FacET - Facial Expression Tracker

1.0.0

Generated by Doxygen 1.5.5

Tue Mar 24 22:42:26 2009

Contents

Chapter 1

Main Page

This document was generated for FacET (Facial Expression Tracker), a library of image processing procedures for detecting and parameterising face components (eg. eyes, eyebrows, lips, forehead wrinkles).

Such face features are useful for classification of human emotions based on facial expression.

The current version of FacET was developed by Marek Wnuk and is based on the application created for Marcin Namysl's Master thesis (supervised by Marek Wnuk, Wroclaw University of Technology, Institute of Computer Engineering, Control and Robotics):

„*Wykorzystanie systemu wizyjnego do rozpoznawania emocji czlowieka*" („*Vision system in human emotions recognition* ").

FacET is a library for detecting and parameterising face components.

1.1 FacET library

The main public function of the library is face features detection:

```
void detectFeat(IplImage *src, IplImage *dst);
```

Using the src as input image, it fills the facesList - list of facepar_t structures of features for all the faces detected in the source image:

```
std:: list<facepar_t> facesList;
```

Structure of the type facepar_t contains the face features values after detection:

```
roix    -- face ROI upper left corner X coordinate (pixels)
roiy    -- face ROI upper left corner Y coordinate (pixels)
angle   -- face declination angle (not verified, for future use)
LEbBnd  -- left eyebrow bend angle (top)
LEbDcl  -- left eyebrow declination angle (side)
LEyOpn  -- distance between the right eyelids (rel. eyeball subregion)
LEbHgt  -- distance between left pupil and eyebrow top (rel. eye subregion)
REbBnd  -- right eyebrow bend angle (top)
REbDcl  -- right eyebrow declination angle (side)
REyOpn  -- distance between the right eyelids (rel. eyeball subregion)
REbHgt  -- distance between right pupil and eyebrow top (rel. eye subregion)
LiAspt  -- aspect ratio of the lips bounding box (percents)
LLiCnr  -- Y position of the left corner of the lips (rel. lips bounding box)
RLiCnr  -- Y position of the right corner of the lips (rel. lips bounding box)
Wrnkls  -- number of horizontal wrinkles in the center of the forehead
Nstrls  -- nostrils baseline width (rel. face width)
TeethA  -- area of the visible teeth (rel. lips bounding box)
```

The dst image is used merely as an output for visualisation (a copy of input image with face regions markers). Both src and dst can point to the same image.

Reading the settings from config file is provided by:

```
bool readSettings(std::string file_name = "default.cfg"),
```

which returns true in case of successful reading, false otherwise.

Initialisation of the facesList is provided by:

```
void cleanFacesList().
```

The library is intended to be platform-independent. It does not contain any GUI components, but it makes use of OpenCV structures and image processing functions.

1.2 Demo application

In order to check the library functionality and to make its usage clear, a demo application is included, as an example of FacET library usage with a simple, text-based user interface.

It provides face features processing in the images captured from camera or a video/image file:

```
#----------------- FacET demo -----------------#
|       started at: 09.03.04.22.40.46          |
|                                              |
| (t) - Main program (mode : camera)           |
| (v) - Main program (mode : video file)       |
| (z) - Main program (mode : image file)       |
| (q) - Quit                                   |
|                                              |
#----------------------------------------------#

>
```

Camera interface and several image capture parameters can be set with command-line options (overriding the defaults, or config file settings):

```
Command line options:

-b INT set brightness
-c INT set contrast
-s INT set saturation
-h INT set hue
-g INT set gain
-n INT set channel
-m INT set video mode
-l INT set mirror mode
-u INT set rotate mode
-v  V4L interface
-f  1394 interface
-p  show this help message
```

A sample configuration file reading is also provided:

```
#==================================#
# *  Default configuration file  *  #
#                                   #
#  Author: Marek Wnuk               #
#  Creation date: 03.03.2009        #
#                                   #
#==================================#

# FacET library parameters
# ========================

# Features detection parameters
EYEBROWS_RATIO 25
LIPS_RATIO 25
FACE_SCALING 4

# Paths to Haar classifier cascades
FACE_PATH /opt/opencv/share/opencv/haarcascades/haarcascade_frontalface_alt.xml
R_EYE_PATH cascades/eyes/thing_shan_eye_cascade.xml
L_EYE_PATH cascades/eyes/thing_shan_eye_cascade.xml
MOUTH_PATH cascades/mouth/Mouth.xml
NOSE_PATH cascades/nose/Nariz.xml

# Image capture parameters
# ========================

BRIGHTNESS 50         # Brightness, contrast, saturation and hue can be
CONTRAST 13           # initialised with any value from [0,100] range
SATURATION 13
HUE 0
GAIN 30
IMAGE_WIDTH 640
IMAGE_HEIGHT 480
FLIP 0       # 0 => flip= FALSE, otherwise flip=TRUE
ROTATE 0              # 0 => rotate= FALSE, otherwise rotate=TRUE

# Application parameters
# ======================

# Application options
TIMING  0             # 0 => timing on, otherwise timing off
IMGREC  1             # 0 => imgrec= FALSE, otherwise imgrec=TRUE
PARREC  1             # 0 => parrec= FALSE, otherwise parrec=TRUE
SHOW    1             # 0 => show= FALSE, otherwise show=TRUE
VERBOSE 1             # verbosity level (0 => no stderr output)
```

```
# Output path for the results of processing
IMAGE_PATH output/
```

The first section, "FacET library parameters" is read by readSettings(), which sets features detection parameters and paths to Haar classifier cascades. Two detection parameters (eyebrows proportions and lips proportions) can be used to trim the features detection methods in FacET library. The preset values are used by the histogram-based threshold calculation procedure for eyebrows and lips detection algorithms.

"Image capture parameters" and "Application parameters" can be also defined, up to the user needs. The user should provide relevant function for his own parameters reading. In the demo, the functions are defined in "demo/src/capture_mw.cpp": readSettings(string file_name), and in "demo/src/main.cpp": readOptions(string file_name) .

The demo application can output the results in several, user-selectable forms. Application options in config file are used for switching them off and on.

If PARREC is set, a new parameter file "1_fea.prm", "2_fea.prm", ... "n_fea.prm", is created for each consecutive run of the detection procedure. If IMGREC is set, output images/videos are saved as "n_img.png" and "n_vid.avi" respectively. Setting the location of the output files is possible with IMAGE_PATH configuration parameter. If SHOW is set, the output image from the detectFeat() library function is displayed in "Result" window (using highgui cvShowImage()).

A sample parameters registration file follows:

```
1; 180 72 2.02136 144 21 45 26 146 22 43 26 229 90 70 0 26.1084 0
2; 180 72 0.674037 141 22 46 24 142 21 45 25 233 86 76 2 24.1379 0
3; 184 72 1.38035 147 18 49 26 141 21 43 27 236 86 73 0 24.1379 0
4; 184 68 0.65106 145 19 42 25 146 17 42 26 241 86 79 2 24.1379 0
5; 184 72 1.99788 148 16 45 26 149 20 42 26 253 78 82 2 24.2574 0
6; 180 68 2.04541 143 21 46 25 146 21 41 24 237 82 79 0 25.2475 0
7; 184 72 1.99788 140 24 45 27 146 18 42 27 239 85 78 1 25.2475 0
8; 188 72 2.75911 144 18 44 26 143 20 43 25 215 65 57 1 23.7624 0
9; 180 68 1.99788 139 24 44 26 143 22 45 25 244 86 82 0 25.7426 0
```

The first record contains:

```
1;       - frame number in corresponding .avi file
180      - face ROI upper left corner X coordinate (pixels)
72       - face ROI upper left corner Y coordinate (pixels)
2.02136 - face declination angle (not verified, for future use)
144      - LEbBnd     left eyebrow bend angle (top)
21       - LEbDcl     left eyebrow declination angle (side)
45       - LEyOpn     distance between the right eyelids (relative to the eyeball subregion)
26       - LEbHgt     distance between left pupil and eyebrow top (relative to the eye subregion)
146      - REbBnd     right eyebrow bend angle (top)
22       - REbDcl     right eyebrow declination angle (side)
43       - REyOpn     distance between the right eyelids (relative to the eyeball subregion)
26       - REbHgt     distance between right pupil and eyebrow top (relative to the eye subregion)
229      - LiAspt     aspect ratio of the lips bounding box (percents)
90       - LLiCnr     Y position of the left corner of the lips (relative to the lips bounding box)
70       - RLiCnr     Y position of the right corner of the lips (relative to the lips bounding box)
0        - Wrnkls     number of horizontal wrinkles in the center of the forehead
26.1084 - Nstrls     nostrils baseline width (relative to the face width)
0        - TeethA     area of the visible teeth (relative to the lips bounding box)
```

In case of more than one face detected: face position, declination, and 14 parameters are appended to the same line (frame number).

