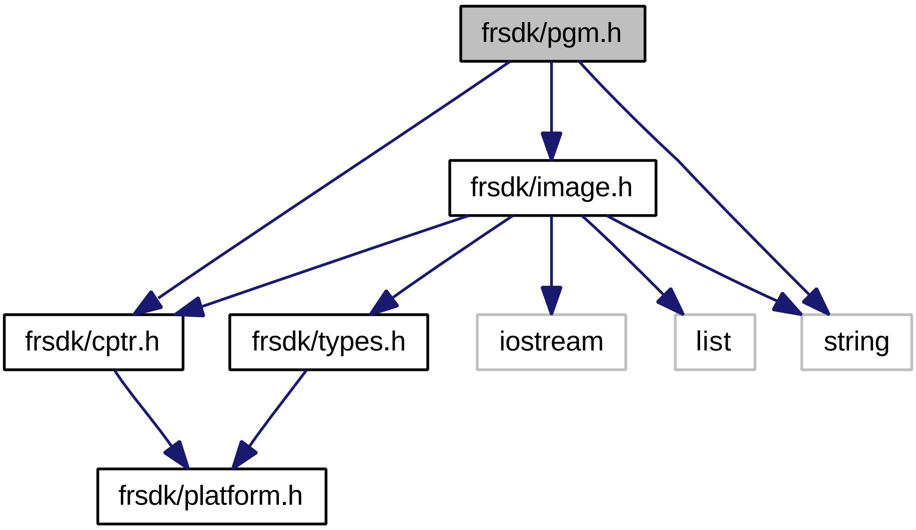
0.37.27.2.2 void setMagic (const std::string & *magic*)

0.37.28 frsdk/pgm.h File Reference

PGM (portable gray map) image format support.

```
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <string>
```

Include dependency graph for pgm.h:



Namespaces

- FRsdk

    The global name space for the SDK.

- FRsdk::Pgm

    PGM/PPM image format support.

Constant Groups

- FRsdk

    The global name space for the SDK.

- FRsdk::Pgm

    PGM/PPM image format support.

Functions

- Image FRsdk::Pgm::load (const std::string &filename, const ImageIO::PropertiesFeedback &fb=ImageIO::-PropertiesFeedback())

    constructs an image representation from the specified pgm format file

- void FRsdk::Pgm::save (const Image &img, const std::string &filename)

    Saves the image to filename; gray scale images will be stored in pgm-format, color images in ppm-format.

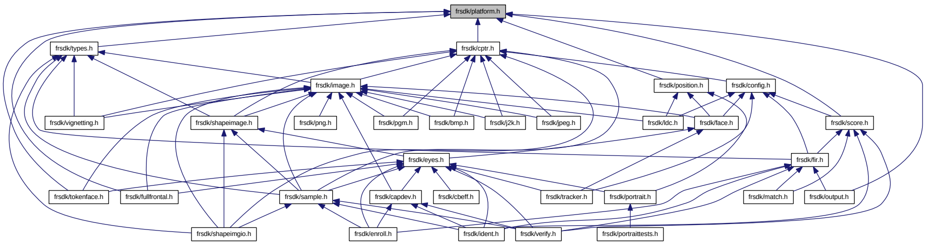
0.37.28.1 Detailed Description

PGM (portable gray map) image format support.

0.37.29 frsdk/platform.h File Reference

some platform supporting stuff

This graph shows which files directly or indirectly include this file:



Macros

- #define FRSDK_SHARED __declspec(dllimport)
- #define FRSDK_EXTERN extern
- #define FRSDK_EXPLICIT(arg) FRSDK_EXTERN template class FRSDK_SHARED arg

  explicit template instantiation for win32 for disable warnings
- #define FRSDK_EXPLICIT_STRUCT2(arg1, arg2) FRSDK_EXTERN template struct FRSDK_SHARED arg1,arg2

  explicit template instantiation for win32 for disable warnings

0.37.29.1 Detailed Description

some platform supporting stuff

0.37.29.2 Macro Definition Documentation

0.37.29.2.1 #define FRSDK_EXPLICIT(*arg*) FRSDK_EXTERN template class FRSDK_SHARED arg

explicit template instantiation for win32 for disable warnings

0.37.29.2.2 #define FRSDK_EXPLICIT_STRUCT2(*arg1*, *arg2*) FRSDK_EXTERN template struct FRSDK_SHARED arg1,arg2

explicit template instantiation for win32 for disable warnings

0.37.29.2.3 #define FRSDK_EXTERN extern

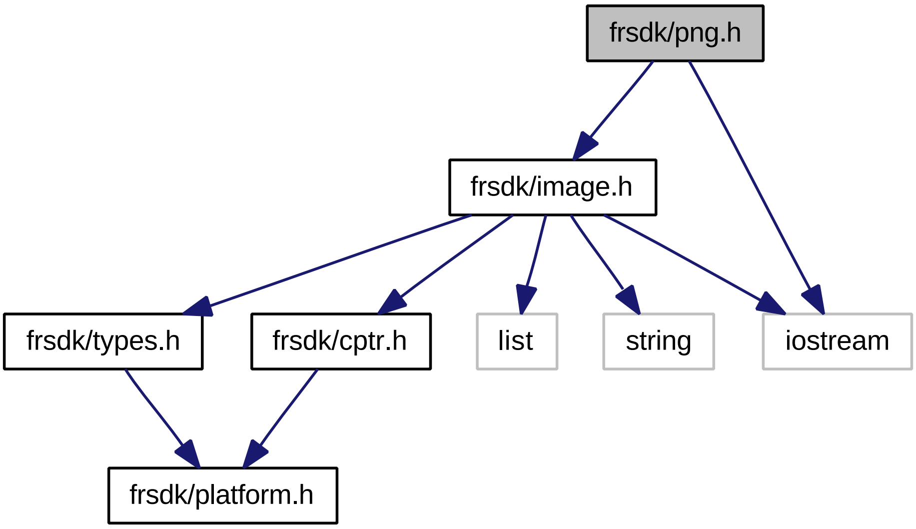
0.37.29.2.4 #define FRSDK_SHARED __declspec(dllimport)

0.37.30 frsdk/png.h File Reference

PNG (Portable Network Graphics) image format support.

```
#include <frsdk/image.h>
#include <iostream>
```

Include dependency graph for png.h:



Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::Png

  PNG (Portable Network Graphics) image format support.

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::Png

  PNG (Portable Network Graphics) image format support.

Functions

- Image FRsdk::Png::load (const std::string &fileName, const ImageIO::PropertiesFeedback &fb=ImageIO::-PropertiesFeedback())

  Constructs an image representation from image file specified by name.
- Image FRsdk::Png::load (std::istream &is, const ImageIO::PropertiesFeedback &fb=ImageIO::Properties-Feedback())

  constructs an image representation from the specified PNG formated stream
- void FRsdk::Png::save (const Image &img, std::ostream &os, int compressionLevel=6)

  Write image as PNG image to the given stream, the stream will not be closed.

0.37.30.1 Detailed Description

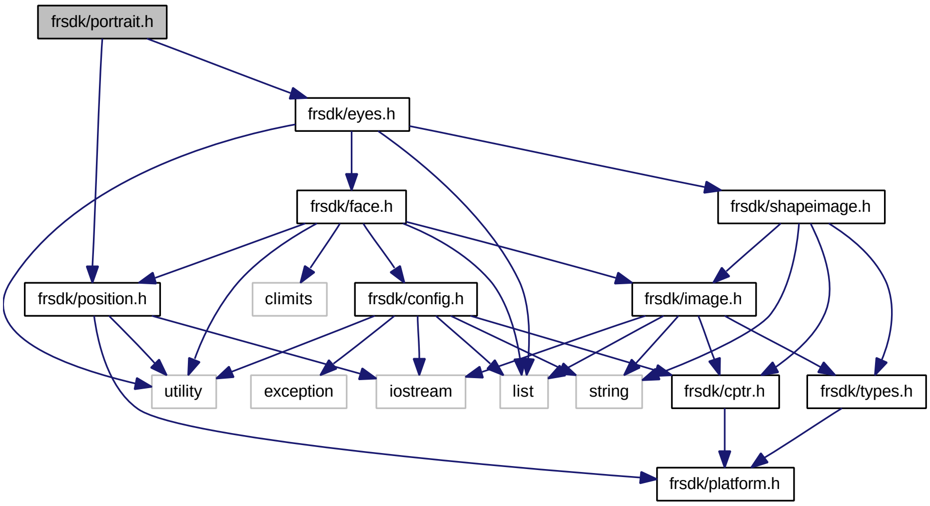
PNG (Portable Network Graphics) image format support.

0.37.31 frsdk/portrait.h File Reference

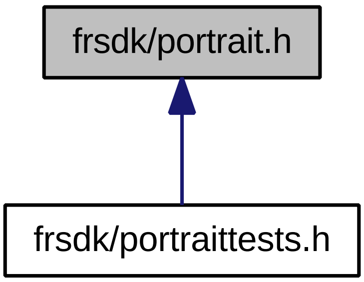
Portrait characteristics analysis support.

```
#include <frsdk/position.h>
#include <frsdk/eyes.h>
```

Include dependency graph for portrait.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FRsdk::Portrait::EthnicityMeasurements

  measurements for ethnicity detection, contains the probability that a person belongs to the ethnicity class
- class FRsdk::Portrait::Characteristics

  Portrait Characteristics.
- class FRsdk::Portrait::Analyzer

  Portrait Characteristics Analyzer, create Portrait characteristics from annotated images.

Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::Portrait

  Portrait characteristics and feature tests.

Constant Groups

- FRsdk

    The global name space for the SDK.
- FRsdk::Portrait

    Portrait characteristics and feature tests.

Functions

- Box FRsdk::Portrait::earToEarChinCrownSurroundingBox (const Characteristics &)

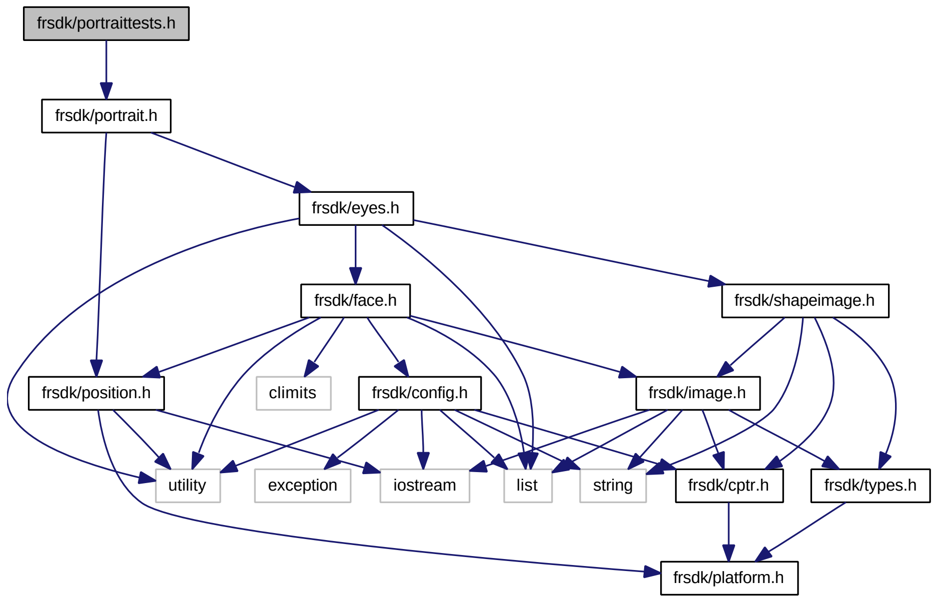
    This function returns the smallest surrounding box of the face according to the chin, crown and ear positions estimated in portrait characteristics.

0.37.31.1 Detailed Description

Portrait characteristics analysis support.

0.37.32 frsdk/portraittests.h File Reference

`#include <frsdk/portrait.h>`
Include dependency graph for portraittests.h:



Classes

- class FRsdk::ISO_19794_5::FullFrontal::Compliance

    Compliance assessment results.
- class FRsdk::ISO_19794_5::FullFrontal::Boundaries

    Boundaries are used from the FullFrontal::Test.
- class FRsdk::ISO_19794_5::FullFrontal::Test

    Compliance assessment.
- class FRsdk::Portrait::Feature::Set

    Feature assessment results.
- class FRsdk::Portrait::Feature::Test

    Test for features in a portrait.

Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::ISO_19794_5

  Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::FullFrontal

  Support for Full Frontal type as defined in ISO_19794_5 section 8.
- FRsdk::Portrait

  Portrait characteristics and feature tests.
- FRsdk::Portrait::Feature

  Test for features in the portrait.

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::ISO_19794_5

  Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::FullFrontal

  Support for Full Frontal type as defined in ISO_19794_5 section 8.
- FRsdk::Portrait

  Portrait characteristics and feature tests.
- FRsdk::Portrait::Feature

  Test for features in the portrait.

Enumerations

- enum FRsdk::Portrait::Feature::Gender { FRsdk::Portrait::Feature::male, FRsdk::Portrait::Feature::female }

  Gender.
- enum FRsdk::Portrait::Feature::Ethnicity { FRsdk::Portrait::Feature::white, FRsdk::Portrait::Feature::black, F-Rsdk::Portrait::Feature::asian }

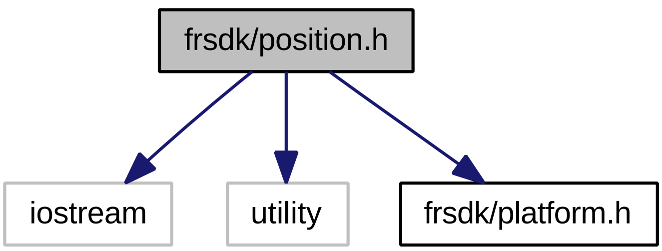
  Ethnicity.

0.37.33 frsdk/position.h File Reference

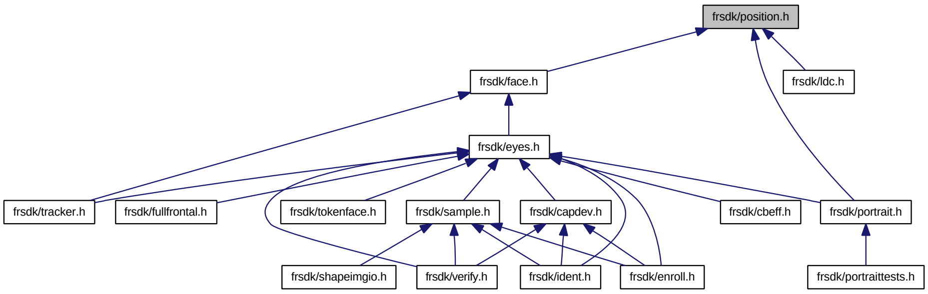
continuous two-dimensional coordinates abstraction

```
#include <iostream>
#include <utility>
#include <frsdk/platform.h>
```

Include dependency graph for position.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::Position

    explicit template instantiation for win32

- class FRsdk::Box

    The class Box describes a rectangular box in discrete 2D coordinates defined by 2 opposite points, origin and end, where origin.x <= end.x and origin.y <= end.y.