Chapter 2

Class Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

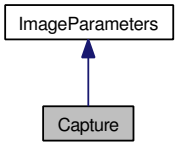
Chapter 5

Class Documentation

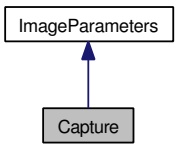
5.1 Capture Class Reference

`#include <capture_mw.h>`

Inheritance diagram for Capture:



Collaboration diagram for Capture:



Public Member Functions

- bool readSettings (std::string file_name="default.cfg")
- void setImageParameters (CvCapture *capture, Capture *cam)
- void camSetup (CvCapture *capture, int cam_if, int device, int channel, int norm, int quiet)

5.1.1 Detailed Description

Includes methods for camera image capture setup.

Definition at line 359 of file capture_mw.h.

5.1.2 Member Function Documentation

5.1.2.1 bool Capture::readSettings (std::string *file_name* = "default.cfg")

Read the settings from config file

Parameters:

file_name - settings file name

Return values:

true - succesful reading of the config file

false - otherwise

5.1.2.2 void Capture::setImageParameters (CvCapture * *capture*, Capture * *cam*)

Modify camera image parameters

Parameters:

capture - camera descriptor

ipar - camera settings

Definition at line 248 of file capture_mw.cpp.

Referenced by detectFromStream().

5.1.2.3 void Capture::camSetup (CvCapture * *capture*, int *cam_if*, int *device*, int *channel*, int *norm*, int *quiet*)

Setup and check camera options

Parameters:

file_name - settings file name

cam_if - camera interface (0-auto, 2-v4l, 3-1394)

device - device number

channel - channel number (if applicable)

norm - video standard (0-AUTO, 1-NTSC, 2-PAL, 3-SECAM)

quiet - 1 - be quiet, 0 - be verbose

Definition at line 131 of file capture_mw.cpp.

References ImageParameters::hue.

Referenced by detectFromStream().

The documentation for this class was generated from the following files:

- demo/inc/capture_mw.h
- demo/src/capture_mw.cpp

5.2 DetectionOptions Class Reference

`#include <options.h>`

Public Member Functions

- DetectionOptions ()
- DetectionOptions (bool tf, bool ir, bool pr, bool sh, int v)
- void setOptions (bool tf, bool ir, bool pr, bool sh, int v)
- void setTiming (int val)
- bool readTiming ()
- void setImgrec (int val)
- bool readImgrec ()
- void setParrec (int val)
- bool readParrec ()
- void setShow (int val)
- bool readShow ()
- void setVerbose (int val)
- int readVerbose () const
- void setOutputImagePath (std::string file_name)
- std::string readOutputImagePath () const

Protected Attributes

- bool timing
- bool imgrec
- bool parrec
- bool show
- int verbose
- std::string output_image_path

5.2.1 Detailed Description

Defines the options for face features detection procedures (timing, recording, verbosity). Provides methods for options reading and setting.

Definition at line 42 of file options.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 DetectionOptions::DetectionOptions () `[inline]`

Parameterless constructor

Sets the default values of the components

Definition at line 70 of file options.h.

References output_image_path.

5.2.2.2 DetectionOptions::DetectionOptions (bool *tf*, bool *ir*, bool *pr*, bool *sh*, int *v*) [inline]

Constructor

Sets the provided values of the components

Parameters:

tf - procedures timing flag

ir - image recording flag

pr - parameters recording flag

sh - output display flag

v - verbosity value

Definition at line 85 of file options.h.

5.2.3 Member Function Documentation

5.2.3.1 void DetectionOptions::setOptions (bool *tf*, bool *ir*, bool *pr*, bool *sh*, int *v*) [inline]

Sets the detection options

Parameters:

tf - new procedures timing flag

ir - new image recording flag

pr - new parameters recording flag

sh - new output display flag

v - new verbosity value

Definition at line 96 of file options.h.

References imgrec, parrec, show, timing, and verbose.

5.2.3.2 void DetectionOptions::setTiming (int *val*) [inline]

Sets the value of timing flag

Parameters:

val - if 0 then timing is set to FALSE, otherwise timing is set to TRUE

Definition at line 106 of file options.h.

References timing.

Referenced by readOptions().

5.2.3.3 bool DetectionOptions::readTiming () [inline]

Reads the value of timing flag

Returns:

value of timing flag

Definition at line 112 of file options.h.

References timing.

Referenced by detectFromImage(), and detectFromStream().

5.2.3.4 void DetectionOptions::setImgrec (int *val*) [inline]

Sets the value of imgrec flag

Parameters:

val - if 0 then imgrec is set to FALSE, otherwise imgrec is set to TRUE

Definition at line 120 of file options.h.

References imgrec.

Referenced by readOptions().

5.2.3.5 bool DetectionOptions::readImgrec () [inline]

Reads the value of imgrec flag

Returns:

value of imgrec flag

Definition at line 126 of file options.h.

References imgrec.

Referenced by detectFromImage(), and detectFromStream().

5.2.3.6 void DetectionOptions::setParrec (int *val*) [inline]

Sets the value of parrec flag

Parameters:

val - if 0 then parrec is set to FALSE, otherwise parrec is set to TRUE

Definition at line 134 of file options.h.

References parrec.

Referenced by readOptions().

5.2.3.7 bool DetectionOptions::readParrec () [inline]

Reads the value of parrec flag

Returns:

value of parrec flag

Definition at line 140 of file options.h.

References parrec.

Referenced by detectFromImage(), and detectFromStream().

5.2.3.8 void DetectionOptions::setShow (int *val*) [inline]

Sets the value of show flag

Parameters:

val - if 0 then show is set to FALSE, otherwise show is set to TRUE

Definition at line 148 of file options.h.

References show.

Referenced by readOptions().

5.2.3.9 bool DetectionOptions::readShow () [inline]

Reads the value of show flag

Returns:

value of show flag

Definition at line 154 of file options.h.

References show.

Referenced by detectFromImage(), and detectFromStream().

5.2.3.10 void DetectionOptions::setVerbose (int *val*) [inline]

Sets the verbosity value

Parameters:

val - new verbosity value

Definition at line 160 of file options.h.

References verbose.

Referenced by readOptions().

5.2.3.11 int DetectionOptions::readVerbose () const [inline]

Reads the verbosity value

Returns:

image verbosity value

Definition at line 166 of file options.h.

References verbose.

Referenced by detectFromImage(), and detectFromStream().

5.2.3.12 void DetectionOptions::setOutputImagePath (std::string *file_name*) [inline]

Sets the output image path

Parameters:

file_name - output image path

Definition at line 173 of file options.h.

References output_image_path.

5.2.3.13 std::string DetectionOptions::readOutputImagePath () const [inline]

Reads the output image path

Returns:

file_name - output image path

Definition at line 181 of file options.h.

References output_image_path.

Referenced by showMenu().

5.2.4 Member Data Documentation

5.2.4.1 bool DetectionOptions::timing [protected]

Detection procedures timing

Definition at line 47 of file options.h.

Referenced by readTiming(), setOptions(), and setTiming().

5.2.4.2 bool DetectionOptions::imgrec [protected]

Image/video recording

Definition at line 50 of file options.h.

Referenced by readImgrec(), setImgrec(), and setOptions().

5.2.4.3 bool DetectionOptions::parrec [protected]

Parameters recording

Definition at line 53 of file options.h.

Referenced by readParrec(), setOptions(), and setParrec().

5.2.4.4 bool DetectionOptions::show [protected]

Image/video results display

Definition at line 56 of file options.h.

Referenced by readShow(), setOptions(), and setShow().

5.2.4.5 int DetectionOptions::verbose [protected]

Verbosity

Definition at line 59 of file options.h.

Referenced by readVerbose(), setOptions(), and setVerbose().

5.2.4.6 std::string DetectionOptions::output_image_path [protected]

Output image path name

Definition at line 62 of file options.h.

Referenced by DetectionOptions(), readOutputImagePath(), and setOutputImagePath().

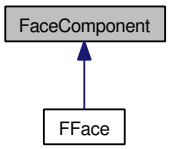
The documentation for this class was generated from the following file:

- demo/inc/options.h

5.3 FaceComponent Class Reference

`#include <facet.h>`

Inheritance diagram for FaceComponent:



Public Member Functions

- FaceComponent ()
- FaceComponent (CvPoint pt1, CvPoint pt2)
- FaceComponent (CvPoint pt1, CvPoint pt2, CvPoint pt3)
- FaceComponent (CvPoint pt1, CvPoint pt2, CvPoint pt3, CvPoint pt4)
- FaceComponent operator- (FaceComponent el)
- FaceComponent operator+ (FaceComponent el)
- void operator+= (FaceComponent el)
- void clear ()

Public Attributes

- CvPoint p1
- CvPoint p2
- CvPoint p3
- CvPoint p4

5.3.1 Detailed Description

Describes the corners of face subregion rectangle.