Namespaces

- FRsdk

    The global name space for the SDK.

Constant Groups

- FRsdk

    The global name space for the SDK.

0.37.33.1 Detailed Description

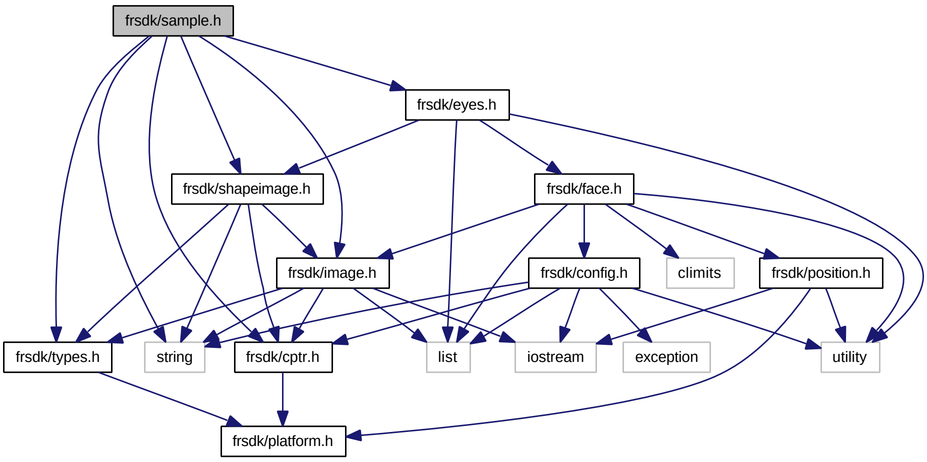
continuous two-dimensional coordinates abstraction

0.37.34 frsdk/sample.h File Reference

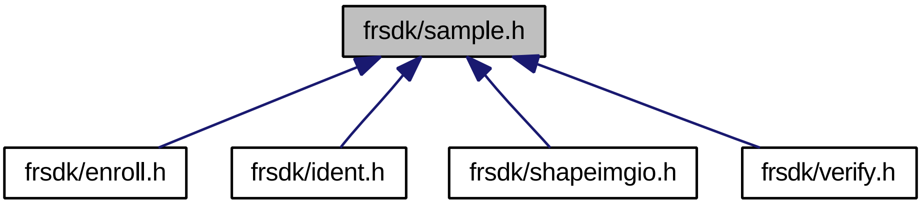
the image abstraction

```
#include <frsdk/types.h>
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <frsdk/shapeimage.h>
#include <frsdk/eyes.h>
#include <string>
```

Include dependency graph for sample.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::Sample

    Data sample containing mandatory intensity image and optional shape data and eye positions.
- struct FRsdk::Sample::Vector3D
- class FRsdk::Sample::ItemNotSet

    Specific exception type indicating access to optional data not present in Sample.

Namespaces

- FRsdk

    The global name space for the SDK.

Constant Groups

- FRsdk

    The global name space for the SDK.

Typedefs

- typedef std::list< Sample > FRsdk::SampleSet

    container used for collection of Samples within FaceVACS-SDK

0.37.34.1 Detailed Description

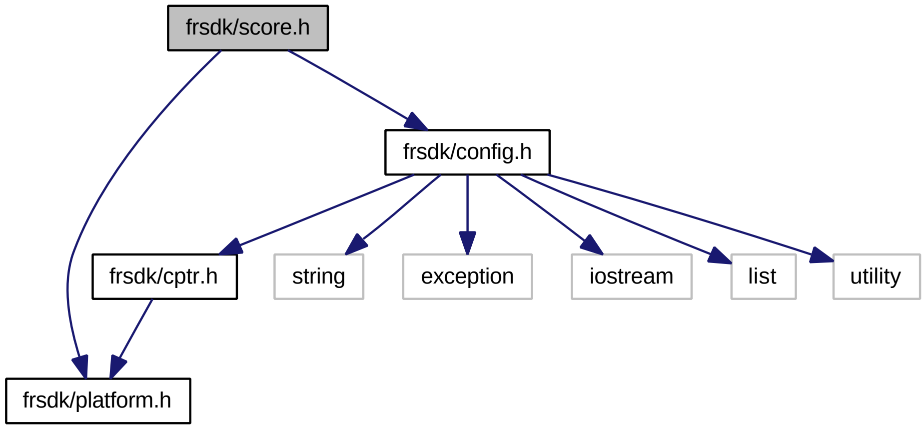
the image abstraction

0.37.35 frsdk/score.h File Reference

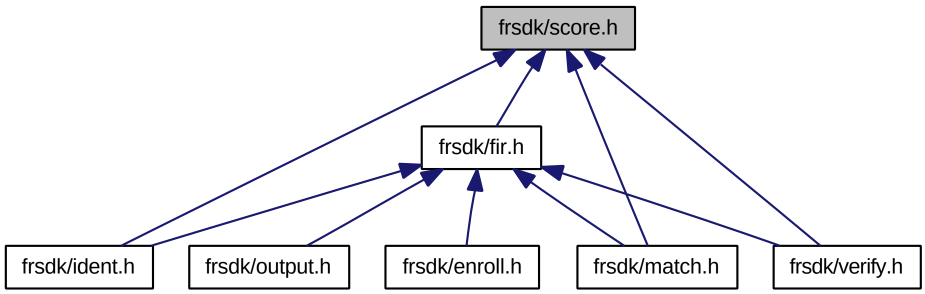
score abstraction

```
#include <frsdk/platform.h>
#include <frsdk/config.h>
```

Include dependency graph for score.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FRsdk::Score

  This class represents a score for representing the comparison result between a FIR and the biometric evidence.
- class FRsdk::ScoreMappings

  FAR,FRR / Score mappings.

Namespaces

- FRsdk

  The global name space for the SDK.

Constant Groups

- FRsdk

    The global name space for the SDK.

0.37.35.1 Detailed Description

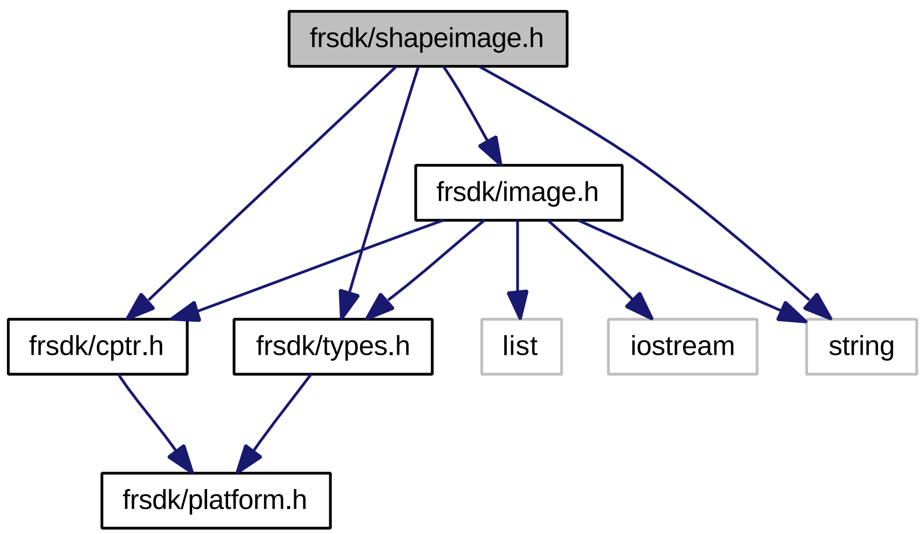
score abstraction

0.37.36 frsdk/shapeimage.h File Reference

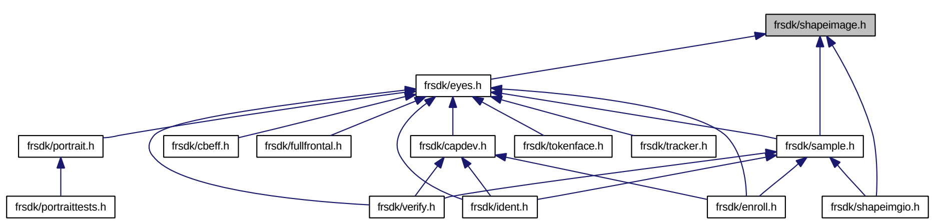
the image abstraction

```
#include <frsdk/types.h>
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <string>
```

Include dependency graph for shapeimage.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FRsdk::Vertex
- class FRsdk::ShapeImageBody

    Abstract shape image body.

- class FRsdk::ShapeImage

    explicit template instantiation for win32

- class FRsdk::PointSetBody
- class FRsdk::PointSet

Namespaces

- FRsdk
  The global name space for the SDK.

Constant Groups

- FRsdk
  The global name space for the SDK.

Functions

- CountedPtr< ShapeImageBody > FRsdk::createImageBody (const Vertex ∗, const bool ∗mask, unsigned int width, unsigned int height, bool takeOwnerShip)
  create a ShapeImageBody from vertex and mask data.

0.37.36.1 Detailed Description

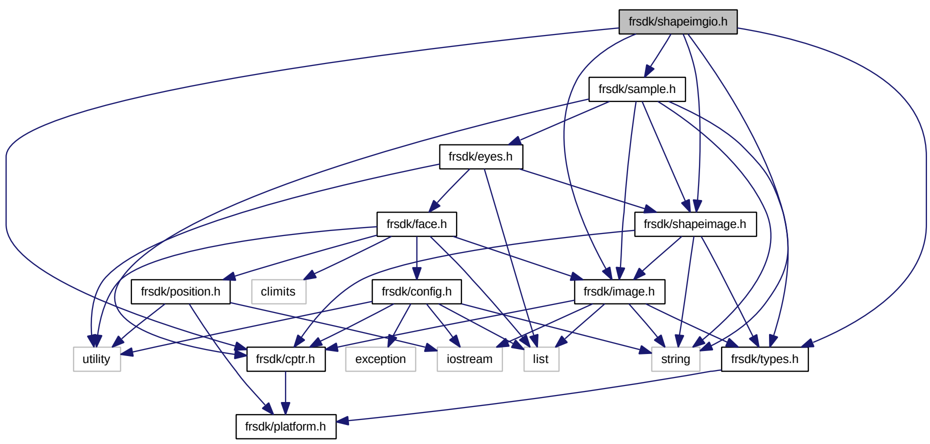
the image abstraction

0.37.37 frsdk/shapeimgio.h File Reference

the image abstraction

```
#include <frsdk/types.h>
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <frsdk/shapeimage.h>
#include <frsdk/sample.h>
#include <string>
```

Include dependency graph for shapeimgio.h:



Namespaces

- FRsdk
  The global name space for the SDK.

Constant Groups

- FRsdk

  The global name space for the SDK.

Functions

- ShapeImage FRsdk::loadShapeImage (const std::string &)

  Constructs a ShapeImage from the given file which must be in one of the supported formats.
- ShapeImage FRsdk::loadShapeImage (std::istream &is)

  Builds a ShapeImage from the given stream which must provide one of the supported formats.
- Sample FRsdk::loadShapeSample (const std::string &)

  Constructs a Shape sample from the given file which must be in one of the supported formats.

0.37.37.1 Detailed Description

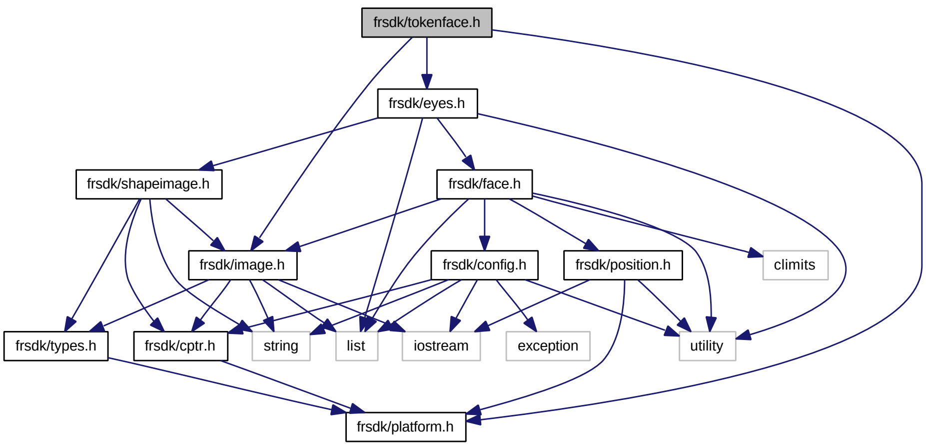
the image abstraction

0.37.38 frsdk/tokenface.h File Reference

ISO/IEC 19794-5 Token Face Image extraction.

```
#include <frsdk/platform.h>
#include <frsdk/image.h>
#include <frsdk/eyes.h>
```

Include dependency graph for tokenface.h:



Classes

- class FRsdk::ISO_19794_5::TokenFace::Creator

  Extract a Token Face Image from the source image.
- class FRsdk::ISO_19794_5::TokenFace::Creator::PaddingRatioExceeded

  Exception of this type is thrown when the padding ratio of the to be extracted TokenFace exceeds the pre-configured threshold.

Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::ISO_19794_5

  Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::TokenFace

  Support for Token Face Image type as defined in ISO_19794_5 9.2.

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::ISO_19794_5

  Support for image formats defined by ISO/IEC 19794-5:2005.
- FRsdk::ISO_19794_5::TokenFace

  Support for Token Face Image type as defined in ISO_19794_5 9.2.

0.37.38.1 Detailed Description

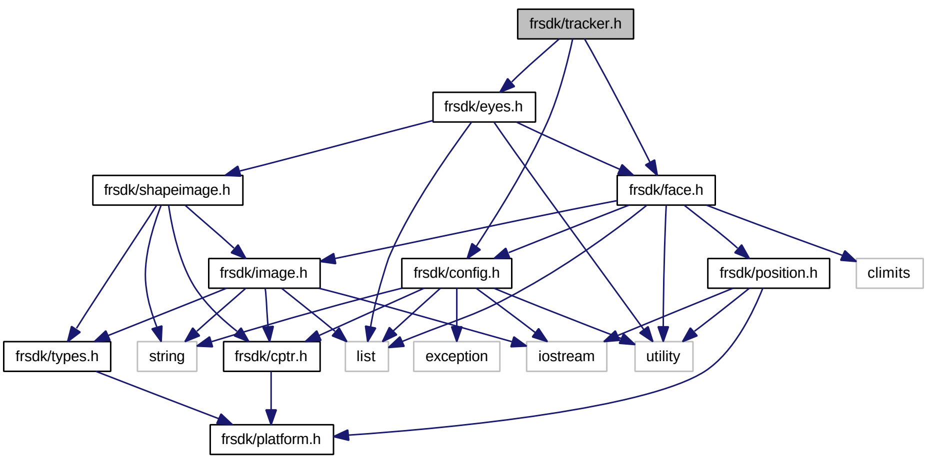
ISO/IEC 19794-5 Token Face Image extraction.