Definition at line 243 of file facet.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 FaceComponent::FaceComponent () `[inline]`

Parameterless constructor

Sets the default values of the components

Definition at line 258 of file facet.h.

Referenced by eyeDetection(), operator+(), and operator-().

5.3.2.2 FaceComponent::FaceComponent (CvPoint *pt1*, CvPoint *pt2*) [inline]

Constructor

Parameters:

pt1 - first point of face component
pt2 - second point of face component

Definition at line 265 of file facet.h.

5.3.2.3 FaceComponent::FaceComponent (CvPoint *pt1*, CvPoint *pt2*, CvPoint *pt3*) [inline]

Constructor

Parameters:

pt1 - first point of face component
pt2 - second point of face component
pt3 - third point of face component

Definition at line 274 of file facet.h.

5.3.2.4 FaceComponent::FaceComponent (CvPoint *pt1*, CvPoint *pt2*, CvPoint *pt3*, CvPoint *pt4*) [inline]

Constructor

Parameters:

pt1 - first point of face component
pt2 - second point of face component
pt3 - third point of face component
pt4 - fourth point of face component

Definition at line 284 of file facet.h.

5.3.3 Member Function Documentation

5.3.3.1 FaceComponent FaceComponent::operator- (FaceComponent *el*) [inline]

Overloading operator of subtraction of two objects of FaceComponent class

Parameters:

el - input component

Returns:

difference of input component and the object

Definition at line 293 of file facet.h.

References FaceComponent(), p1, p2, p3, and p4.

5.3.3.2 FaceComponent FaceComponent::operator+ (FaceComponent *el*) [inline]

Overloading operator of addition of two objects of FaceComponent class

Parameters:

el - input component

Returns:

sum of input component and the object

Definition at line 306 of file facet.h.

References FaceComponent(), p1, p2, p3, and p4.

5.3.3.3 void FaceComponent::operator+= (FaceComponent *el*) [inline]

Overloading operator of accumulation of two objects of FaceComponent class

Parameters:

el - input component

Definition at line 317 of file facet.h.

5.3.3.4 void FaceComponent::clear () [inline]

Resets the fields values

Definition at line 322 of file facet.h.

References p1, p2, p3, and p4.

Referenced by FFace::clearElements().

5.3.4 Member Data Documentation

5.3.4.1 CvPoint FaceComponent::p1

A point of face component

Definition at line 246 of file facet.h.

Referenced by clear(), detectFacialRegions(), Facet::detectFeat(), findFaceRegions(), operator+(), and operator-().

5.3.4.2 CvPoint FaceComponent::p2

A point of face component

Definition at line 248 of file facet.h.

Referenced by clear(), detectFacialRegions(), Facet::detectFeat(), findFaceRegions(), operator+(), and operator-().

5.3.4.3 CvPoint FaceComponent::p3

A point of face component

Definition at line 250 of file facet.h.

Referenced by clear(), operator+(), and operator-().

5.3.4.4 CvPoint FaceComponent::p4

A point of face component

Definition at line 252 of file facet.h.

Referenced by clear(), operator+(), and operator-().

The documentation for this class was generated from the following file:

- inc/facet.h

5.4 facepar_t Struct Reference

`#include <facet.h>`

5.4.1 Detailed Description

Structure of face features for saving values after detection:

roix – face ROI upper left corner X coordinate (pixels)

roiy – face ROI upper left corner Y coordinate (pixels)

angle – face declination angle (not verified, for future use)

LEbBnd – left eyebrow bend angle (top)

LEbDcl – left eyebrow declination angle (side)

LEyOpn – distance between the right eyelids (rel. eyeball subregion)

LEbHgt – distance between left pupil and eyebrow top (rel. eye subregion)

REbBnd – right eyebrow bend angle (top)

REbDcl – right eyebrow declination angle (side)

REyOpn – distance between the right eyelids (rel. eyeball subregion)

REbHgt – distance between right pupil and eyebrow top (rel. eye subregion)

LiAspt – aspect ratio of the lips bounding box (percents)

LLiCnr – Y position of the left corner of the lips (rel. lips bounding box)

RLiCnr – Y position of the right corner of the lips (rel. lips bounding box)

Wrnkls – number of horizontal wrinkles in the center of the forehead

Nstrls – nostrils baseline width (rel. face width)

TeethA – area of the visible teeth (rel. lips bounding box)

Definition at line 794 of file facet.h.

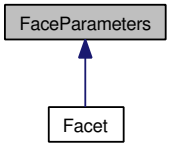
The documentation for this struct was generated from the following file:

- inc/facet.h

5.5 FaceParameters Class Reference

`#include <facet.h>`

Inheritance diagram for FaceParameters:



Public Member Functions

- FaceParameters ()
- void setEyebrowsRatio (int val)
- void setLipsRatio (int val)
- int readEyebrowsRatio () const
- int readLipsRatio () const

Protected Attributes

- int eyebrow_proportions
  Parameter defining eyebrows proportions.
- int lips_proportions
  Parameter defining lips proportions.

5.5.1 Detailed Description

Defines the face parameters (eyebrows and lips proportions and face region height. Provides methods for parameters reading and setting.

Definition at line 73 of file facet.h.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 FaceParameters::FaceParameters () `[inline]`

Parameterless constructor

Sets the default values of the components

Definition at line 97 of file facet.h.

5.5.3 Member Function Documentation

5.5.3.1 void FaceParameters::setEyebrowsRatio (int *val*) `[inline]`

Sets the eyebrows proportions

Parameters:

val - eyebrows proportions value

Definition at line 104 of file facet.h.

References eyebrow_proportions.

5.5.3.2 void FaceParameters::setLipsRatio (int *val*) [inline]

Sets the lips proportions

Parameters:

val - lips proportions value

Definition at line 110 of file facet.h.

References lips_proportions.

5.5.3.3 int FaceParameters::readEyebrowsRatio () const [inline]

Reads the eyebrow proportions

Returns:

eyebrow proportions value

Definition at line 116 of file facet.h.

References eyebrow_proportions.

5.5.3.4 int FaceParameters::readLipsRatio () const [inline]

Reads the lips proportions

Returns:

lips proportions value

Definition at line 122 of file facet.h.

References lips_proportions.

5.5.4 Member Data Documentation

5.5.4.1 int FaceParameters::eyebrow_proportions [protected]

Parameter defining eyebrows proportions.

Used in procedure for histogram-based finding of the threshold in eyebrows detection

Definition at line 82 of file facet.h.

Referenced by readEyebrowsRatio(), and setEyebrowsRatio().

5.5.4.2 int FaceParameters::lips_proportions [protected]

Parameter defining lips proportions.

Used in procedure for histogram-based finding of the threshold in lips detection

Definition at line 89 of file facet.h.

Referenced by readLipsRatio(), and setLipsRatio().

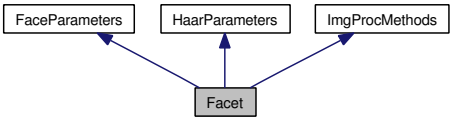
The documentation for this class was generated from the following file:

- inc/facet.h

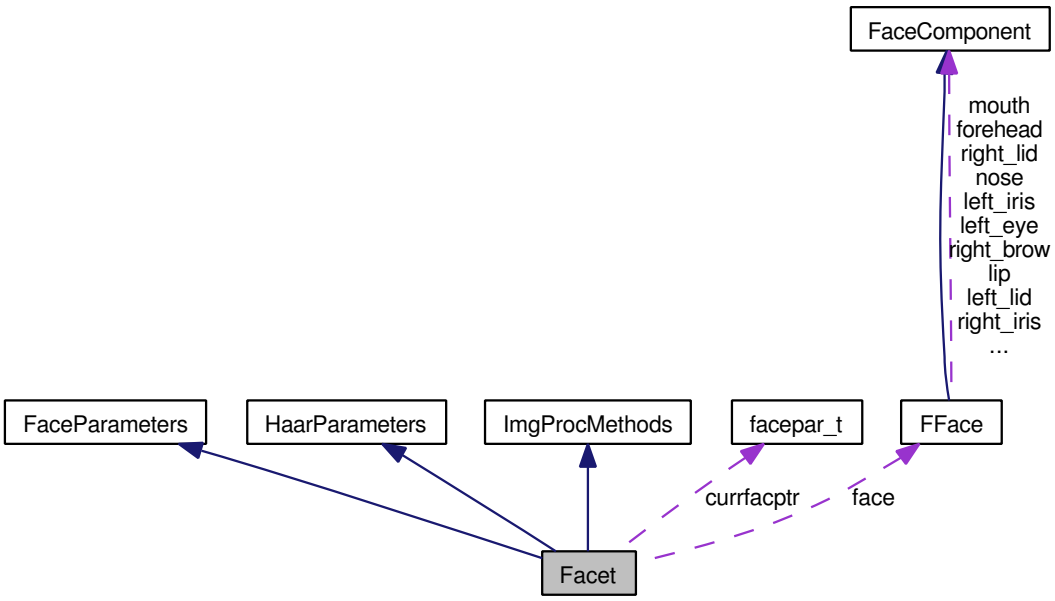
5.6 Facet Class Reference

`#include <facet.h>`

Inheritance diagram for Facet:



Collaboration diagram for Facet:



Public Member Functions

- bool readSettings (std::string file_name="default.cfg")
- void cleanFacesList ()
- void detectFeat (IplImage *src, IplImage *dst)

Public Attributes

- FFace face
- std::list< facepar_t > facesList
- bool mouthHaar
- bool noseHaar
- bool eyeballsHaar

Static Public Attributes

- static facepar_t * currfacptr = NULL

5.6.1 Detailed Description

Includes methods for face features detection and parameterisation.