0.37.39 frsdk/tracker.h File Reference

Face Tracking Engine.

```
#include <frsdk/face.h>
#include <frsdk/eyes.h>
#include <frsdk/config.h>
```

Include dependency graph for tracker.h:



Classes

- class FRsdk::Face::Tracker

  The Face Tracker locates and tracks faces across a sequence of images in an efficient way by analyzing the spatial and temporal dependencies between faces in subsequent images.
- struct FRsdk::Face::Tracker::Location

  The location of a face being tracked by the face tracker.

Namespaces

- FRsdk

  The global name space for the SDK.
- FRsdk::Face

  the name space for the face finding facility

Constant Groups

- FRsdk

  The global name space for the SDK.
- FRsdk::Face

  the name space for the face finding facility

0.37.39.1 Detailed Description

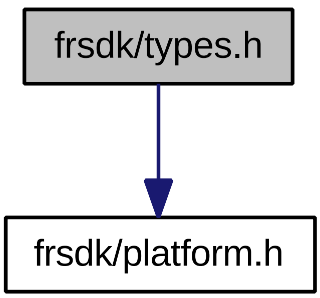
Face Tracking Engine.

0.37.40 frsdk/types.h File Reference

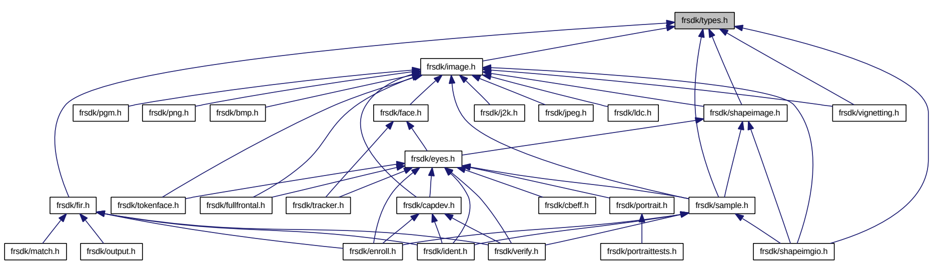
some type definitions

`#include <frsdk/platform.h>`

Include dependency graph for types.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct FRsdk::Rgb

  Red, green and blue color model.

Namespaces

- FRsdk

  The global name space for the SDK.

Constant Groups

- FRsdk

  The global name space for the SDK.

Typedefs

- typedef unsigned char FRsdk::Byte

  A 8 bit byte representation.

0.37.40.1 Detailed Description

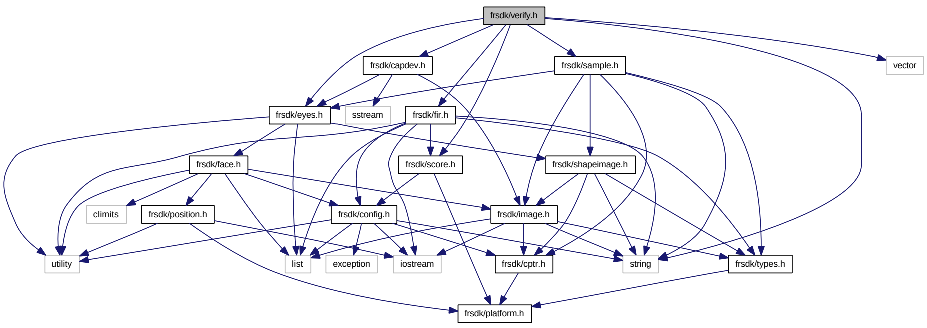
some type definitions

0.37.41 frsdk/verify.h File Reference

support for the verification use case

```
#include <frsdk/capdev.h>
#include <frsdk/fir.h>
#include <frsdk/eyes.h>
#include <frsdk/score.h>
#include <frsdk/sample.h>
#include <string>
#include <vector>
```

Include dependency graph for verify.h:



Classes

- class FRsdk::Verification::FeedbackBody

  Body class for Feedback.
- class FRsdk::Verification::Feedback

  explicit template instantiation for win32
- class FRsdk::Verification::Processor

  this class represents the interface to the verification process Calls of process() are serialized but using multiple Processors verifications can be done in parallel (one processor per thread).

Namespaces

- FRsdk

    The global name space for the SDK.
- FRsdk::Verification

    the namespace for the verification facility

Constant Groups

- FRsdk

    The global name space for the SDK.
- FRsdk::Verification

    the namespace for the verification facility

0.37.41.1 Detailed Description

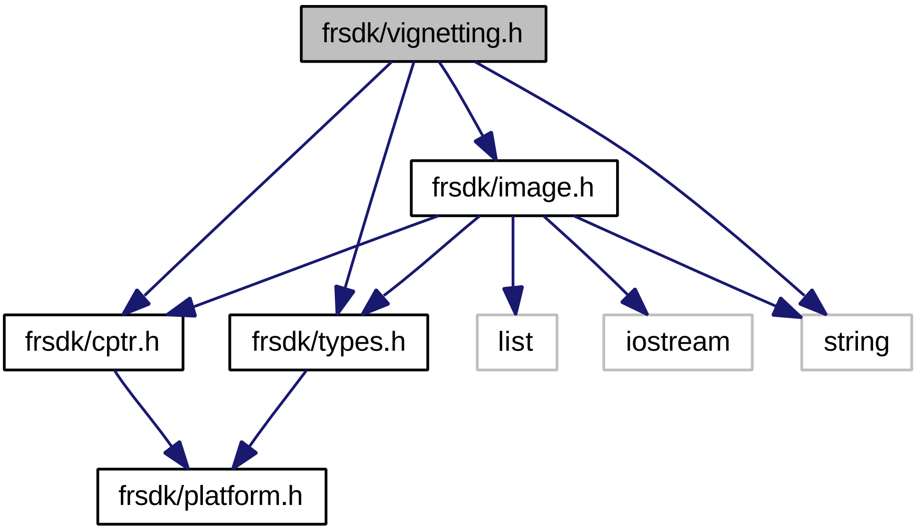
support for the verification use case

0.37.42 frsdk/vignetting.h File Reference

vignetting of images

```
#include <frsdk/types.h>
#include <frsdk/cptr.h>
#include <frsdk/image.h>
#include <string>
```

Include dependency graph for vignetting.h:



Namespaces

- FRsdk

    The global name space for the SDK.

Constant Groups

- FRsdk

  The global name space for the SDK.

Enumerations

- enum FRsdk::SmoothingFunction { FRsdk::Fixed, FRsdk::Linear, FRsdk::Gaussian }

  There are threee ways to blend the margin to the target color: Gaussian blends intensisty of each color channel with a half gaussian function, Linear blend is propotional to the border distance and Fixed sets all pixel of margin to the border color.

Functions

- FRsdk::Image FRsdk::vignetting (const FRsdk::Image &, const FRsdk::Rgb &, int margin=10, int radius=0, SmoothingFunction sf=Gaussian)

  In photographic areas vignetting is a intensity blending of round or eliptical margin regions.

0.37.42.1 Detailed Description

vignetting of images

0.38 Example Documentation

0.38.1 acquisition.cc

The following example demonstrates how to implement a simple image acquisition application. For a detailed explanation see Tutorial - Simple image acquisition application.

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.55 $
//


#include <frsdk/cbeff.h>
#include <frsdk/config.h>
#include <frsdk/eyes.h>
#include <frsdk/jpeg.h>
#include <frsdk/j2k.h>
#include <frsdk/bmp.h>
#include <frsdk/png.h>
#include <frsdk/pgm.h>
#include <frsdk/portrait.h>
#include <frsdk/portraittests.h>
#include <frsdk/tokenface.h>

#include "cmdline.h"

#include <cmath>
#include <cstdlib>

#include <sstream>
#include <fstream>
#include <exception>

using namespace std;
using namespace FRsdk;

ostream& operator<<( ostream& o, const FRsdk::Position& p) {
  o << " [ " << p.x() << " , " << p.y() << " ] ";
  return o;
}
```

```
#ifdef UNDER_CE
void printMemoryStatus( const std::string& where ) {
  using namespace std;
  MEMORYSTATUS      ms;
  STORE_INFORMATION si;
  TCHAR   szBuf[MAX_PATH];

  // Program memory.
  ms.dwLength = sizeof( ms);
  GlobalMemoryStatus(&ms);

  cout << "Memory status at:" << where << endl;
  cout
      << dec
      << "Total RAM: " << ms.dwTotalPhys / 1024 << "KB" << endl
      << "Free     : " << ms.dwAvailPhys / 1024 << "KB" << endl
      << "Used     : " << (ms.dwTotalPhys - ms.dwAvailPhys) / 1024 << "KB"
      << endl
      << "ProcVirt : " << (ms.dwTotalVirtual - ms.dwAvailVirtual) / 1024 << "KB"
      << endl;

  // Storage memory.
  GetStoreInformation(&si);
  cout <<  "Storage RAM: " << si.dwFreeSize / 1024<< "KB" << endl;
}
#endif

ostream& operator<<( ostream& o, const FRsdk::Portrait::Characteristics& pc){
  o << "Characteristics: "<< endl;
  o << "\t isColor(): " << (pc.isColor()?" true ":" false ") << endl;
  o << "\t width(): " << pc.width() << endl;
  o << "\t height(): " << pc.height() << endl;
  o << "\t numberOfFaces(): " << pc.numberOfFaces() << endl;
  o << "\t exposure(): " << pc.exposure() << endl;
  o << "\t grayScaleDensity(): " << pc.grayScaleDensity() << endl;
  o << "\t hotSpots(): " << float(pc.hotSpots()) << endl;
  o << "\t backgroundUniformity(): " << float( pc.backgroundUniformity()) << endl;
#ifndef UNDER_CE
  o << "\t deviationFromUniformLighting(): " << pc.deviationFromUniformLighting() << endl;
#endif
  o << "\t sharpness(): " << pc.sharpness() << endl;
  o << "\t eye0Open(): " << pc.eye0Open() << endl;
  o << "\t eye1Open(): " << pc.eye1Open() << endl;
  o << "\t eye0GazeFrontal(): " << pc.eye0GazeFrontal() << endl;
  o << "\t eye1GazeFrontal(): " << pc.eye1GazeFrontal() << endl;
  if( pc.isColor()) {
    o << "\t eye0Red(): " << pc.eye0Red() << endl;
    o << "\t eye1Red(): " << pc.eye1Red() << endl;
    o << "\t naturalSkinColour(): " << pc.naturalSkinColour() << endl;
  }
  o << "\t eye0Tinted(): " << pc.eye0Tinted() << endl;
  o << "\t eye1Tinted(): " << pc.eye1Tinted() << endl;
  o << "\t poseAngleRoll(): " << pc.poseAngleRoll() << endl;
  o << "\t deviationFromFrontalPose(): " << pc.deviationFromFrontalPose() << endl;
#ifdef UNDER_CE
  printMemoryStatus( "Characteristics: " );
#endif
  o << "\t isMale(): " << float(pc.isMale()) << endl;
  o << "\t age(): " << pc.age() << endl;
  o << "\t mouthClosed(): " << pc.mouthClosed() << endl;
  o << "\t glasses(): " << pc.glasses() << endl;
  o << "\t eye0(): " << pc.eye0() << endl;
  o << "\t eye1(): " << pc.eye1() << endl;
  o << "\t eyeDistance(): " << pc.eyeDistance() << endl;
  o << "\t faceCenter(): " << pc.faceCenter() << endl;
  o << "\t widthOfHead(): " << pc.widthOfHead() << endl;
  o << "\t lengthOfHead(): " << pc.lengthOfHead() << endl;
  o << "\t chin(): " << pc.chin() << endl;
  o << "\t crown(): " << pc.crown() << endl;
  o << "\t ear0(): " << pc.ear0() << endl;
  o << "\t ear1(): " << pc.ear1() << endl;
  FRsdk::Box facebox =  FRsdk::Portrait::earToEarChinCrownSurroundingBox( pc);
  o << "\t face surrounding box: (("
    << facebox.originx() << ", " << facebox.originy() << "), ("
    << facebox.endx() << ", " << facebox.endy() << "))" << endl;
#ifdef UNDER_CE
  printMemoryStatus( "Characteristics: " );
#endif
  return o;
}

const char* printBool( const bool& b) {
  if( b) return " Passed " ;
  return " Failed " ;
}

void
```

```
printComplianceResult( ostream& o,
                       const FRsdk::ISO_19794_5::FullFrontal::Compliance& ffc,
                       const FRsdk::Portrait::Characteristics& pc )
{
  o << "FullFrontal compliance: " << endl;
  o << "\t onlyOneFaceVisible(): " << printBool( ffc.onlyOneFaceVisible()) << endl;
  o << "\t goodVerticalFacePosition(): " << printBool( ffc.
      goodVerticalFacePosition())
    << endl;
  o << "\t horizontallyCenteredFace(): " << printBool( ffc.
      horizontallyCenteredFace())
    << endl;
  o << "\t widthOfHead(): " << printBool( ffc.widthOfHead()) << endl;
  o << "\t lengthOfHead(): " << printBool( ffc.lengthOfHead()) << endl;
  o << "\t widthOfHeadBestPractice(): "
    << printBool( ffc.widthOfHeadBestPractice()) << endl;
  o << "\t lengthOfHeadBestPractice(): "
    << printBool( ffc.lengthOfHeadBestPractice()) << endl;
  o << "\t resolution(): " << printBool( ffc.resolution()) << endl;
  o << "\t resolutionBestPractice(): " << printBool( ffc.resolutionBestPractice()) << endl;
  o << "\t imageWidthToHeightBestPractice(): "
    << printBool( ffc.imageWidthToHeightBestPractice()) << endl;
  o << "\t goodExposure(): " << printBool( ffc.goodExposure()) << endl;
  o << "\t goodGrayScaleProfile(): " << printBool( ffc.goodGrayScaleProfile()) << endl;
  if ( pc.isColor() ) {
    o << "\t hasNaturalSkinColour(): "
      << printBool( ffc.hasNaturalSkinColour()) << endl;
  }
  o << "\t noHotSpots(): " << printBool( ffc.noHotSpots()) << endl;
  o << "\t isBackgroundUniformBestPractice(): "
    << printBool( ffc.isBackgroundUniformBestPractice()) << endl;
  o << "\t isFrontal(): " << printBool( ffc.isFrontal()) << endl;
  o << "\t isFrontalBestPractice(): "
    << printBool( ffc.isFrontalBestPractice()) << endl;
#ifndef UNDER_CE
  o << "\t isLightingUniform(): " << printBool( ffc.isLightingUniform()) << endl;
#endif
  o << "\t eyesOpenBestPractice(): "
    << printBool( ffc.eyesOpenBestPractice()) << endl;
  o << "\t eyesGazeFrontalBestPractice(): "
    << printBool( ffc.eyesGazeFrontalBestPractice()) << endl;
  if ( pc.isColor() ) {
      o << "\t eyesNotRedBestPractice(): "
        << printBool( ffc.eyesNotRedBestPractice()) << endl;
  }
  o << "\t noTintedGlasses(): " << printBool( ffc.noTintedGlasses()) << endl;
  o << "\t isSharp(): " << printBool( ffc.isSharp()) << endl;
  o << "\t mouthClosedBestPractice(): "
    << printBool( ffc.mouthClosedBestPractice()) << endl;
  o << endl;
}


ostream& operator<<( ostream& o,
                     const FRsdk::Portrait::Feature::Gender& g){
  if( g == FRsdk::Portrait::Feature::female)
    o << "female";
  else
    o << "male";
  return o;
}

ostream&
operator<<( ostream& o,
            const FRsdk::Portrait::Feature::Ethnicity& e){
  if( e == FRsdk::Portrait::Feature::black)
    o << "black" ;
  else   if( e == FRsdk::Portrait::Feature::asian)
    o << "asian" ;
  else
    o << "white" ;
  return o;
}
ostream&
operator<<( ostream& o,
            const FRsdk::Portrait::Feature::Set& e){
 o << "Portrait Feature: " << endl;
 // o << "\t gender(): " << e.gender() << endl;
 o << "\t ethnicity(): " << e.ethnicity() << endl;
 o << "\t wearsGlasses(): " << (e.wearsGlasses()?" true" : " false ") << endl;
 return o;
}


void printBoundaries( ostream& o,
                      const FRsdk::ISO_19794_5::FullFrontal::Boundaries& ffb,
                      const FRsdk::Portrait::Characteristics& pc )
```

```
{
  o << "Fullfrontal boundaries: " << endl;
  o << "\t tintedGlassesThreshold(): " << ffb.tintedGlassesThreshold() << endl;
  o << "\t wearsGlassesThreshold(): " << ffb.wearsGlassesThreshold() << endl;
  o << "\t hotSpotsThreshold(): " << ffb.hotSpotsThreshold() << endl;
  o << "\t closedMouthThreshold(): " << ffb.closedMouthThreshold() << endl;
  o << "\t minimalSharpness(): " << ffb.minimalSharpness() << endl;
  o << "\t eyesNotRedThreshold(): " << ffb.eyesNotRedThreshold() << endl;
  o << "\t eyesGazeFrontalThreshold(): " << ffb.eyesGazeFrontalThreshold()
    << endl;
  o << "\t eyesOpenThreshold(): " << ffb.eyesOpenThreshold() << endl;
#ifndef UNDER_CE
  o << "\t maxDeviationFromLighting(): " << ffb.maxDeviationFromLighting()
    << endl;
#endif
  o << "\t poseMaxRotation(): " << ffb.poseMaxRotation() << endl;
  o << "\t maxFrontalPoseDeviation(): " << ffb.maxFrontalPoseDeviation()
    << endl;
  o << "\t minimalGrayScaleBounding(): " << ffb.minimalGrayScaleBounding()
    << endl;
  o << "\t highExposerThreshold(): " << ffb.highExposerThreshold() << endl;
  o << "\t lowExposerThreshold(): " << ffb.lowExposerThreshold() << endl;
  o << "\t lowerBoundVerticalPosition(): " << ffb.lowerBoundVerticalPosition()
    << " ( " << ffb.lowerBoundVerticalPosition() * pc.height() << " ) "
    << endl;
  o << "\t upperBoundVerticalPosition(): " << ffb.upperBoundVerticalPosition()
    << " ( " << ffb.upperBoundVerticalPosition() * pc.height() << " ) "
    << endl;
  o << "\t leftBoundCenteredFace(): " << ffb.leftBoundCenteredFace()
    << " ( " << ffb.leftBoundCenteredFace() * pc.width() << " ) "
    << endl;
  o << "\t rightBoundCenteredFace(): " << ffb.rightBoundCenteredFace()
    << " ( " << ffb.rightBoundCenteredFace() * pc.width() << " ) "
    << endl;
  float eardist = pc.ear0() + pc.ear1();
  o << "\t minHeadWidth(): " << ffb.minHeadWidth()
    << " ( " <<  eardist << " ) " << endl;
  o << "\t minWidthOfHeadRatio(): " << ffb.minWidthOfHeadRatio()
    << " ( " << pc.width()/pc.widthOfHead()  << " ) "<< endl;
  o << "\t maxWidthOfHeadRatio(): " << ffb.maxWidthOfHeadRatio() << endl;
  o << "\t minLengthOfHeadRatio(): " << ffb.minLengthOfHeadRatio()
    << " ( " << pc.height()/pc.lengthOfHead()  << " ) "<< endl;
  o << "\t maxLengthOfHeadRatio(): " << ffb.maxLengthOfHeadRatio() << endl;
}


namespace {
  class AcquisitionError: public std::exception
  {
  public:
    AcquisitionError( const std::string& msg_) throw(): msg( msg_) {}
    ~AcquisitionError() throw() { }
    const char* what() const throw() { return msg.c_str(); }
  private:
    std::string msg;
  };
}

class ImageCoutFeedback: public FRsdk::ImageIO::PropertiesFeedbackBody{
public:
  ImageCoutFeedback() {}
  void compressionMode(const ImageIO::ColorMode& cm) {
      switch( cm){
      case ImageIO::Unknown:
        cout << "Colormode: " << "Unknown" << endl;
        break;
      case ImageIO::RGB:
        cout << "Colormode: " << "RGB" << endl;
        break;
      case ImageIO::RGBA:
        cout << "Colormode: " << "RGBA" << endl;
        break;
      case ImageIO::Alpha:
        cout << "Colormode: " << "Alpha" << endl;
        break;
      case ImageIO::Intensity:
        cout << "Colormode: " << "Intensity (Grayscale)" << endl;
        break;
      case ImageIO::Palette:
        cout << "Colormode: " << "Palette" << endl;
        break;
      case ImageIO::YCbCr:
        cout << "Colormode: " << "YCbCr (YUV)" << endl;
        break;
      case ImageIO::YUY2:
        cout << "Colormode: " << "YUY2" << endl;
        break;
```

```
      case ImageIO::YCCK:
        cout << "Colormode: " << "YCCK" << endl;
        break;
      case ImageIO::CMYK:
        cout << "Colormode: " << "CMYK" << endl;
        break;
      case ImageIO::YVYU:
        cout << "Colormode: " << "YVYU" << endl;
        break;
      case ImageIO::UYVY:
        cout << "Colormode: " << "UYVY" << endl;
        break;
      case ImageIO::RLE8:
        cout << "Colormode: " << "RLE8" << endl;
        break;
      case ImageIO::RLE4:
        cout << "Colormode: " << "RLE4" << endl;
        break;
      case ImageIO::BITFIELDS:
        cout << "Colormode: " << "BITFIELDS" << endl;
        break;
      case ImageIO::YVU9:
        cout << "Colormode: " << "YVU9" << endl;
        break;
      case ImageIO::YV12:
        cout << "Colormode: " << "YV12" << endl;
        break;
      case ImageIO::I420:
        cout << "Colormode: " << "I420" << endl;
        break;
      case ImageIO::IYUV:
        cout << "Colormode: " << "IYUV" << endl;
        break;
      case ImageIO::Y800:
        cout << "Colormode: " << "Y800" << endl;
        break;
      case ImageIO::Y8:
        cout << "Colormode: " << "Y8" << endl;
        break;
      case ImageIO::CIELAB:
        cout << "Colormode: " << "CIELAB" << endl;
        break;
      case ImageIO::ICCLAB:
        cout << "Colormode: " << "ICCLAB" << endl;
        break;
      case ImageIO::ITULAB:
        cout << "Colormode: " << "ITULAB" << endl;
        break;
      case ImageIO::LOGL:
        cout << "Colormode: " << "LOGL" << endl;
        break;
      case ImageIO::LOGLUV:
        cout << "Colormode: " << "LOGLUV" << endl;
        break;
      case ImageIO::MASK:
        cout << "Colormode: " << "MASK" << endl;
        break;
      case ImageIO::SEPARATED:
        cout << "Colormode: " << "SEPARATED" << endl;
        break;
      }

  }
  void pixelDepth( unsigned int pd){
    cout << "pixel depth: " << pd << endl;
  }
};


int usage()
{
  cerr << "usage: acquisition -cfg <config file> -img <image file> "
       << "[ -cbeff <cbeff file> [ -token <token file>]]" << endl
       << "[-mineye <relative minimal eye distance>]" << endl
       << "[-maxeye <relative maximal eye distance>]" << endl
       << "\tconfig file   ... frsdk configuration file" << endl
       << "\timage file    ... an image source file name (BMP, JPEG, JPEG2000, PGM or PNG format)" << endl
       << "\tcbeff file    ... a cbeff destination file name" << endl
       << "\ttoken file    ... a token face image destination file name (PNG format)"
       << endl
       << "\trelative minimal eye distance ... the minimal eye distance to look for, relative to image
       width, default: 0.1" << endl
       << "\trelative maximal eye distance ... the maximal eye distance to look for, relative to image
       width, default: 0.4" << endl << endl;
  return 1;
}
```

```
int main( int argc, const char* argv[] )
{
  try {
    FRsdk::CmdLine cmd( argc, argv);
    if( cmd.hasflag("-h")) return usage();
    if( !cmd.getspaceflag("-cfg")) return usage();
    if( !cmd.getspaceflag("-img")) return usage();


    // initialize and resource allocation
    ifstream configIStream( cmd.getspaceflag("-cfg"), ios::in);
    FRsdk::Configuration cfg( configIStream);

    FRsdk::Face::Finder faceFinder( cfg);
    FRsdk::Eyes::Finder eyesFinder( cfg);
    FRsdk::Portrait::Analyzer portraitAnalyzer( cfg);
    FRsdk::ISO_19794_5::FullFrontal::Test iso19794Test( cfg);
    FRsdk::Portrait::Feature::Test featureTest( cfg);
    FRsdk::ISO_19794_5::TokenFace::Creator tfcreator( cfg);

    FRsdk::CountedPtr<FRsdk::Image> img;

    float mindist = 0.1f;
    if( cmd.getspaceflag("-mineye")){
      mindist = atof( cmd.getspaceflag("-mineye"));
    }
    float maxdist = 0.4f;
    if( cmd.getspaceflag("-maxeye")){
      maxdist = atof( cmd.getspaceflag("-maxeye"));
    }

    // try opening jpg image
    try {
      img = new FRsdk::Image
        ( FRsdk::Jpeg::load
          ( string ( cmd.getspaceflag("-img")),
            FRsdk::ImageIO::PropertiesFeedback( new ImageCoutFeedback)));
    }
    catch (exception& e) {}
#ifndef UNDER_CE
    // try opening jpg2000 image
    if (img == 0) try {
      img = new FRsdk::Image
        ( FRsdk::Jpeg2000::load
          ( string( cmd.getspaceflag("-img")),
            FRsdk::ImageIO::PropertiesFeedback( new ImageCoutFeedback)));
    }
    catch (exception& e) {}
#endif
    // try opening bmp image
    if (img == 0) try {
      img = new FRsdk::Image
        ( FRsdk::Bmp::load
          ( string( cmd.getspaceflag("-img")),
            FRsdk::ImageIO::PropertiesFeedback( new ImageCoutFeedback)));
    }
    catch (exception& e) {}
    // try opening pgm/ppm image
    if (img == 0) try {
      img = new FRsdk::Image
        ( FRsdk::Pgm::load
          ( string( cmd.getspaceflag("-img")),
            FRsdk::ImageIO::PropertiesFeedback( new ImageCoutFeedback)));
    }
    catch (exception& e) {}
    // try opening png file
    if (img == 0) try {
      std::ifstream pngFile;
      pngFile.open( cmd.getspaceflag("-img"), ios::binary);
      img = new FRsdk::Image
        ( FRsdk::Png::load
          ( pngFile,
            FRsdk::ImageIO::PropertiesFeedback( new ImageCoutFeedback)));
    }
    catch (exception& e) {}
    if (img == 0)
      cout << "<image file> contains no recognized image file format"
           << endl;

    // doing face finding
    FRsdk::Face::LocationSet faceLocations =
      faceFinder.find (*img, mindist, maxdist);

    if( faceLocations.size() < 1) {
      throw AcquisitionError( "Unable to locate face");
    } else {
```

```
    const FRsdk::Face::Location& l = faceLocations.front();
    cout << "Found face: " << l.pos <<", width="
         << l.width << ", confidence="
         << l.confidence << endl;
  }

  // doing eyes finding with the first face location
  // (the one with the highest confidence)
  FRsdk::Eyes::LocationSet eyesLocations =
    eyesFinder.find (*img, faceLocations.front());

  if( eyesLocations.size() < 1) {
    throw AcquisitionError( "Unable to locate eyes");
  } else {
    const FRsdk::Eyes::Location& l = eyesLocations.front();
    cout << "Found eyes: first " << l.first
         << " confidence=" << l.firstConfidence << endl
         << "             second " << l.second
         << " confidence=" << l.secondConfidence
         << endl;
  }

  // now lets bundle the annotated image
  FRsdk::AnnotatedImage annotatedImage( *img, eyesLocations.front());

  // analyse portrait characteristics
  FRsdk::Portrait::Characteristics pc =
    portraitAnalyzer.analyze( annotatedImage);
  cout << pc << endl;

  // Test features
  FRsdk::Portrait::Feature::Set features = featureTest.assess( pc);
  cout << features << endl;

  // Test compliance with ISO 19794-5 Full Frontal image requirements
  FRsdk::ISO_19794_5::FullFrontal::Compliance
    isoCompliance = iso19794Test.assess( pc);
  printComplianceResult( cout, isoCompliance, pc );

  printBoundaries( cout,  iso19794Test.boundaries(), pc);
  cout << endl;

  if( !isoCompliance.isCompliant()) {
    cout << "Acquired image " << cmd.getspaceflag("-img")
         << " not compliant with ISO 19794-5 requirements"
         << endl;
  } else {
    cout << "Acquired image " << cmd.getspaceflag("-img")
         << " compliant with ISO 19794-5 requirements" << endl;
  }

  if( !isoCompliance.isBestPractice()) {
    cout << "Acquired image " << cmd.getspaceflag("-img")
         << " not compliant with ISO 19794-5 requirements and recommendations of best practices"
         << endl;
  } else {
    cout << "Acquired image " << cmd.getspaceflag("-img")
         << " compliant with ISO 19794-5 requirements and recomendations of best practices" << endl;
  }