Definition at line 818 of file facet.h.

5.6.2 Member Function Documentation

5.6.2.1 bool Facet::readSettings (std::string *file_name* = "default.cfg")

Read the settings from config file

Parameters:

file_name - settings file name

Return values:

true - succesful reading of the config file
false - otherwise

5.6.2.2 void Facet::cleanFacesList ()

Remove all the elements of the faces list (facesList)

Definition at line 1539 of file facet.cpp.

References facesList.

Referenced by detectFromImage(), and detectFromStream().

5.6.2.3 void Facet::detectFeat (IplImage * *src*, IplImage * *dst*)

Detect the face features in the image

Parameters:

src - input image
dst - output image

Definition at line 185 of file facet.cpp.

References currfacptr, eyeballsHaar, face, HaarParameters::face_detection_scale, facesList, findFaceRegions(), FFace::forehead, facepar_t::LEbHgt, FFace::left_brow, FFace::left_eye, FFace::left_iris, FFace::left_lid, facepar_t::LEyOpn, FFace::lip, FFace::mouth, mouthHaar, FFace::nose, noseHaar, FaceComponent::p1, FaceComponent::p2, facepar_t::REbHgt, facepar_t::REyOpn, FFace::right_brow, FFace::right_eye, FFace::right_iris, and FFace::right_lid.

Referenced by detectFromImage(), and detectFromStream().

5.6.3 Member Data Documentation

5.6.3.1 FFace Facet::face

Face components data

Definition at line 826 of file facet.h.

Referenced by detectFeat(), detectFromImage(), and detectFromStream().

5.6.3.2 facepar_t * Facet::currfacptr = NULL [static]

Pointer to structure with the detected values

Definition at line 829 of file facet.h.

Referenced by detectFeat().

5.6.3.3 std:: list<facepar_t> Facet::facesList

List of the detected faces and parameters

Definition at line 832 of file facet.h.

Referenced by cleanFacesList(), detectFeat(), and writeDataToFile().

5.6.3.4 bool Facet::mouthHaar

Haar method for mouth detection flag

Definition at line 835 of file facet.h.

Referenced by detectFeat().

5.6.3.5 bool Facet::noseHaar

Haar method for nose detection flag

Definition at line 838 of file facet.h.

Referenced by detectFeat().

5.6.3.6 bool Facet::eyeballsHaar

Haar method for eyeballs detection flag

Definition at line 841 of file facet.h.

Referenced by detectFeat().

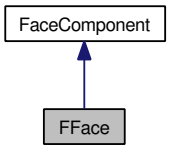
The documentation for this class was generated from the following files:

- inc/facet.h
- src/facet.cpp

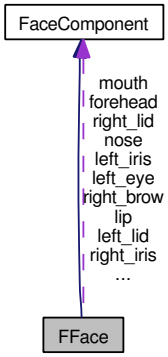
5.7 FFace Class Reference

`#include <facet.h>`

Inheritance diagram for FFace:



Collaboration diagram for FFace:



Public Member Functions

- void clearElements ()

Public Attributes

- FaceComponent mouth
- FaceComponent left_eye
- FaceComponent right_eye
- FaceComponent left_iris
- FaceComponent right_iris
- FaceComponent left_lid
- FaceComponent right_lid
- FaceComponent left_brow
- FaceComponent right_brow
- FaceComponent lip
- FaceComponent forehead
- FaceComponent nose
- int width

5.7.1 Detailed Description

Describes the face components. Contains attributes of selected components.

Definition at line 330 of file facet.h.

5.7.2 Member Function Documentation

5.7.2.1 void FFace::clearElements ()

Resets the fields values

Definition at line 49 of file face_regions.cpp.

References FaceComponent::clear(), forehead, left_brow, left_eye, left_iris, left_lid, lip, mouth, nose, right_brow, right_eye, right_iris, and right_lid.

Referenced by detectFromImage(), and detectFromStream().

5.7.3 Member Data Documentation

5.7.3.1 FaceComponent FFace::mouth

Component describing mouth

Definition at line 334 of file facet.h.

Referenced by clearElements(), Facet::detectFeat(), and findFaceRegions().

5.7.3.2 FaceComponent FFace::left_eye

Component describing left eye

Definition at line 336 of file facet.h.

Referenced by clearElements(), Facet::detectFeat(), and findFaceRegions().

5.7.3.3 FaceComponent FFace::right_eye

Component describing right eye

Definition at line 338 of file facet.h.

Referenced by clearElements(), Facet::detectFeat(), and findFaceRegions().

5.7.3.4 FaceComponent FFace::left_iris

Component describing left iris

Definition at line 340 of file facet.h.

Referenced by clearElements(), Facet::detectFeat(), and findFaceRegions().

5.7.3.5 FaceComponent FFace::right_iris

Component describing right iris

Definition at line 342 of file facet.h.

Referenced by clearElements(), Facet::detectFeat(), and findFaceRegions().

5.7.3.6 FaceComponent FFace::left_lid

Component describing left eyelid

Definition at line 344 of file facet.h.

Referenced by clearElements(), and Facet::detectFeat().

5.7.3.7 FaceComponent FFace::right_lid

Component describing right eyelid

Definition at line 346 of file facet.h.

Referenced by clearElements(), and Facet::detectFeat().

5.7.3.8 FaceComponent FFace::left_brow

Component describing left eyebrow

Definition at line 348 of file facet.h.

Referenced by clearElements(), and Facet::detectFeat().

5.7.3.9 FaceComponent FFace::right_brow

Component describing right eyebrow

Definition at line 350 of file facet.h.

Referenced by clearElements(), and Facet::detectFeat().

5.7.3.10 FaceComponent FFace::lip

Component describing lips (p1,p2 - mouth rectangle, p3,p4 - lips corners

Definition at line 352 of file facet.h.

Referenced by clearElements(), and Facet::detectFeat().

5.7.3.11 FaceComponent FFace::forehead

Component describing forehead

Definition at line 354 of file facet.h.

Referenced by clearElements(), and Facet::detectFeat().

5.7.3.12 FaceComponent FFace::nose

Component describing nose

Definition at line 356 of file facet.h.

Referenced by clearElements(), and Facet::detectFeat().

5.7.3.13 int FFace::width

Face width

Definition at line 358 of file facet.h.

Referenced by findFaceRegions().

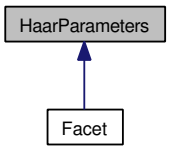
The documentation for this class was generated from the following files:

- inc/facet.h
- src/face_regions.cpp

5.8 HaarParameters Class Reference

`#include <facet.h>`

Inheritance diagram for HaarParameters:



Public Member Functions

- HaarParameters ()
- void setFaceScale (double val)
- double readFaceScale () const
- void readFaceCascade (std::string face)
- void readMouthCascade (std::string mouth)
- void readNoseCascade (std::string nose)
- void readLeftEyeCascade (std::string eye_left)
- void readRightEyeCascade (std::string eye_right)

Protected Attributes

- double face_detection_scale
- CvHaarClassifierCascade * cascade_face
- CvHaarClassifierCascade * cascade_mouth
- CvHaarClassifierCascade * cascade_nose
- CvHaarClassifierCascade * cascade_leftEye
- CvHaarClassifierCascade * cascade_rightEye

Static Protected Attributes

- static CvMemStorage * face_storage = cvCreateMemStorage(0)

5.8.1 Detailed Description

Defines parameters for Haar classifier (image scaling for face detection, Haar cascades read from corresponding files for face, mouth, nose and eyes). Provides methods for parameters reading and setting.

Definition at line 132 of file facet.h.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 HaarParameters::HaarParameters () `[inline]`

Parameterless constructor

Sets the default values of the components

Definition at line 166 of file facet.h.

References cascade_face, cascade_leftEye, cascade_mouth, cascade_nose, and cascade_rightEye.

5.8.3 Member Function Documentation

5.8.3.1 void HaarParameters::setFaceScale (double *val*) [inline]

Sets image scaling for face detection with Haar classifier

Parameters:

val - new image scaling value

Definition at line 175 of file facet.h.

References face_detection_scale.

5.8.3.2 double HaarParameters::readFaceScale () const [inline]

Reads image scaling for face detection with Haar classifier

Returns:

image scaling value

Definition at line 183 of file facet.h.

References face_detection_scale.

5.8.3.3 void HaarParameters::readFaceCascade (std::string *face*) [inline]

Reads Haar cascade for face

Definition at line 188 of file facet.h.

References cascade_face.

5.8.3.4 void HaarParameters::readMouthCascade (std::string *mouth*) [inline]

Reads Haar cascade for mouth

Definition at line 195 of file facet.h.

References cascade_mouth.

5.8.3.5 void HaarParameters::readNoseCascade (std::string *nose*) [inline]

Reads Haar cascade for nose

Definition at line 202 of file facet.h.

References cascade_nose.

5.8.3.6 void HaarParameters::readLeftEyeCascade (std::string *eye_left*) [inline]

Reads Haar cascade for left eye

Definition at line 209 of file facet.h.

References cascade_leftEye.

5.8.3.7 void HaarParameters::readRightEyeCascade (std::string *eye_right*) [inline]

Reads Haar cascade for right eye

Definition at line 216 of file facet.h.

References cascade_rightEye.

5.8.4 Member Data Documentation

5.8.4.1 double HaarParameters::face_detection_scale [protected]

The value of image downscaling ratio for face detection with Haar classifier.

Definition at line 139 of file facet.h.

Referenced by Facet::detectFeat(), readFaceScale(), and setFaceScale().

5.8.4.2 CvMemStorage * HaarParameters::face_storage = cvCreateMemStorage(0) [static, protected]

Memory storage for face detection with Haar classifier.

Definition at line 143 of file facet.h.

5.8.4.3 CvHaarClassifierCascade* HaarParameters::cascade_face [protected]

Haar cascade for face

Definition at line 146 of file facet.h.

Referenced by HaarParameters(), and readFaceCascade().

5.8.4.4 CvHaarClassifierCascade* HaarParameters::cascade_mouth [protected]

Haar cascade for mouth

Definition at line 149 of file facet.h.

Referenced by HaarParameters(), and readMouthCascade().

5.8.4.5 CvHaarClassifierCascade* HaarParameters::cascade_nose [protected]

Haar cascade for nose

Definition at line 152 of file facet.h.

Referenced by HaarParameters(), and readNoseCascade().

5.8.4.6 CvHaarClassifierCascade* HaarParameters::cascade_leftEye [protected]

Haar cascade for left eye

Definition at line 155 of file facet.h.

Referenced by HaarParameters(), and readLeftEyeCascade().

5.8.4.7 CvHaarClassifierCascade* HaarParameters::cascade_rightEye [protected]

Haar cascade for right eye

Definition at line 158 of file facet.h.

Referenced by HaarParameters(), and readRightEyeCascade().

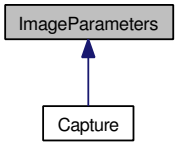
The documentation for this class was generated from the following files:

- inc/facet.h
- src/facet.cpp

5.9 ImageParameters Class Reference

`#include <captparam.h>`

Inheritance diagram for ImageParameters:



Public Member Functions

- ImageParameters ()
- ImageParameters (int wdth, int hght, int b, int c, int s, int h, int g, bool f, bool r)
- void setParameters (int wdth, int hght, int b, int c, int s, int h, int g, bool f, bool r)
- void setWidth (int val)
- int readWidth () const
- void setHeight (int val)
- int readHeight () const
- void setBrightness (int val)
- int readBrightness () const
- void setContrast (int val)
- int readContrast () const
- void setSaturation (int val)
- int readSaturation () const
- void setHue (int val)
- int readHue () const
- void setGain (int val)
- int readGain () const
- void setFlip (int val)
- bool readFlip ()
- void setRotate (int val)
- bool readRotate ()
- ImageParameters ()
- ImageParameters (int wdth, int hght, int b, int c, int s, int h, int g, bool f, bool r)
- void setParameters (int wdth, int hght, int b, int c, int s, int h, int g, int f, int r)
- void setWidth (int val)
- int readWidth () const
- void setHeight (int val)
- int readHeight () const
- void setBrightness (int val)
- int readBrightness () const
- void setContrast (int val)
- int readContrast () const
- void setSaturation (int val)
- int readSaturation () const
- void setHue (int val)
- int readHue () const

- void setGain (int val)
- int readGain () const
- void setFlip (int val)
- int readFlip ()
- void setRotate (int val)
- int readRotate ()

Protected Attributes

- int width
- int height
- int brightness
- int contrast
- int saturation
- int hue
- int gain
- bool flip
- bool rotate
- int flip
- int rotate

5.9.1 Detailed Description

Defines the image parameters (brightness, contrast, saturation and hue). Provides methods for parameters reading and setting.

Definition at line 48 of file captparam.h.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 ImageParameters::ImageParameters () [inline]

Parameterless constructor

Sets the default values of the components

Definition at line 93 of file captparam.h.

5.9.2.2 ImageParameters::ImageParameters (int *wdth*, int *hght*, int *b*, int *c*, int *s*, int *h*, int *g*, bool *f*, bool *r*) [inline]

Constructor

Sets the provided values of the components

Parameters:

wdth - image width value

hght - image height value

b - brightness value

c - contrast value

s - saturation value

h - hue value

g - gain value

f - flip value

r - rotate value

Definition at line 111 of file captparam.h.

5.9.2.3 ImageParameters::ImageParameters () [inline]

Parameterless constructor

Sets the default values of the components

Definition at line 201 of file capture_mw.h.

5.9.2.4 ImageParameters::ImageParameters (int *wdth*, int *hght*, int *b*, int *c*, int *s*, int *h*, int *g*, bool *f*, bool *r*) [inline]

Constructor

Sets the provided values of the components

Parameters:

wdth - image width value

hght - image height value

b - brightness value

c - contrast value

s - saturation value

h - hue value

g - gain value

f - flip value

r - rotate value

Definition at line 219 of file capture_mw.h.

5.9.3 Member Function Documentation

5.9.3.1 void ImageParameters::setParameters (int *wdth*, int *hght*, int *b*, int *c*, int *s*, int *h*, int *g*, bool *f*, bool *r*) [inline]

Sets the image parameters

Parameters:

wdth - new image width value

hght - new image height value

b - new brightness value

c - new contrast value

s - new saturation value

h - new hue value

g - new gain value

f - new flip value

r - new rotate value

Definition at line 128 of file captparam.h.

References brightness, contrast, flip, gain, height, hue, rotate, saturation, and width.

5.9.3.2 void ImageParameters::setWidth (int *val*) [inline]

Sets the image width value

Parameters:

val - new image width value

Definition at line 138 of file captparam.h.

References width.

5.9.3.3 int ImageParameters::readWidth () const [inline]

Reads the image width value

Returns:

image width value

Definition at line 144 of file captparam.h.

References width.

5.9.3.4 void ImageParameters::setHeight (int *val*) [inline]

Sets the image height value

Parameters:

val - new image height value

Definition at line 150 of file captparam.h.

References height.

5.9.3.5 int ImageParameters::readHeight () const [inline]

Reads the image height value

Returns:

image height value

Definition at line 156 of file captparam.h.

References height.

5.9.3.6 void ImageParameters::setBrightness (int *val*) [inline]

Sets the brightness value

Parameters:

val - new brightness value

Definition at line 162 of file captparam.h.

References brightness.

5.9.3.7 int ImageParameters::readBrightness () const [inline]

Reads the brightness value

Returns:

brightness value

Definition at line 168 of file captparam.h.

References brightness.

5.9.3.8 void ImageParameters::setContrast (int *val*) [inline]

Sets the contrast value

Parameters:

val - new contrast value

Definition at line 174 of file captparam.h.

References contrast.

5.9.3.9 int ImageParameters::readContrast () const [inline]

Reads the contrast value

Returns:

contrast value

Definition at line 180 of file captparam.h.

References contrast.

5.9.3.10 void ImageParameters::setSaturation (int *val*) [inline]

Sets the saturation value

Parameters:

val - new saturation value

Definition at line 186 of file captparam.h.

References saturation.

5.9.3.11 int ImageParameters::readSaturation () const [inline]

Reads the saturation value

Returns:

saturation value

Definition at line 192 of file captparam.h.

References saturation.

5.9.3.12 void ImageParameters::setHue (int *val*) [inline]

Sets the hue value

Parameters:

val - new hue value

Definition at line 198 of file captparam.h.