  // create a Token Face Image according to ISO 19794-5
  FRsdk::AnnotatedImage iso19794Img = tfcreator.extract( annotatedImage);

  // store the portrait to file using CBEFF complient Token Face Image
  // format
  FRsdk::AnnotatedImageSet annotatedImages;
  annotatedImages.push_back( iso19794Img);


  if( cmd.getspaceflag("-cbeff")) {
    std::ofstream out( cmd.getspaceflag("-cbeff"),
                       std::ios::out | std::ios::binary);
    FRsdk::ISO_19794_5::TokenFace::write( out, annotatedImages);
    if( cmd.getspaceflag("-token") ) {
      std::ofstream outimg (cmd.getspaceflag("-token"),
                            std::ios::out | std::ios::binary);
      // FRsdk::Jpeg::save (iso19794Img.first, outimg); // save jpeg image
      // FRsdk::Bmp::save (iso19794Img.first,
      //  std::string (cmd.getspaceflag("-token"))); // save bmp image
      // FRsdk::Pgm::save (iso19794Img.first,
      //  std::string (cmd.getspaceflag("-token"))); // save pgm image
      FRsdk::Png::save (iso19794Img.first, outimg); // save png image
    }
  }

  cout << "Acquisition process done" << endl;
```

```
  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const AcquisitionError& e) {
    cout << "Acquisition Error: " << e.what() << endl;
    return EXIT_SUCCESS;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( ... ) {
#ifdef UNDER_CE
    printMemoryStatus( "Characteristics: " );
#endif
    cout << "caught unknown exception"  << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.2 capdev.cc

For a detailed explanation see Tutorial - How to write a CaptureDevice

```
// -*- c++ -*-
// Copyright (c) 2002 Cognitec Systems GmbH
//
// $Revision: 1.1 $
//

#include <frsdk/config.h>
#include <frsdk/jpeg.h>
#include <frsdk/bmp.h>
#include <frsdk/pgm.h>

#include <frsdk/capdev.h>

#include <stdio.h>
#include <cstdlib>
#include <fstream>
#include <iostream>
#include <exception>
#include <list>

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "capdev {config file} [device name]"
       << endl << endl
       << "\tconfig file ... the frsdk config file" << endl
       << "\tdevice name ... the name of the capture device"
       << endl << endl;
  return 1;
}

using namespace FRsdk;


int main( int argc, char** argv)
{
  try {

    if( argc < 2 || argc > 3 ) return usage();

    // initialize and resource allocation
    Configuration cfg( argv[1]);

    cout << "creating capture device" << endl;

    string devname;
    if( argc == 3 )
      devname = argv[2];
```

```
    CaptureDevice capDev = createCaptureDevice( cfg, devname);
    cout << "done" << endl;

    cout << "capturing 1st image..." << endl;
    Image img = capDev.capture();
    cout << "done, got image of size " << img.width() << " x "
         << img.height() <<endl;
    cout << "capturing 2nd image..." << endl;
    img = capDev.capture();
    cout << "done" << endl;
    cout << "capturing 3rd image..." << endl;
    img = capDev.capture();
    cout << "done" << endl;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    // don't exit with error code here.
    // if the process gets here, it could be loaded and startet, what to test
    // is the main objective of this exeutable
  }

  return EXIT_SUCCESS;
}
```

0.38.3 cropfullfrontal.cc

For a detailed explanation see Tutorial - Simple application to crop full frontal images.

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.19 $
//


#include <frsdk/cbeff.h>
#include <frsdk/config.h>
#include <frsdk/eyes.h>
#include <frsdk/jpeg.h>
#include <frsdk/portrait.h>
#include <frsdk/portraittests.h>
#include <frsdk/fullfrontal.h>
#include <frsdk/tokenface.h>

#include "cmdline.h"

#include <sstream>
#include <fstream>
#include <exception>
#include <cstdlib>

using namespace std;

ostream& operator<<( ostream& o, const FRsdk::Position& p)
{
  o << "[" << p.x() << ", " << p.y() << "]";
  return o;
}

namespace {
  class AcquisitionError: public std::exception
  {
  public:
    AcquisitionError( const std::string& msg_) throw(): msg( msg_) {}
    ~AcquisitionError() throw() { }
    const char* what() const throw() { return msg.c_str(); }
  private:
    std::string msg;
  };
}


int usage()
{
  cerr << "usage: cropfullfrontal -cfg <frsdk configuration file> "
       << "-img <jpeg file> " << endl
       << "[-out <full frontal image file>]" << endl
       << "[-mineye <relative minimal eye distance>]" << endl
       << "[-maxeye <relative maximal eye distance>]" << endl
       << endl;
  return 1;
}
```

```
int main( int argc, const char* argv[] )
{
  try {
    FRsdk::CmdLine cmd( argc, argv);
    if( cmd.hasflag("-h")) return usage();
    if( !cmd.getspaceflag("-cfg")) return usage();
    if( !cmd.getspaceflag("-img")) return usage();

    // initialize and resource allocation
    ifstream configIStream( cmd.getspaceflag("-cfg"), ios::in);
    FRsdk::Configuration cfg( configIStream);

    FRsdk::Face::Finder faceFinder( cfg);
    FRsdk::Eyes::Finder eyesFinder( cfg);
    FRsdk::Portrait::Analyzer portraitAnalyzer( cfg);
    FRsdk::ISO_19794_5::FullFrontal::Test iso19794Test( cfg);
    FRsdk::Portrait::Feature::Test featureTest( cfg);
    FRsdk::ISO_19794_5::FullFrontal::Creator ffcreator( cfg);


    FRsdk::Image img( FRsdk::Jpeg::load( string( cmd.getspaceflag("-img"))));

    float mindist = 0.1f;
    if( cmd.getspaceflag("-mineye")){
      mindist = atof( cmd.getspaceflag("-mineye"));
    }
    float maxdist = 0.4f;
    if( cmd.getspaceflag("-maxeye")){
      maxdist = atof( cmd.getspaceflag("-maxeye"));
    }


    // doing face finding
    FRsdk::Face::LocationSet faceLocations =
      faceFinder.find( img, mindist, maxdist);

    if( faceLocations.size() < 1) {
      throw AcquisitionError( "Unable to locate face");
    } else {
      const FRsdk::Face::Location& l = faceLocations.front();
      cout << "Found face: " << l.pos <<", width="
           << l.width << ", confidence="
           << l.confidence << endl;
    }

    // doing eyes finding with the first face location
    // (the one with the highest confidence)
    FRsdk::Eyes::LocationSet eyesLocations =
      eyesFinder.find( img, faceLocations.front());

    if( eyesLocations.size() < 1) {
      throw AcquisitionError( "Unable to locate eyes");
    } else {
      const FRsdk::Eyes::Location& l = eyesLocations.front();
      cout << "Found eyes: [ first " << l.first
           << " confidence=" << l.firstConfidence
           << ", second " << l.second
           << " confidence=" << l.secondConfidence
           << "]" << endl;
    }

    // now lets bundle the annotated image
    FRsdk::AnnotatedImage annotatedImage( img, eyesLocations.front());

    annotatedImage = ffcreator.extract( annotatedImage);

    // analyse portrait characteristics
    FRsdk::Portrait::Characteristics pc =
      portraitAnalyzer.analyze( annotatedImage);

    // Test features
    FRsdk::Portrait::Feature::Set features = featureTest.assess( pc);

    // Glasses
    if( features.wearsGlasses()) {
      cout << "Feature test: Person with glasses. (" << pc.glasses()
           << ")" << endl;
    } else {
      cout << "Feature test: Person without glasses. (" << pc.glasses()
           << ")" << endl;
    } // Glasses


    // Test compliance with ISO 19794-5 Full Frontal image requirements
    FRsdk::ISO_19794_5::FullFrontal::Compliance isoCompliance =
```

```
      iso19794Test.assess( pc);

    if( !isoCompliance.onlyOneFaceVisible()) {
      cout << "More than one face is visible!" << endl;
    }
    if( !isoCompliance.goodVerticalFacePosition()) {
      cout << "Bad vertical face position!" << endl;
    }
    if( !isoCompliance.horizontallyCenteredFace()) {
      cout << "Face not centered horizontally!" << endl;
    }
    if( !isoCompliance.widthOfHead())  {
        cout << "Bad sizing (Width)!" << endl;
    }
    if( !isoCompliance.lengthOfHead()) {
      cout << "Bad sizing (Height)!" << endl;
    }
    if( !isoCompliance.resolution()) {
      cout << "Bad resolution (not enough pixels of head width)!" << endl;
    }
    if( !isoCompliance.goodExposure()) {
      cout << "Bad exposure!" << endl;
    }
    if( !isoCompliance.goodGrayScaleProfile()) {
      cout << "Gray scale profile is not good!" << endl;
      cout << "Gray scale density: " << pc.grayScaleDensity() << endl;
    }
    if ( pc.isColor() ) {
      if( !isoCompliance.hasNaturalSkinColour()) {
        cout << "No natural Skin colour!" << endl;
        cout << "Natural skin colour: " << pc.naturalSkinColour() << endl;
      }
    }
    if( !isoCompliance.noHotSpots()) {
      cout << "Hot spots!" << endl;
      cout << "Hot spots: " << pc.hotSpots() << endl;
    }
    if( !isoCompliance.isFrontal()) {
      cout << "Face is not frontal!" << endl;
      cout << "Deviation from frontal pose: "
           << pc.deviationFromFrontalPose() << endl;
      cout << "Pose angle roll: " << pc.poseAngleRoll() << endl;
    }
#ifndef UNDER_CE
    if( !isoCompliance.isLightingUniform()) {
      cout << "Lighting is not uniform!" << endl;
      cout << "Deviation from uniform lighting: "
           << pc.deviationFromUniformLighting() << endl;
    }
#endif
    if( !isoCompliance.isSharp()) {
      cout << "Sharpness does not fit requirements!" << endl;
      cout << "Sharpness: " << pc.sharpness() << endl;
    }
    if( !isoCompliance.noTintedGlasses()) {
      cout << "Tinted glasses!" << endl;
      cout << "Tinted Eyes confidence: ["
           << pc.eye0Tinted() << ", " << pc.eye1Tinted() << "]" << endl;
    }
    if( !isoCompliance.isCompliant()) {
      cout << "Acquired image not compliant with ISO 19794-5 requirements!"
           << endl;
    } else {
      cout << "Cropped image seems to be ISO 19794-5 compliant." << endl;
    }

    cout << "Processing done" << endl;

    if( cmd.getspaceflag("-out") ){
      std::ofstream outimg( cmd.getspaceflag("-out"),
                            std::ios::out | std::ios::binary);
      FRsdk::Jpeg::save( annotatedImage.first, outimg);
      cout << "Image storing done" << endl;
    }
  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const AcquisitionError& e) {
    cout << "Acquisition Error: " << e.what() << endl;
    return EXIT_FAILURE;
  }
```

```
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.4 edialog.h

The following example demonstrates how to write simple enrollment feedback class. For a detailed explanation see Tutorial - Making set enrollments .

```
// -*- c++ -*-
// Copyright (c) 2002 Cognitec Systems GmbH
//
// $Revision: 1.1 $
//

#ifndef EDIALOG_H
#define EDIALOG_H

#include <frsdk/enroll.h>
#include <frsdk/cptr.h>
#include <fstream>
#include <iostream>

// small helper for tracing purpose
std::ostream&
operator<<( std::ostream& o, const FRsdk::Position& p)
{
  o << "[" << p.x() << ", " << p.y() << "]";
  return o;
}

namespace {
  class InvalidFIRAccessError: public std::exception
  {
  public:
    InvalidFIRAccessError() throw():
      msg("Trying to access invalid FIR") {}
    ~InvalidFIRAccessError() throw() { }
    const char* what() const throw() { return msg.c_str(); }
  private:
    std::string msg;
  };
}

// the concrete feedback which prints to stdout
class EnrolCoutFeedback : public FRsdk::Enrollment::FeedbackBody
{
public:
  EnrolCoutFeedback( const std::string& firFilename)
    : firFN( firFilename), firvalid(false) { }
  ~EnrolCoutFeedback() {}

  // the feedback interface
  void start() {
    firvalid = false;
    std::cout << "start" << std::endl;
  }

  void processingImage( const FRsdk::Image& img)
  {
    std::cout << "processing image[" << img.name() << "]" << std::endl;
  }

  void eyesFound( const FRsdk::Eyes::Location& eyeLoc)
  {
    std::cout << "found eyes at ["<< eyeLoc.first
              << " " << eyeLoc.second << "; confidences: "
              << eyeLoc.firstConfidence << " "
              << eyeLoc.secondConfidence << "]" << std::endl;
  }

  void eyesNotFound()
  {
    std::cout << "eyes not found" << std::endl;
  }

  void sampleQualityTooLow() {
    std::cout << "sampleQualityTooLow" << std::endl;
  }
```

```
  void sampleQuality( const float& f) {
    std::cout << "Sample Quality: " << f << std::endl;
  }

  void success( const FRsdk::FIR& fir_)
  {
    fir = new FRsdk::FIR(fir_);
    std::cout
      << "successful enrollment";
    if(firFN != std::string("")) {

      std::cout << " FIR[filename,id,size] = [\""
      << firFN.c_str() << "\",\"" << (fir->version()).c_str() << "\","
      << fir->size() << "]";
      // write the fir
      std::ofstream firOut( firFN.c_str(),
                            std::ios::binary|std::ios::out|std::ios::trunc);
      firOut << *fir;
    }
    firvalid = true;
    std::cout << std::endl;
  }

  void failure() { std::cout << "failure" << std::endl; }

  void end() { std::cout << "end" << std::endl; }

  const FRsdk::FIR& getFir() const {
    // call only if success() has been invoked
    if(!firvalid)
      throw InvalidFIRAccessError();

    return *fir;
  }

  bool firValid() const {
    return firvalid;
  }

private:
  FRsdk::CountedPtr<FRsdk::FIR> fir;
  std::string firFN;
  bool firvalid;
};

#endif
```

0.38.5 enroll.cc

The following example demonstrates how to write a simple enroller. For a detailed explanation see Tutorial - Making set enrollments .