References hue.

5.9.3.13 int ImageParameters::readHue () const [inline]

Reads the hue value

Returns:

hue value

Definition at line 204 of file captparam.h.

References hue.

5.9.3.14 void ImageParameters::setGain (int *val*) [inline]

Sets the hue value

Parameters:

val - new hue value

Definition at line 210 of file captparam.h.

References gain.

5.9.3.15 int ImageParameters::readGain () const [inline]

Reads the hue value

Returns:

hue value

Definition at line 216 of file captparam.h.

References gain.

5.9.3.16 void ImageParameters::setFlip (int *val*) [inline]

Sets the value of flip flag

Parameters:

val - if 0 then flip is set to FALSE, otherwise flip is set to TRUE

Definition at line 223 of file captparam.h.

References flip.

5.9.3.17 bool ImageParameters::readFlip () [inline]

Reads the value of flip flag

Returns:

value of flip flag

Definition at line 229 of file captparam.h.

References flip.

5.9.3.18 void ImageParameters::setRotate (int *val*) [inline]

Sets the value of rotate flag

Parameters:

val - if 0 then rotate is set to FALSE, otherwise rotate is set to TRUE

Definition at line 237 of file captparam.h.

References rotate.

5.9.3.19 bool ImageParameters::readRotate () [inline]

Reads the value of rotate flag

Returns:

value of rotate flag

Definition at line 243 of file captparam.h.

References rotate.

5.9.3.20 void ImageParameters::setParameters (int *wdth*, int *hght*, int *b*, int *c*, int *s*, int *h*, int *g*, int *f*, int *r*) [inline]

Sets the image parameters

Parameters:

wdth - new image width value

hght - new image height value

b - new brightness value

c - new contrast value

s - new saturation value

h - new hue value

g - new gain value

f - new flip value

r - new rotate value

Definition at line 236 of file capture_mw.h.

References brightness, contrast, flip, gain, height, hue, rotate, saturation, and width.

5.9.3.21 void ImageParameters::setWidth (int *val*) [inline]

Sets the image width value

Parameters:

val - new image width value

Definition at line 246 of file capture_mw.h.

References width.

5.9.3.22 int ImageParameters::readWidth () const [inline]

Reads the image width value

Returns:

image width value

Definition at line 252 of file capture_mw.h.

References width.

5.9.3.23 void ImageParameters::setHeight (int *val*) [inline]

Sets the image height value

Parameters:

val - new image height value

Definition at line 258 of file capture_mw.h.

References height.

5.9.3.24 int ImageParameters::readHeight () const [inline]

Reads the image height value

Returns:

image height value

Definition at line 264 of file capture_mw.h.

References height.

5.9.3.25 void ImageParameters::setBrightness (int *val*) [inline]

Sets the brightness value

Parameters:

val - new brightness value

Definition at line 270 of file capture_mw.h.

References brightness.

5.9.3.26 int ImageParameters::readBrightness () const [inline]

Reads the brightness value

Returns:

brightness value

Definition at line 276 of file capture_mw.h.

References brightness.

5.9.3.27 void ImageParameters::setContrast (int *val*) [inline]

Sets the contrast value

Parameters:

val - new contrast value

Definition at line 282 of file capture_mw.h.

References contrast.

5.9.3.28 int ImageParameters::readContrast () const [inline]

Reads the contrast value

Returns:

contrast value

Definition at line 288 of file capture_mw.h.

References contrast.

5.9.3.29 void ImageParameters::setSaturation (int *val*) [inline]

Sets the saturation value

Parameters:

val - new saturation value

Definition at line 294 of file capture_mw.h.

References saturation.

5.9.3.30 int ImageParameters::readSaturation () const [inline]

Reads the saturation value

Returns:

saturation value

Definition at line 300 of file capture_mw.h.

References saturation.

5.9.3.31 void ImageParameters::setHue (int *val*) [inline]

Sets the hue value

Parameters:

val - new hue value

Definition at line 306 of file capture_mw.h.

References hue.

5.9.3.32 int ImageParameters::readHue () const [inline]

Reads the hue value

Returns:

hue value

Definition at line 312 of file capture_mw.h.

References hue.

5.9.3.33 void ImageParameters::setGain (int *val*) [inline]

Sets the hue value

Parameters:

val - new hue value

Definition at line 318 of file capture_mw.h.

References gain.

5.9.3.34 int ImageParameters::readGain () const [inline]

Reads the hue value

Returns:

hue value

Definition at line 324 of file capture_mw.h.

References gain.

5.9.3.35 void ImageParameters::setFlip (int *val*) [inline]

Sets the value of flip flag

Parameters:

val - if 0 then flip is set to FALSE, otherwise flip is set to TRUE

Definition at line 331 of file capture_mw.h.

References flip.

5.9.3.36 int ImageParameters::readFlip () [inline]

Reads the value of flip flag

Returns:

value of flip flag

Definition at line 337 of file capture_mw.h.

References flip.

5.9.3.37 void ImageParameters::setRotate (int *val*) [inline]

Sets the value of rotate flag

Parameters:

val - if 0 then rotate is set to FALSE, otherwise rotate is set to TRUE

Definition at line 345 of file capture_mw.h.

References rotate.

5.9.3.38 int ImageParameters::readRotate () [inline]

Reads the value of rotate flag

Returns:

value of rotate flag

Definition at line 351 of file capture_mw.h.

References rotate.

5.9.4 Member Data Documentation

5.9.4.1 int ImageParameters::width [protected]

Image width

Definition at line 53 of file captparam.h.

Referenced by readWidth(), setParameters(), and setWidth().

5.9.4.2 int ImageParameters::height [protected]

Image height

Definition at line 56 of file captparam.h.

Referenced by readHeight(), setHeight(), and setParameters().

5.9.4.3 int ImageParameters::brightness [protected]

Image brightness

Definition at line 59 of file captparam.h.

Referenced by readBrightness(), setBrightness(), and setParameters().

5.9.4.4 int ImageParameters::contrast [protected]

Image contrast

Definition at line 62 of file captparam.h.

Referenced by readContrast(), setContrast(), and setParameters().

5.9.4.5 int ImageParameters::saturation [protected]

Image saturation

Definition at line 65 of file captparam.h.

Referenced by readSaturation(), setParameters(), and setSaturation().

5.9.4.6 int ImageParameters::hue [protected]

Image hue

Definition at line 68 of file captparam.h.

Referenced by Capture::camSetup(), readHue(), setHue(), and setParameters().

5.9.4.7 int ImageParameters::gain [protected]

Gain

Definition at line 71 of file captparam.h.

Referenced by readGain(), setGain(), and setParameters().

5.9.4.8 bool ImageParameters::flip [protected]

Horizontal image flipping

True - mirror-like image False - normal image

Definition at line 78 of file captparam.h.

Referenced by readFlip(), setFlip(), and setParameters().

5.9.4.9 bool ImageParameters::rotate [protected]

Image rotation by 180 deg.

True - seascape image False - normal image

Definition at line 85 of file captparam.h.

Referenced by readRotate(), setParameters(), and setRotate().

5.9.4.10 int ImageParameters::flip [protected]

Horizontal image flipping

True - mirror-like image False - normal image

Definition at line 186 of file capture_mw.h.

5.9.4.11 int ImageParameters::rotate [protected]

Image rotation by 180 deg.

True - seascape image False - normal image

Definition at line 193 of file capture_mw.h.

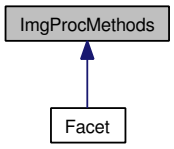
The documentation for this class was generated from the following files:

- demo/inc/captparam.h
- demo/inc/capture_mw.h

5.10 ImgProcMethods Class Reference

`#include <facet.h>`

Inheritance diagram for ImgProcMethods:



Public Member Functions

- int findConnectedRegions (IplImage *src, bool draw=false, IplImage *dst=0)
- void hysteresisThresholding (IplImage *src, IplImage *dst, int R, int L)
- bool checkPixelNeighbourhood (IplImage *src, int y, int x, int R)
- void createOutline (IplImage *src, IplImage *dst)
- void removeBoundaryBlobs (IplImage *src, IplImage *dst, std::bitset< 4 > flag)
- void removeSmallBlobs (IplImage *src, IplImage *dst, int iter)
- void findContours (IplImage *src, CvSeq *&contours)
- IplImage * showHistogram (const IplImage *src, bool display=false, CvSize size=cvSize(512, 200))
- void stretchHistogram (const IplImage *src, IplImage *dst)
- int findThresholdByHist (IplImage *src, int wsp)
- int doHoughTransform (IplImage *src, std::list< CvPoint > &linesList)
- void detectObjectHaar (IplImage *src, CvHaarClassifierCascade *cascade, FaceComponent &element, double scale=1, double scale_factor=1.2, int min_neighbors=5)

Static Protected Attributes

- static CvMemStorage * contours_storage = cvCreateMemStorage(0)
- static CvMemStorage * storage = cvCreateMemStorage(0)

5.10.1 Detailed Description

Implements selected image processing methods.

Definition at line 514 of file facet.h.

5.10.2 Member Function Documentation

5.10.2.1 int ImgProcMethods::findConnectedRegions (IplImage * *src*, bool *draw* = `false`, IplImage * *dst* = `0`)

Counts connected areas in input image

Parameters:

src - input image

draw - contours drawing flag

dst - output image for contours

Returns:

count of connected areas in input image

Definition at line 51 of file image_processing.cpp.

References contours_storage.

5.10.2.2 void ImgProcMethods::hysteresisThresholding (IplImage * *src*, IplImage * *dst*, int *R*, int *L*)

Hysteresis thresholding

Parameters:

src - input image

dst - output image

R - lower (radical) threshold

L - upper (liberal) threshold

Definition at line 263 of file image_processing.cpp.

References checkPixelNeighbourhood().

5.10.2.3 bool ImgProcMethods::checkPixelNeighbourhood (IplImage * *src*, int *y*, int *x*, int *R*)

Checks the neighbourhood of (x,y) for the presence of pixels darker than R threshold.

Parameters:

src - input image

y - Y coordinate of pixel

x - X coordinate of pixel

R - lower (radical) threshold

Return values:

true - if a pixel darker than R threshold was found in (x,y) neighbourhood

false - otherwise

Definition at line 252 of file image_processing.cpp.

Referenced by hysteresisThresholding().

5.10.2.4 void ImgProcMethods::createOutline (IplImage * *src*, IplImage * *dst*)

Creates silhouette outline.

Uses morphological gradient.

Parameters:

src - input image

dst - output image

Definition at line 194 of file image_processing.cpp.

5.10.2.5 void ImgProcMethods::removeBoundaryBlobs (IplImage * *src*, IplImage * *dst*, std::bitset< 4 > *flag*)

Removes the blobs on the image boudary.

Uses morphological operators.

Parameters:

src - input image

dst - output image

flag - flag deciding along which boundaries blobs will be removed; 4-bit value LRTB (left, right, top, bottom), 1 marks for removing; eg. 0011 means removing top and bottom boundary blobs.

5.10.2.6 void ImgProcMethods::removeSmallBlobs (IplImage * *src*, IplImage * *dst*, int *iter*)

Removes small blobs.

Uses morphological operators.

Parameters:

src - input image

dst - output image

iter - number of iterations

Definition at line 177 of file image_processing.cpp.

5.10.2.7 void ImgProcMethods::findContours (IplImage * *src*, CvSeq *& *contours*)

Finds contours in the image

Parameters:

src - input image

contours - output contours sequence

Definition at line 206 of file image_processing.cpp.

References contours_storage.

5.10.2.8 IplImage * ImgProcMethods::showHistogram (const IplImage * *src*, bool *display* = `false`, CvSize *size* = `cvSize(512,200)`)

Displays image histogram

Parameters:

src - input image

display - histogram window display flag

size - histogram image size

Returns:

Calculated histogram image.

Definition at line 288 of file image_processing.cpp.

5.10.2.9 void ImgProcMethods::stretchHistogram (const IplImage * *src*, IplImage * *dst*)

Transforms input image by histogram stretching

Parameters:

src - input image

dst - output image

Source www: http://tech.groups.yahoo.com/group/OpenCV/message/23708

Definition at line 399 of file image_processing.cpp.

5.10.2.10 int ImgProcMethods::findThresholdByHist (IplImage * *src*, int *wsp*)

Calculates threshold on the histogram basis

Parameters:

src - input image

wsp - ratio (wsp/100) of white pixels after input image thresholding ([0,100] range)

Returns:

calculated threshold value

Definition at line 411 of file image_processing.cpp.

5.10.2.11 int ImgProcMethods::doHoughTransform (IplImage * *src*, std::list< CvPoint > & *linesList*)

Hough transform (line searching)

Parameters:

src - input image

linesList - list of detected lines

Returns:

number of detected lines

5.10.2.12 void ImgProcMethods::detectObjectHaar (IplImage * *src*, CvHaarClassifierCascade * *cascade*, FaceComponent & *element*, double *scale* = 1, double *scale_factor* = 1.2, int *min_neighbors* = 5)

Detects any object using Haar classifier

Parameters:

src - input image

cascade - Haar cascade for the object

element - face component of the object

scale - result rectangle scaling ratio

scale_factor - Haar classifier scaling factor

min_neighbors - min count of neighbours

Definition at line 78 of file image_processing.cpp.

References storage.

5.10.3 Member Data Documentation

5.10.3.1 CvMemStorage * ImgProcMethods::contours_storage = cvCreateMemStorage(0) [static, protected]

Memory storage for contour finding operations

Definition at line 519 of file facet.h.

Referenced by findConnectedRegions(), and findContours().

5.10.3.2 CvMemStorage * ImgProcMethods::storage = cvCreateMemStorage(0) [static, protected]

Memory storage for other operations

Definition at line 522 of file facet.h.

Referenced by detectObjectHaar().

The documentation for this class was generated from the following files:

- inc/facet.h
- src/facet.cpp
- src/image_processing.cpp

5.11 Timer Class Reference

`#include <timer.h>`

Public Member Functions

- Timer ()
- bool start ()
- void stop ()
- double readSec ()
- double readSecSinceStart ()
- double readMSec ()
- double readMSecSinceStart ()

5.11.1 Detailed Description

Used for code sections timing by starting the timer and reading its value in appropriate points of code. Both milliseconds and seconds reads are provided.

Definition at line 56 of file timer.h.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 Timer::Timer () `[inline]`

Parameterless constructor

Definition at line 102 of file timer.h.

5.11.3 Member Function Documentation

5.11.3.1 bool Timer::start ()

Starts time measurement unless timer is running.

Return values:

true if the timer was idle

false if timer was running

Definition at line 73 of file timer.cpp.

Referenced by detectFromImage(), and detectFromStream().

5.11.3.2 void Timer::stop ()

Stops time measurement and sets running value to false

Definition at line 97 of file timer.cpp.

Referenced by detectFromImage(), and detectFromStream().

5.11.3.3 double Timer::readSec ()

Reads number of seconds since last read.

Returns:

number of seconds since last read

Definition at line 125 of file timer.cpp.

References BILLION.

Referenced by readMSec().

5.11.3.4 double Timer::readSecSinceStart ()

Reads number of seconds since timer start.

Returns:

number of seconds since timer start

Definition at line 165 of file timer.cpp.

References BILLION.

Referenced by readMSecSinceStart().

5.11.3.5 double Timer::readMSec ()

Reads number of milliseconds since last read.

Returns:

number of milliseconds since last read

Definition at line 183 of file timer.cpp.

References readSec().

5.11.3.6 double Timer::readMSecSinceStart ()

Reads number of milliseconds since timer start.

Returns:

number of milliseconds since timer start

Definition at line 187 of file timer.cpp.

References readSecSinceStart().

Referenced by detectFromImage(), and detectFromStream().

The documentation for this class was generated from the following files:

- demo/inc/timer.h
- demo/src/timer.cpp

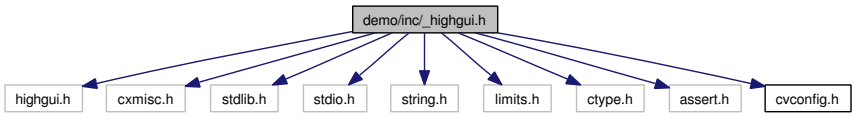
Chapter 6

File Documentation

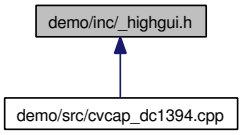
6.1 demo/inc/_highgui.h File Reference

```
#include "highgui.h"
#include "cxmisc.h"
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <limits.h>
#include <ctype.h>
#include <assert.h>
#include "cvconfig.h"
```

Include dependency graph for _highgui.h:



This graph shows which files directly or indirectly include this file:



6.1.1 Detailed Description

This file is quoted here from OpenCV sources (opencv-1.0.0/otherlibs/highgui) to facilitate the compilation of cvcap_dc1394.cpp, which was modified to become compatible with cvcap_v4l.cpp. This improves 1394

camera interface in OpenCV.

Definition in file _highgui.h.

6.2 demo/inc/captparam.h File Reference

Classes

- class ImageParameters

6.2.1 Detailed Description

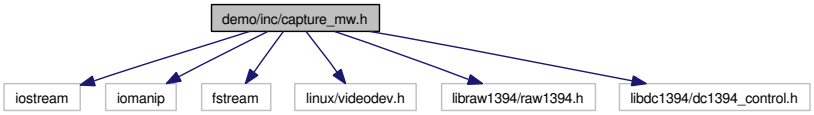
The file includes the definitions of camera image parameters used for image capturing setup.