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.29 $
//

#include <exception>
#include <list>
#include <cstdlib>

#include <frsdk/config.h>
#include <frsdk/image.h>
#include <frsdk/enroll.h>

#include "cmdline.h"
#include "edialog.h"

using namespace std;

int usage()
{
  cerr << "usage:\n"
       << "enroll -cfg <config file> -fir <fir> "
          "-imgs <image0,image1,image2,...>\n\n"
       << "\tconfig file ... the frsdk config file\n"
       << "\tfir         ... a filename for a FIR\n"
       << "\tjpeg image  ... one or more jpeg images for enrollment.\n" << endl;
  return EXIT_FAILURE;
```

```
}

// main --------------------------------------------------------------------
int main( int argc, const char* argv[] )
{
  try {

    FRsdk::CmdLine cmd( argc, argv );
    if ( cmd.hasflag("-h")) return usage();
    if ( !cmd.getspaceflag("-cfg")) return usage();
    if ( !cmd.getspaceflag("-fir")) return usage();
    if ( !cmd.getspaceflag("-imgs")) return usage();

    // read the configuration file
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    // get the file names of the enrollment images
    string delimImgFileNames = cmd.getspaceflag( "-imgs" );
    list<string> imgFileNames =
      FRsdk::parseDelimitedText< list<string> >( delimImgFileNames );

    if ( imgFileNames.size() == 0 )
    {
      cerr << "There are no input images specified!" << endl;
      usage();
    }

    cout << "Loading the input images..." << endl;
    FRsdk::SampleSet enrollmentImages;
    list<string>::const_iterator it = imgFileNames.begin();
    while ( it != imgFileNames.end() )
    {
      const string& imgFileName = *it;
      cout  << "  \"" << imgFileName << "\"" << endl;
      FRsdk::Image img( FRsdk::ImageIO::load( imgFileName ) );
      enrollmentImages.push_back( FRsdk::Sample( img ) );
      ++it;
    }
    cout << "...Done.\n"
         << enrollmentImages.size() << " image(s) loaded." << endl;
    if ( enrollmentImages.size() == 0 )
    {
      cerr << "There are no samples to process!" << endl;
      return EXIT_FAILURE;
    }

    cout << "Start processing ... " << flush;

    // create an enrollment processor
    FRsdk::Enrollment::Processor proc( cfg);

    // create the needed interaction instances
    FRsdk::Enrollment::Feedback
      feedback( new EnrolCoutFeedback( cmd.getspaceflag("-fir")));

    // do the enrollment
    proc.process( enrollmentImages.begin(),
                  enrollmentImages.end(), feedback);
    cout << "...Done." << endl;

  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.6 eyesfind.cc

The following example demonstrates how to write a simple eye finder. For a detailed explanation see Tutorial - Locating eyes .

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.37 $
//


#include <frsdk/config.h>
#include <frsdk/eyes.h>
#include <frsdk/jpeg.h>
#include "cmdline.h"
#include <iostream>
#include <exception>
#include <cstdlib>

using namespace std;

ostream& operator<<( ostream& o, const FRsdk::Position& p)
{
  o << "[" << p.x() << ", " << p.y() << "]";
  return o;
}

int
usage()
{
  cerr << "usage: eyesfind -cfg <config file> -img <jpeg file>"
       << endl
       << "[-mineye <relative minimal eye distance>]" << endl
       << "[-maxeye <relative maximal eye distance>]" << endl
       << endl
       << "\tconfig file  ... frsdk configuration file" << endl
       << "\tjpeg file    ... a jpeg image source file name " << endl
       << "\trelative minimal eye distance ... the minimal eye distance to look for, relative to image
       width, default: 0.1" << endl
       << "\trelative maximal eye distance ... the maximal eye distance to look for, relative to image
       width, default: 0.4" << endl << endl;
  return 1;
}

int main( int argc, const char* argv[] )
{
  try {

    FRsdk::CmdLine cmd( argc, argv);
    if( cmd.hasflag("-h")) return usage();
    if( !cmd.getspaceflag("-cfg")) return usage();
    if( !cmd.getspaceflag("-img")) return usage();

    // initialize and resource allocation
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    FRsdk::Face::Finder faceFinder (cfg);

    FRsdk::Eyes::Finder eyesFinder( cfg);

    FRsdk::Image img( FRsdk::Jpeg::load( string( cmd.getspaceflag("-img"))));

    float mindist = 0.1f;
    if( cmd.getspaceflag("-mineye")){
      mindist = atof( cmd.getspaceflag("-mineye"));
    }
    float maxdist = 0.4f;
    if( cmd.getspaceflag("-maxeye")){
      maxdist = atof( cmd.getspaceflag("-maxeye"));
    }

    // doing face finding
    FRsdk::Face::LocationSet faceLocations =
      faceFinder.find (img, mindist, maxdist);

    std::cout << "number of faces found: " << faceLocations.size () << endl;
    std::cout << "----------------------" << endl;

    FRsdk::Face::LocationSet::const_iterator faceIter = faceLocations.begin();

    while( faceIter != faceLocations.end()) {

      cout << "Face location: " << (*faceIter).pos << endl;

      // doing eyes finding
      FRsdk::Eyes::LocationSet eyesLocations =
        eyesFinder.find( img, *faceIter);

      FRsdk::Eyes::LocationSet::const_iterator eyesIter =eyesLocations.begin();
      while( eyesIter != eyesLocations.end()) {
```

```
        cout << "Eye locations: [ first " << (*eyesIter).first
             << " confidence=" << (*eyesIter).firstConfidence
             << ", second " << (*eyesIter).second
             << " confidence=" << (*eyesIter).secondConfidence
             << "]" << endl;
        eyesIter++;
      }
      if (eyesLocations.size () == 0) std::cout << "--- no eyes found! ---" << endl;

      std::cout << "----------------------" << endl;
      faceIter++;
    }
  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.7 facefind.cc

The following example demonstrates how to write a simple face finding program. For a detailed explanation see Tutorial - Finding faces .

```
// -*- c++ -*-
// Copyright (c) 2002 Cognitec Systems GmbH
//
// $Revision: 1.9 $

//


#include <frsdk/config.h>
#include <frsdk/face.h>
#include <frsdk/jpeg.h>
#include <iostream>
#include <iomanip>
#include <exception>
#include "../image.h"

using namespace std;
using namespace Fn;


int usage()
{
  cerr << "usage: facefind <frsdk configuration file>" << endl;
  return 1;
}

int main( int argc, char** argv)
{

  unsigned int i = 0;
  try {

    if( argc != 2) return usage();

    // initialize and resource allocation
    FRsdk::Configuration cfg( argv[1]);
    FRsdk::Face::Finder faceFinder( cfg);
    cout << "searching faces in Images of dimensions:" << endl;

    for( ; i <= 100; i++) {
      try{
        FNImageReference fnimg( FNImageReferenceFactory::create
                                ( FNImage<FNByte>( 10000, i)));
        cout << i << ": [" << fnimg.w() << "," << fnimg.h() << "] " << endl;
        cout.flush();
        FRsdk::Image img( new FRsdk::ImageFN( fnimg));

        // doing face finding
        FRsdk::Face::LocationSet locations = faceFinder.find( img);
```

```
        FRsdk::Face::LocationSet::const_iterator faceIter = locations.begin();
        while( faceIter != locations.end()) {
          cout << "Face location: [" << (*faceIter).pos.x() << ", "
               << (*faceIter).pos.y() << "]" << endl;
          faceIter++;
        }
      }
      catch( std::exception& e){
        cout << "Error at index: " << i << ", " << e.what() << endl;
      }
      catch( ... ){
        cout << "Unknown Error at index: " << i << endl;
      }
    }
    for( i=0 ; i <= 100; i++) {
      try{
        FNImageReference fnimg( FNImageReferenceFactory::create
                                ( FNImage<FNByte>( i, 10000)));
        cout << i << ": [" << fnimg.w() << "," << fnimg.h() << "] " << endl;
        cout.flush();
        FRsdk::Image img( new FRsdk::ImageFN( fnimg));

        // doing face finding
        FRsdk::Face::LocationSet locations = faceFinder.find( img);

        FRsdk::Face::LocationSet::const_iterator faceIter = locations.begin();
        while( faceIter != locations.end()) {
          cout << "Face location: [" << (*faceIter).pos.x() << ", "
               << (*faceIter).pos.y() << "]" << endl;
          faceIter++;
        }
      }
      catch( std::exception& e){
        cout << "Error at index: " << i << ", " << e.what() << endl;
      }
      catch( ... ){
        cout << "Unknown Error at index: " << i << endl;
      }
    }
    for( i=0 ; i <= 100; i++) {
      try{
        FNImageReference fnimg( FNImageReferenceFactory::create
                              ( FNImage<FNByte>( i*10, i*10)));
        cout << i << ": [" << fnimg.w() << "," << fnimg.h() << "] " << endl;
        cout.flush();
        FRsdk::Image img( new FRsdk::ImageFN( fnimg));

        // doing face finding
        FRsdk::Face::LocationSet locations = faceFinder.find( img);

        FRsdk::Face::LocationSet::const_iterator faceIter = locations.begin();
        while( faceIter != locations.end()) {
          cout << "Face location: [" << (*faceIter).pos.x() << ", "
               << (*faceIter).pos.y() << "]" << endl;
          faceIter++;
        }
      }
      catch( std::exception& e){
        cout << "Error at index: " << i << ", " << e.what() << endl;
      }
      catch( ... ){
        cout << "Unknown Error at index: " << i << endl;
      }
    }

  }
  catch( exception& e) {
    cout << "Error: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  cout << endl;

  return EXIT_SUCCESS;
}
```

0.38.8 identify.cc

The following example demonstrates how to write a simple identifier. For a detailed explanation see Tutorial - Making set identifications .

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
```

```
// All rights reserved.
//
// $Revision: 1.33 $
//


#include <frsdk/config.h>
#include <frsdk/jpeg.h>
#include <frsdk/ident.h>
#include "cmdline.h"
#include <iostream>
#include <exception>
#include <list>
#include <string>
#include <cstdlib>

#include "idialog.h"

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "identify -cfg <config file> -img <jpeg image> -firs <fir0,fir1,fir2,...> [-thr <threshold> |
       -far <requestedFAR> | -frr <requestedFRR>] [-maxmatch <number of matches>]"
       << endl << endl
       << "\tconfig file ... the frsdk config file" << endl
       << "\tjpeg image  ... a jpeg image file to be processed"
       << endl << endl
       << "\tfir         ... one or more FIR files to build "
       << "identification population." << endl
       << "\tthreshold   ... threshold for successfull matches (default is score for FAR of 0.001)" << endl
       << "\trequestedFAR ... request score for this FAR" << endl
       << "\trequestedFRR ... request score for this FRR" << endl
       << "\tnumber of matches ... maximal number of matches in returneed match list"
       << endl << endl;
  return 1;
}

// main ----------------------------------------------------------------------
int main( int argc, const char* argv[] )
{
  try {
    FRsdk::CmdLine cmd( argc, argv);
    if( cmd.hasflag("-h")) return usage();
    if( !cmd.getspaceflag("-cfg")) return usage();
    if( !cmd.getspaceflag("-img")) return usage();
    if( !cmd.getspaceflag("-firs")) return usage();

    // initialize and resource allocation
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    // load the image for identification
    FRsdk::SampleSet identificationImages;

    identificationImages.push_back
      ( FRsdk::Sample(FRsdk::Jpeg::load( string( cmd.getspaceflag("-img")))));

    // load the fir population for identification
    FRsdk::Population population( cfg);
    FRsdk::FIRBuilder firBuilder( cfg);

    std::string firs = cmd.getspaceflag("-firs");
    size_t pos = 0;
    size_t fpos = 0;

    while( (pos = firs.find(',', fpos)) != std::string::npos){
      std::string firn = firs.substr( fpos, pos-fpos);
      fpos = pos +1;
      cout  << "[" << firn << "]" << endl;
      ifstream firIn( firn.c_str(), ios::in|ios::binary);
      population.append( firBuilder.build( firIn), firn.c_str());
    }
    if( fpos < firs.size()){
      std::string firn = firs.substr( fpos, pos-fpos);
      cout  << "[" << firn << "]" << endl;
      ifstream firIn( firn.c_str(), ios::in|ios::binary);
      population.append( firBuilder.build( firIn), firn.c_str());
    } // population complete


    // request Score match list size
    FRsdk::ScoreMappings sm( cfg);
    FRsdk::Score score = sm.requestFAR( 0.001f);
    if( cmd.getspaceflag("-far")){
      if( cmd.getspaceflag("-frr") || cmd.getspaceflag("-score")) return usage();
    }
```

```
    if( cmd.getspaceflag("-frr")){
      if( cmd.getspaceflag("-score")) return usage();
    }

    if( cmd.getspaceflag("-frr")){
      score = sm.requestFRR( atof( cmd.getspaceflag("-frr")));
    }
    if( cmd.getspaceflag("-far")){
      score = sm.requestFAR( atof( cmd.getspaceflag("-far")));
    }
    if( cmd.getspaceflag("-score")){
      score = FRsdk::Score( atof( cmd.getspaceflag("-score")));
    }
    cout << "used matching threshold: " << score << endl;

    unsigned int numofmatches = 3;
    if( cmd.getspaceflag("-maxmatch")){
      numofmatches = atoi( cmd.getspaceflag("-maxmatch"));
    }
    cout << "maximal matchlist size:" << numofmatches << endl;


    // create the needed interaction instances
    FRsdk::Identification::Feedback feedback( new IdentifyCoutFeedback());

    cout << "start processing ..." << endl;

    // create an identification processor
    FRsdk::Identification::Processor proc( cfg, population);

    // do the identification
    proc.process( identificationImages.begin(), identificationImages.end(),
                  score, feedback, numofmatches);

  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.9 idialog.h

The following example demonstrates how to write simple identification feedback class. For a detailed explanation see Tutorial - Making set identifications .

```
// -*- c++ -*-
// Copyright (c) 2002 Cognitec Systems GmbH
//
// $Revision: 1.1 $
//

#ifndef IDIALOG_H
#define IDIALOG_H

#include <frsdk/ident.h>
#include <fstream>
#include <iostream>

// small helper for tracing purpose
std::ostream&
operator<<( std::ostream& o, const FRsdk::Position& p)
{
  o << "[" << p.x() << ", " << p.y() << "]";
  return o;
}

// the concrete feedback which prints to stdout
class IdentifyCoutFeedback : public FRsdk::Identification::FeedbackBody
{
public:
  ~IdentifyCoutFeedback() {}
```

```
  // the feedback interface
  void start() { std::cout << "start" << std::endl; }

  void processingImage( const FRsdk::Image& img)
  {
    std::cout << "processing image[" << img.name() << "]" << std::endl;
  }

  void eyesFound( const FRsdk::Eyes::Location& eyeLoc)
  {
    std::cout << "found eyes at ["<< eyeLoc.first
              << " " << eyeLoc.second << "; confidences: "
              << eyeLoc.firstConfidence << " "
              << eyeLoc.secondConfidence << "]" << std::endl;
  }

  void eyesNotFound()
  {
    std::cout << "eyes not found" << std::endl;
  }

  void sampleQuality( const float& f) {
    std::cout << "Sample Quality: " << f << std::endl;
  }

  void sampleQualityTooLow()
  {
    std::cout << "sampleQualityTooLow" << std::endl;
  }

  void matches( const FRsdk::Matches& matches)
  {
    FRsdk::Matches::const_iterator iter = matches.begin();

    while( iter != matches.end()) {
      FRsdk::Match m = *iter;
      std::cout << "match on fir[" << m.first << "] got Score["
           << (float)m.second << "]" << std::endl;
      iter++;
    }
  } //matches

  void end() { std::cout << "end" << std::endl; }

};

#endif
```

0.38.10 imagebody.cc

The following example demonstrates how to write a customized FRsdk::ImageBody to support custom image formats.FaceVACS-SDK. For a detailed explanation see How to write an ImageBody .

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.4 $
//


#include <exception>

using namespace std;

#include <frsdk/cptr.h>
#include <frsdk/image.h>

namespace FRsdk {
  class MemoryImageBody: public ImageBody
  {
  public:
    // construct memory image from an RGB array.
    // memory will be copied (i.e. not owned)
    MemoryImageBody( const Rgb* b, unsigned int w, unsigned int h,
                     const std::string& name = "");
    ~MemoryImageBody();
    unsigned int height() const  { return h; }
    unsigned int width() const  { return w; }
    const Byte* grayScaleRepresentation() const { return b; }
    const Rgb* colorRepresentation() const  { return rgb; }

    // extended interface
```

```
    std::string name() const {
      return n;
    }

  private:
    void buildRgbRepresentation( const Rgb* rgb_);
    void buildByteRepresentation();
    unsigned int w;
    unsigned int h;
    Byte* b;
    Rgb* rgb;
    std::string n;
  };

  MemoryImageBody::MemoryImageBody( const Rgb* rgb_,
                                    unsigned int w_,
                                    unsigned int h_,
                                    const std::string& name)
    : w(w_), h(h_), b(0), rgb(0), n(name)
  {
    buildRgbRepresentation(rgb_);
    buildByteRepresentation();
  }

  void
  MemoryImageBody::buildRgbRepresentation(const Rgb* rgb_)
  {
    rgb = new Rgb[w * h * sizeof(Rgb)];
    // this assumes that the rgb array rgb_ points to
    // has the layout required by FRsdk::Images:
    // each pixel is [Blue Green Red NotUsed]; no padding
    //
    // otherwise, copy pixel-wise and rearrange
    // color pixels and/or cut padding

    memcpy(rgb, rgb_, w * h * sizeof(Rgb));
  }

  void
  MemoryImageBody::buildByteRepresentation()
  {
    b = new Byte[w * h ];

    Rgb* colorp = rgb;
    Byte* grayp = b;

    for( unsigned int i = 0; i < h; i++ )
      for( unsigned int k = 0; k < w; k++ ) {

        float f = (float) colorp->r;
        f += (float) colorp->g;
        f += (float) colorp->b;
        f /= 3.0f;

        if( f > 255.0f) f = 255.0f;

        *grayp = (Byte) f;
        colorp++;
        grayp++;
      }
  }

  MemoryImageBody::~MemoryImageBody()
  {
    delete[] rgb;
    delete[] b;
  }
}
```

0.38.11 match.cc

The following example illustrates the use of the low level match interfaces. For a detailed explanation see Tutorial - Using the low level match interfaces .

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.20 $
//

#include <frsdk/config.h>
```

```
#include <frsdk/match.h>
#include "cmdline.h"
#include <stdio.h>
#include <fstream>
#include <iostream>
#include <exception>
#include <list>
#include <cstdlib>

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "match -cfg <config file> -probe <fir> -gallery <fir0,fir1,fir2,...> [-thr <threshold> | -far
       <requestedFAR> | -frr <requestedFRR>] [-maxmatch <number of matches>]" << endl
       << endl << endl
       << "\tconfig file ... the frsdk config file" << endl
       << "\tfir         ... the fir to test against" << endl
       << "\tfir0,fir1,...  ... one or more FIR files for the population"
       << endl
       << "\tthreshold   ... threshold for successfull matches (default is score for FAR of 0.001)" << endl
       << "\trequestedFAR ... request score for this FAR" << endl
       << "\trequestedFRR ... request score for this FRR" << endl
       << "\tnumber of matches ... maximal number of matches in returneed match list"
       << endl << endl;
  return 1;
}

// main -----------------------------------------------------------------------
int main( int argc, const char* argv[] )
{
  try {

    FRsdk::CmdLine cmd( argc, argv);
    if( cmd.hasflag("-h")) return usage();
    if( !cmd.getspaceflag("-cfg")) return usage();
    if( !cmd.getspaceflag("-probe")) return usage();
    if( !cmd.getspaceflag("-gallery")) return usage();


    // initialize and resource allocation
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    // load the fir
    ifstream firStream( cmd.getspaceflag("-probe"), ios::in|ios::binary);
    FRsdk::FIRBuilder firBuilder( cfg);
    FRsdk::FIR fir = firBuilder.build( firStream);

    // load the fir population for identification
    std::string firs = cmd.getspaceflag("-gallery");
    size_t pos = 0;
    size_t fpos = 0;
    FRsdk::Population population( cfg);
    while( (pos = firs.find(',', fpos)) != std::string::npos){
      std::string firn = firs.substr( fpos, pos-fpos);
      fpos = pos +1;
      cout  << "[" << firn << "]" << endl;
      ifstream firIn( firn.c_str(), ios::in|ios::binary);
      population.append( firBuilder.build( firIn), firn.c_str());
    }
    if( fpos < firs.size()){
      std::string firn = firs.substr( fpos, pos-fpos);
      cout  << "[" << firn << "]" << endl;
      ifstream firIn( firn.c_str(), ios::in|ios::binary);
      population.append( firBuilder.build( firIn), firn.c_str());
    }

    // request Score match list size
    FRsdk::ScoreMappings sm( cfg);
    FRsdk::Score score = sm.requestFAR( 0.001f);
    if( cmd.getspaceflag("-far")){
      if( cmd.getspaceflag("-frr") || cmd.getspaceflag("-score")) return usage();
    }
    if( cmd.getspaceflag("-frr")){
      if( cmd.getspaceflag("-score")) return usage();
    }

    if( cmd.getspaceflag("-frr")){
      score = sm.requestFRR( atof( cmd.getspaceflag("-frr")));
    }
    if( cmd.getspaceflag("-far")){
      score = sm.requestFAR( atof( cmd.getspaceflag("-far")));
    }
    if( cmd.getspaceflag("-score")){
      score = FRsdk::Score( atof( cmd.getspaceflag("-score")));
    }
```

```
    cout << "used matching threshold: " << score << endl;

    unsigned int numofmatches = 3;
    if( cmd.getspaceflag("-maxmatch")){
      numofmatches = atoi( cmd.getspaceflag("-maxmatch"));
    }
    cout << "maximal matchlist size:" << numofmatches << endl;

    // initialize matching facility
    FRsdk::FacialMatchingEngine me( cfg);

    // bestMatches() takes care about the configured number of threads
    // to be used in comparison algorithm.
    FRsdk::CountedPtr<FRsdk::Matches> matches =
      me.bestMatches( fir, population, FRsdk::Score( score), numofmatches);

    // print the match results
    for( FRsdk::Matches::const_iterator iter = matches->begin();
         iter != matches->end(); iter++) {
      FRsdk::Match match = *iter;
      cout << "[" << match.first << "] \t:" << match.second << endl;
    }

    //compare() does not care about the configured number of Threads
    //for the comparison algorithm. It uses always one thrad to
    //compare all inorder to preserve the order of the scores
    //according to the order in the population (orer of adding FIRs to
    //the population)
    FRsdk::CountedPtr<FRsdk::Scores> scores = me.compare( fir, population);

    // print the results
    unsigned int n = 0;
    for( FRsdk::Scores::const_iterator siter = scores->begin();
         siter != scores->end(); siter++) {
      cout << "[ #" << n++ << " ] \t:" << float( *siter) << endl;
    }


  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.12 tracklife.cc

The following example demonstrates the usage of the tracker interface if frames with a varying framerate are available.FaceVACS-SDK.

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.12 $
//

#include <frsdk/config.h>
#include <frsdk/tracker.h>
#include <frsdk/jpeg.h>
#include <frsdk/capdev.h>
#include "cmdline.h"

#ifdef WIN32
#include <windows.h>
#else
#include <sys/time.h>
#endif

#include <time.h>
#include <cstdlib>
#include <iostream>
#include <fstream>
```

```
#include <sstream>
#include <exception>
#include <list>

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "tracklife -cfg <config file> -dev <device name> [ -fr <framecount>]"
       << endl << endl
       << "\tconfig file ... the frsdk config file" << endl
       << "\tdevice name ... the symbolic name of the capture device " << endl
       << "\t                as used in the configuration editor" << endl
       << "\tframecount  ... each image after frame count processed " << endl
       << "\t                images, will be stored (default: 10)" << endl
       << endl;
  return 1;
}

namespace FRsdk {
  class RealTime {
  public:

    RealTime(){
#ifdef WIN32
      QueryPerformanceFrequency(&freqCtr);
      QueryPerformanceCounter(&timerCtr);
#else
      gettime( to_);
#endif

    }
    ~RealTime() {}

    void reset(){
#ifdef WIN32
      QueryPerformanceCounter(&timerCtr);
#else
      gettime( to_);
#endif
    }

    operator float() const
    {
#ifdef WIN32
      LARGE_INTEGER timerCtr1;
      QueryPerformanceCounter(&timerCtr1);
      LARGE_INTEGER diffCtr;
      diffCtr.QuadPart= 10000 *
        (timerCtr1.QuadPart - timerCtr.QuadPart);
      diffCtr.QuadPart /= freqCtr.QuadPart;
      return float(diffCtr.LowPart) / 10000.0f;
#else
      struct timeval t_; gettime( t_);
      return float( t_.tv_sec - to_.tv_sec) +
        (t_.tv_usec - to_.tv_usec) / 1000000.0;
#endif
    }

  private:
#ifdef WIN32
    LARGE_INTEGER timerCtr;
    LARGE_INTEGER freqCtr;
#else
    struct timeval to_;
    void gettime( struct timeval& t) const { gettimeofday( &t, 0);}
#endif
  };
}


int main( int argc, const char* argv[] )
{
  try {

    FRsdk::CmdLine cmd( argc, argv);
    if( cmd.hasflag("-h")) return usage();
    if( !cmd.getspaceflag("-cfg")) return usage();
    if( !cmd.getspaceflag("-dev")) return usage();


    // initialize and resource allocation
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    cout << "creating capture device for [" << cmd.getspaceflag("-dev") << "] ... ";
```

```
    typedef FRsdk::Face::Tracker Tracker;
    Tracker tracker( cfg);

    int framecount = 10;
    if( cmd.getspaceflag("-fr"))
      framecount = atoi( cmd.getspaceflag("-fr"));

    FRsdk::CaptureDevice capDev =
      FRsdk::createCaptureDevice( cfg, std::string( cmd.getspaceflag("-dev")));

    cout << "done" << endl;
    FRsdk::RealTime elapsed;

    int counter = 0;
    int imgsavecounter = 0;
    for(; true;) {
      counter ++;
      // grab image
      FRsdk::Image img = capDev.capture();
      if( !(counter%framecount)){
        imgsavecounter++;
        std::ostringstream ostr;
        ostr << imgsavecounter << ".jpg" << ends;
        FRsdk::Jpeg::save( img, ostr.str());
        cout << "stored image: " << imgsavecounter << endl;
      }

      // get time
      // track objects in frame
      unsigned int time = (unsigned int)(1000* float(elapsed));
      Tracker::Locations trackerInfos = tracker.processImage( img, time );
      // print time and locations
      cout << time << " ";
      std::list< FRsdk::Face::Location> controlFaces;
      Tracker::Locations::const_iterator it = trackerInfos.begin();
      while ( it != trackerInfos.end() )
      {
        const FRsdk::Face::Tracker::Location trackerInfo = *it;

        cout << trackerInfo.id << " : ["
             << trackerInfo.faceLocation.pos.x() << ", "
             << trackerInfo.faceLocation.pos.y() << "]  ";

        controlFaces.push_back( trackerInfo.faceLocation );
        ++it;
      }
      capDev.control( img, controlFaces );
      cout << " [Current Gain: " << capDev.gain()
           << ", Current Exposure: " << capDev.exposure()
           <<"]"  << endl;
      cout << endl;
    }
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.13 trackrec.cc

The following example demonstrates the usage of the tracker interface if frames with a constant framerate are available.FaceVACS-SDK.

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.7 $
//

#include <frsdk/config.h>
#include <frsdk/tracker.h>
#include <frsdk/jpeg.h>
#include <frsdk/j2k.h>
#include <frsdk/bmp.h>
#include <frsdk/png.h>
```

```
#include <frsdk/pgm.h>

#include <fstream>
#include <iostream>
#include <exception>
#include <list>
#include <cstdlib>

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "trackrec {config file} {image1} {image2} ... "
       << endl << endl
       << "\tconfig file ... the frsdk config file" << endl
       << "\timageX      ... images" << endl << endl;
  return 1;
}

class ImageLoadError: public std::exception
{
public:
  ImageLoadError( const std::string& msg_) throw(): msg( msg_) {}
  ~ImageLoadError() throw() { }
  const char* what() const throw() { return msg.c_str(); }
private:
  std::string msg;
};


FRsdk::Image loadImage( const string& fn)
{
  // try opening jpg image
  try { return FRsdk::Jpeg::load( fn); } catch( const exception&) {}
#ifndef UNDER_CE
  // try opening jpg2000 image
  try { return FRsdk::Jpeg2000::load( fn); } catch( const exception&) {}
#endif
  // try opening bmp image
  try { return FRsdk::Bmp::load( fn); } catch( const exception&) {}
  // try opening pgm/ppm image
  try { return FRsdk::Pgm::load( fn); } catch( const exception&) {}
  // try opening png file
  try {
    std::ifstream pngFile;
    pngFile.open( fn.c_str(), ios::binary);
    return FRsdk::Png::load( pngFile);
  } catch( const exception&) {}
  throw ImageLoadError( fn + string( " contains no recognized image file format" ));
}


int main( int argc, const char* argv[] )
{
  try {

    if( argc < 3) return usage();

    // initialize and resource allocation
    FRsdk::Configuration cfg( argv[1]);

    // initialize tracker
    typedef FRsdk::Face::Tracker Tracker;
    Tracker tracker( cfg);

    // iterate over given images
    for( int a = 2; a < argc; ++a) {
      const string fn = argv[ a];
      // load image
      FRsdk::Image img = loadImage( fn);
      // track objects in frame
      Tracker::Locations trackerInfos = tracker.processFrame( img );
      // print filename and locations
      cout << fn << " ";

      Tracker::Locations::const_iterator it = trackerInfos.begin();
      while ( it != trackerInfos.end() )
      {
        const FRsdk::Face::Tracker::Location trackerInfo = *it;

        cout << trackerInfo.id << " : ["
             << trackerInfo.faceLocation.pos.x() << ", "
             << trackerInfo.faceLocation.pos.y() << "]  ";
      }
      cout << endl;
    }
```

```
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cerr << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cerr << e.what() << endl;
    return EXIT_FAILURE;
  }
  return EXIT_SUCCESS;
}
```

0.38.14 vdialog.h

The following example demonstrates how to write simple verification feedback class.

```
// -*- c++ -*-
// Copyright (c) 2002 Cognitec Systems GmbH
//
// $Revision: 1.1 $
//

#ifndef VDIALOG_H
#define VDIALOG_H

#include <frsdk/verify.h>

#include <fstream>
#include <iostream>

// small helper for tracing purpose
std::ostream&
operator<<( std::ostream& o, const FRsdk::Position& p)
{
  o << "[" << p.x() << ", " << p.y() << "]";
  return o;
}

// the concrete feedback which prints to stdout
class VerifyCoutFeedback : public FRsdk::Verification::FeedbackBody
{
public:
  ~VerifyCoutFeedback() {}

  // the feedback interface
  void start() { std::cout << "start" << std::endl; }

  void processingImage( const FRsdk::Image& img)
  {
    std::cout << "processing image[" << img.name() << "]" << std::endl;
  }

  void eyesFound( const FRsdk::Eyes::Location& eyeLoc)
  {
    std::cout << "found eyes at ["<< eyeLoc.first
              << " " << eyeLoc.second << "; confidences: "
              << eyeLoc.firstConfidence << " "
              << eyeLoc.secondConfidence << "]" << std::endl;
  }

  void eyesNotFound() {
    std::cout << "eyes not found" << std::endl;
  }

  void sampleQualityTooLow()
  {
    std::cout << "sampleQualityTooLow" << std::endl;
  }

  void sampleQuality( const float& f) {
    std::cout << "Sample Quality: " << f << std::endl;
  }

  void match( const FRsdk::Score& s)
  {
    std::cout << "match got Score[" << (float)s << "]" << std::endl;
  }

  void success() {
    std::cout << "successful verification." << std::endl;
  }

  void failure() { std::cout << "failure" << std::endl; }
```

```
  void end() { std::cout << "end" << std::endl; }
};

#endif
```

0.38.15 verify.cc

The following example demonstrates how to write a simple verifier.For a detailed explanation see Tutorial - Making set verifications .

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.31 $
//

#include <exception>
#include <list>
#include <cstdlib>

#include <frsdk/config.h>
#include <frsdk/image.h>
#include <frsdk/verify.h>

#include "cmdline.h"
#include "vdialog.h"

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "verify -cfg <config file> -fir <fir>  -imgs <image0,image1,image2,...> "
       << "[-thr <threshold> | -far <requestedFAR> | -frr <requestedFRR>]"
       << endl << endl
       << "\tconfig file ... the frsdk config file" << endl
       << "\tfir         ... a filename for a FIR" << endl
       << "\timage0,image1,image2  ... one or more jpeg image files for verification."
       << endl
       << "\tthreshold   ... threshold for successfull matches (default is score for FAR of 0.001)" << endl
       << "\trequestedFAR ... request score for this FAR" << endl
       << "\trequestedFRR ... request score for this FRR" << endl
       << endl << endl;
  return 1;
}

int main( int argc, const char* argv[] )
{
  try {

    FRsdk::CmdLine cmd( argc, argv);
    if ( cmd.hasflag("-h")) return usage();
    if ( !cmd.getspaceflag("-cfg")) return usage();
    if ( !cmd.getspaceflag("-fir")) return usage();
    if ( !cmd.getspaceflag("-imgs")) return usage();

    // read the configuration file
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    // get the file names of the verification images
    string delimImgFileNames = cmd.getspaceflag( "-imgs" );
    list<string> imgFileNames =
      FRsdk::parseDelimitedText< list<string> >( delimImgFileNames );

    if ( imgFileNames.size() == 0 )
    {
      cerr << "There are no input images specified!" << endl;
      usage();
    }

    cout << "Loading the input images..." << endl;
    FRsdk::SampleSet verificationImages;
    list<string>:: const_iterator it = imgFileNames.begin();
    while ( it != imgFileNames.end() )
    {
      const string& imgFileName = *it;
      cout  << "  \"" << imgFileName << "\"" << endl;
      FRsdk::Image img( FRsdk::ImageIO::load( imgFileName ) );
      verificationImages.push_back( FRsdk::Sample( img ) );
      ++it;
    }
```

```
    cout << "...Done.\n"
         << verificationImages.size() << " image(s) loaded." << endl;
    if ( verificationImages.size() == 0 )
    {
      cerr << "There are no samples to process!" << endl;
      return EXIT_FAILURE;
    }

    // get the FIR to verify against
    cout << "reading fir " << cmd.getspaceflag("-fir") << " ... ";
    ifstream firIn( cmd.getspaceflag("-fir"), ios::in|ios::binary);
    FRsdk::FIRBuilder firBuilder( cfg);
    FRsdk::FIR fir = firBuilder.build( firIn);
    cout << "done" << endl;

    // request Score
    FRsdk::ScoreMappings sm( cfg);
    FRsdk::Score score = sm.requestFAR( 0.001f );
    if( cmd.getspaceflag("-far")){
      if( cmd.getspaceflag("-frr") || cmd.getspaceflag("-score")) return usage();
    }
    if( cmd.getspaceflag("-frr")){
      if( cmd.getspaceflag("-score")) return usage();
    }

    if( cmd.getspaceflag("-frr")){
      score = sm.requestFRR( atof( cmd.getspaceflag("-frr")));
    }
    if( cmd.getspaceflag("-far")){
      score = sm.requestFAR( atof( cmd.getspaceflag("-far")));
    }
    if( cmd.getspaceflag("-score")){
      score = FRsdk::Score( atof( cmd.getspaceflag("-score")));
    }
    cout << "required success score: " << score << endl;

    // create the needed interaction instances
    FRsdk::Verification::Feedback feedback( new VerifyCoutFeedback());

    cout << "Start processing ... " << flush;

    // create a verification processor
    FRsdk::Verification::Processor proc( cfg);

    // do the verification
    proc.process( verificationImages.begin(),
                  verificationImages.end(),
                  fir, score, feedback);
    cout << "...Done." << endl;

  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.16 verifyan.cc

The following example illustrates the use of images with annotated eye positions for verification. For a detailed explanation see Tutorial - Using eye annotations .

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.25 $
//


#include <frsdk/config.h>
//#include <frsdk/jpeg.h>
#include <frsdk/image.h>
```

```
#include <frsdk/verify.h>
#include "cmdline.h"

#include <iostream>
#include <exception>
#include <list>
#include <cstdlib>

#include "vdialog.h"

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "verifyan -cfg <config file> -fir <fir>  -imgs <image0,image1,image2,...> "
       << "[-mineye <relative minimal eye distance>] "
       << "[-maxeye <relative maximal eye distance>]"
       << "[-thr <threshold> | -far <requestedFAR> | -frr <requestedFRR>]"
       << endl << endl
       << "\tconfig file ... the frsdk config file" << endl
       << "\tfir         ... a filename for a FIR" << endl
       << "\timage0,image1,image2  ... one or more jpeg image files for verification."
       << endl
       << "\trelative minimal eye distance ... the minimal eye distance to look for, relative to image
       width, default: 0.1" << endl
       << "\trelative maximal eye distance ... the maximal eye distance to look for, relative to image
       width, default: 0.4" << endl
       << "\tthreshold   ... threshold for successfull matches (default is score for FAR of 0.001)" << endl
       << "\trequestedFAR ... request score for this FAR" << endl
       << "\trequestedFRR ... request score for this FRR" << endl
       << endl << endl;
  return 1;
}

int main( int argc, const char* argv[] )
{
  try {

    FRsdk::CmdLine cmd( argc, argv);
    if ( cmd.hasflag("-h")) return usage();
    if ( !cmd.getspaceflag("-cfg")) return usage();
    if ( !cmd.getspaceflag("-fir")) return usage();
    if ( !cmd.getspaceflag("-imgs")) return usage();

    // read the configuration file
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    FRsdk::Face::Finder faceFinder( cfg);
    FRsdk::Eyes::Finder eyesFinder( cfg);

    // get the file names of the verification images
    string delimImgFileNames = cmd.getspaceflag( "-imgs" );
    list<string> imgFileNames =
      FRsdk::parseDelimitedText< list<string> >( delimImgFileNames );

    if ( imgFileNames.size() == 0 )
    {
      cerr << "There are no input images specified!" << endl;
      usage();
    }

    FRsdk::SampleSet verificationSamples;
    list<string>::const_iterator it = imgFileNames.begin();
    for ( ; it != imgFileNames.end() ; ++it )
    {
      using namespace FRsdk;
      const string& imgFileName = *it;

      FRsdk::Image img( FRsdk::ImageIO::load( imgFileName ) );

      // make annotations using face and eyes finder
      FRsdk::Face::LocationSet faceLocations = faceFinder.find( img);
      FRsdk::Face::LocationSet::const_iterator faceIter =
        faceLocations.begin();

      while ( faceIter != faceLocations.end() )
      {
        cout << "Face location: " << (*faceIter).pos << endl;

        // doing eyes finding
        FRsdk::Eyes::LocationSet eyesLocations =
          eyesFinder.find( img, *faceIter);

        FRsdk::Eyes::LocationSet::const_iterator eyesIter =
          eyesLocations.begin();
        while( eyesIter != eyesLocations.end()) {
```

```
          cout << "Eye locations: [ first " << (*eyesIter).first
               << ", second " << (*eyesIter).second << "]" << endl;

          FRsdk::AnnotatedImage a(img, *eyesIter);
          verificationSamples.push_back( FRsdk::Sample( a));
          eyesIter++;
        }

        faceIter++;
      }
    }

    // get the FIR to verify against
    cout << "reading fir " << cmd.getspaceflag("-fir") << " ... ";
    ifstream firIn( cmd.getspaceflag("-fir"), ios::in|ios::binary);
    FRsdk::FIRBuilder firBuilder( cfg);
    FRsdk::FIR fir = firBuilder.build( firIn);
    cout << "done" << endl;

    // request Score
    FRsdk::ScoreMappings sm( cfg);
    FRsdk::Score score = sm.requestFAR( 0.001f);
    if( cmd.getspaceflag("-far")){
      if( cmd.getspaceflag("-frr") || cmd.getspaceflag("-score")) return usage();
    }
    if( cmd.getspaceflag("-frr")){
      if( cmd.getspaceflag("-score")) return usage();
    }

    if( cmd.getspaceflag("-frr")){
      score = sm.requestFRR( atof( cmd.getspaceflag("-frr")));
    }
    if( cmd.getspaceflag("-far")){
      score = sm.requestFAR( atof( cmd.getspaceflag("-far")));
    }
    if( cmd.getspaceflag("-score")){
      score = FRsdk::Score( atof( cmd.getspaceflag("-score")));
    }
    cout << "required success score: " << score << endl;


    // create the needed interaction instances
    FRsdk::Verification::Feedback feedback( new VerifyCoutFeedback());

    cout << "start processing ..." << endl;

    // create a verification processor
    FRsdk::Verification::Processor proc( cfg);

    // do the verification
    proc.process( verificationSamples.begin(),
                  verificationSamples.end(),
                  fir, score, feedback);

  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```

0.38.17 vignetting.cc

For a detailed explanation see Tutorial - How to use vignetting function to blend image borders

```
// -*- c++ -*-
// Copyright @ 2002-2009, Cognitec Systems AG
// All rights reserved.
//
// $Revision: 1.7 $
//
```

```
#include <frsdk/config.h>
#include <frsdk/jpeg.h>
#include <frsdk/vignetting.h>
#include "cmdline.h"
#include <iostream>
#include <exception>
#include <list>
#include <cstdlib>

using namespace std;

int usage()
{
  cerr << "usage:" << endl
       << "vignetting -cfg <config file> img <input jpeg file> -out <output jpeg basename>"
       << endl << endl
       << "\tconfig file     ... the frsdk config file" << endl
       << "\tinput jpeg file    ... jpeg image to be read" << endl
       << "\toutput jpeg basename  ... base name of jpeg file to be stored."
       << endl << endl;
  return 1;
}

int main( int argc, const char* argv[] )
{
  try {

    FRsdk::CmdLine cmd( argc, argv);
    if( cmd.hasflag("-h")) return usage();
    if( !cmd.getspaceflag("-cfg")) return usage();
    if( !cmd.getspaceflag("-img")) return usage();
    if( !cmd.getspaceflag("-out")) return usage();


    // initialize and resource allocation
    FRsdk::Configuration cfg( cmd.getspaceflag("-cfg"));

    FRsdk::Image inimg = FRsdk::Jpeg::load( string( cmd.getspaceflag("-img")));
    string fnamebase = cmd.getspaceflag("-out");

    FRsdk::Image outimg = FRsdk::vignetting
      ( inimg, FRsdk::Rgb( 0xff, 0xff, 0xff), 20, 10, FRsdk::Gaussian);
    string jpegfname = fnamebase + ".gaussian";
    FRsdk::Jpeg::save( outimg, jpegfname);

    outimg = FRsdk::vignetting
      ( inimg, FRsdk::Rgb( 0xff, 0xff, 0xff), 20, 10, FRsdk::Linear);
    jpegfname = fnamebase + ".linear";
    FRsdk::Jpeg::save( outimg, jpegfname);


    outimg = FRsdk::vignetting
      ( inimg, FRsdk::Rgb( 0xff, 0xff, 0xff), 20, 10, FRsdk::Fixed);
    jpegfname = fnamebase + ".fixed";
    FRsdk::Jpeg::save( outimg, jpegfname);


  }
  catch( const FRsdk::FeatureDisabled& e) {
    cout << "Feature not enabled: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( const FRsdk::LicenseSignatureMismatch& e) {
    cout << "License violation: " << e.what() << endl;
    return EXIT_FAILURE;
  }
  catch( exception& e) {
    cout << e.what() << endl;
    return EXIT_FAILURE;
  }

  return EXIT_SUCCESS;
}
```