Definition in file captparam.h.

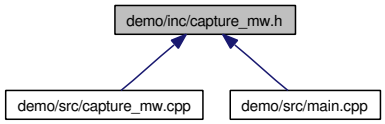
6.3 demo/inc/capture_mw.h File Reference

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <linux/videodev.h>
#include <libraw1394/raw1394.h>
#include <libdc1394/dc1394_control.h>
```

Include dependency graph for capture_mw.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ImageParameters
- class Capture

Defines

- #define MY_CHANNEL_NUMBER 1

6.3.1 Detailed Description

File defines the class Capture for image capturing in OpenCV.

Author:

Marek Wnuk

Date:

2009.03.13

Version:

1.00.00

Definition in file capture_mw.h.

6.3.2 Define Documentation

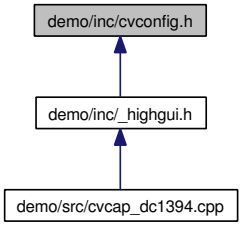
6.3.2.1 #define MY_CHANNEL_NUMBER 1

Definitions and prototypes for camera settings MW 13.03.09

Definition at line 139 of file capture_mw.h.

6.4 demo/inc/cvconfig.h File Reference

This graph shows which files directly or indirectly include this file:



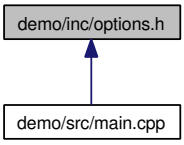
6.4.1 Detailed Description

This file is quoted here from OpenCV sources (opencv-1.0.0) to facilitate the compilation of cvcap_dc1394.cpp, which was modified to become compatible with cvcap_v4l.cpp. This improves 1394 camera interface in OpenCV.

Definition in file cvconfig.h.

6.5 demo/inc/options.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class DetectionOptions

6.5.1 Detailed Description

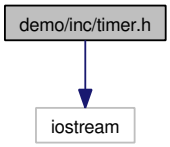
The file includes the definitions of user application options

Definition in file options.h.

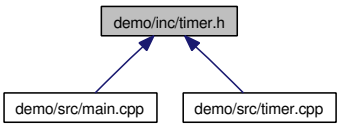
6.6 demo/inc/timer.h File Reference

```
#include <iostream>
```

Include dependency graph for timer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timer

Defines

- #define BILLION 1000000000

6.6.1 Detailed Description

File contains Timer class and functions for code execution timing.

Definition in file timer.h.

6.6.2 Define Documentation

6.6.2.1 #define BILLION 1000000000

One billion (10^9) constant

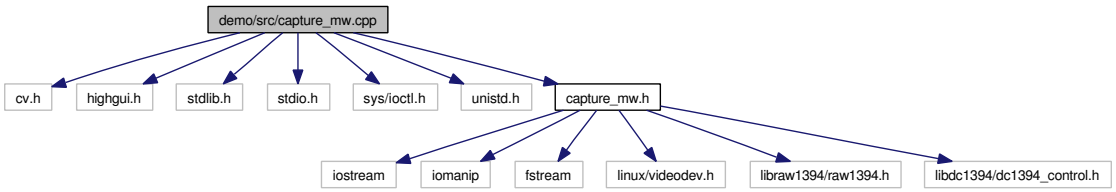
Definition at line 48 of file timer.h.

Referenced by Timer::readSec(), and Timer::readSecSinceStart().

6.7 demo/src/capture_mw.cpp File Reference

```
#include <cv.h>
#include <highgui.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/ioctl.h>
#include <unistd.h>
#include "capture_mw.h"
```

Include dependency graph for capture_mw.cpp:



6.7.1 Detailed Description

This file contains camera access functions for setting and checking a certain subset of common parameters for different camera interface types (actually, 1394 and v4l).

Author:

Marek Wnuk

Date:

2009.03.13

Version:

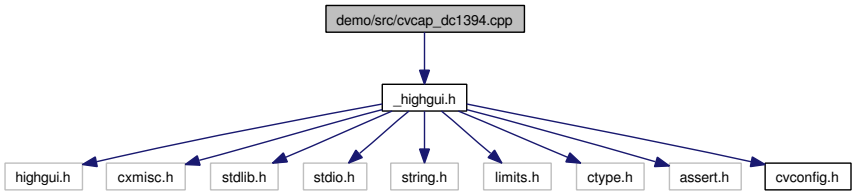
1.00.00

Definition in file capture_mw.cpp.

6.8 demo/src/cvcap_dc1394.cpp File Reference

```
#include "_highgui.h"
```

Include dependency graph for cvcap_dc1394.cpp:



6.8.1 Detailed Description

This file contains the reworked version of the original (see below) cvcap_dc1394.cpp found in opencv-1.0.0/otherlibs/highgui. It features immproved GetProperty functionality (common properties added). The icvGetPropertyCAM_DC1394() and icvSetPropertyCAM_DC1394() are normalized to [0,1] between min and max values, as in V4L. Retun values of [GS]etProperty have been changed to match V4L conventions: >=0 - ok, -1 - error.

Link this (compiled to .o) to your programs as an ad hoc patch (occluding highgui functions) to avoid broken highgui 1394 service.

Author:

Marek Wnuk

Date:

2009.03.09

Version:

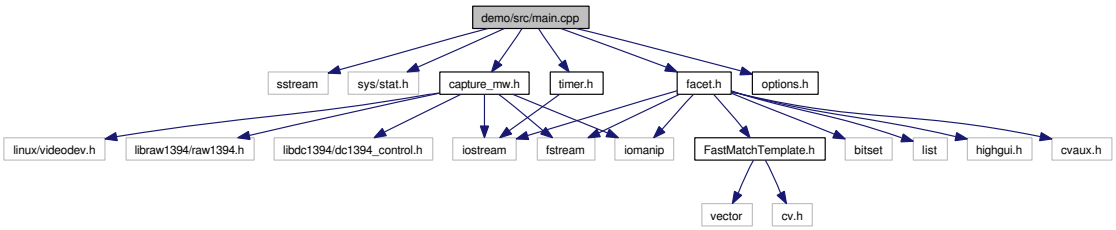
1.00.00

Definition in file cvcap_dc1394.cpp.

6.9 demo/src/main.cpp File Reference

```
#include <sstream>
#include <sys/stat.h>
#include "timer.h"
#include "facet.h"
#include "options.h"
#include "capture_mw.h"
```

Include dependency graph for main.cpp:



Functions

- bool readOptions (string file_name)
- void writeDataToFile (string file_name, int frame_no)
- void detectFromStream (int device)
- void detectFromImage ()
- void selectMenuOption (const char option)
- void showMenu ()
- int main (int argc, char **argv)

Variables

- int quiet = 0
- int detection_runs_counter = 0
- bool detrun = false
- Timer uclock
- std::string out_path
- std::string in_path
- CvCapture * capture
- CvVideoWriter * writer

6.9.1 Detailed Description

The file contains the sources of the FacET demo application.

Definition in file main.cpp.

6.9.2 Function Documentation

6.9.2.1 void detectFromImage ()

Face features detection from still image

Definition at line 321 of file main.cpp.

References Facet::cleanFacesList(), FFace::clearElements(), Facet::detectFeat(), detection_runs_counter, Facet::face, in_path, out_path, DetectionOptions::readImgrec(), Timer::readMSecSinceStart(), DetectionOptions::readParrec(), DetectionOptions::readShow(), DetectionOptions::readTiming(), DetectionOptions::readVerbose(), Timer::start(), Timer::stop(), and writeDataToFile().

Referenced by selectMenuOption().

6.9.2.2 void detectFromStream (int *device*)

Face features detection from video/camera

Definition at line 192 of file main.cpp.

References Capture::camera_if, Capture::camSetup(), capture, Capture::channel_no, Facet::cleanFacesList(), FFace::clearElements(), Facet::detectFeat(), detection_runs_counter, detrun, Capture::device_no, Facet::face, in_path, out_path, quiet, DetectionOptions::readImgrec(), Timer::readMSecSinceStart(), DetectionOptions::readParrec(), DetectionOptions::readShow(), DetectionOptions::readTiming(), DetectionOptions::readVerbose(), Capture::setImageParameters(), Timer::start(), Timer::stop(), Capture::video_norm, writeDataToFile(), and writer.

Referenced by selectMenuOption().

6.9.2.3 int main (int *argc*, char ** *argv*)

Main function of an application example

Definition at line 492 of file main.cpp.

References Capture::camera_if, Capture::channel_no, Capture::device_no, quiet, readOptions(), showMenu(), and Capture::video_norm.

6.9.2.4 bool readOptions (string *file_name*)

Read demo options from the config file

Parameters:

file_name - config file name

Definition at line 95 of file main.cpp.

References DetectionOptions::setImgrec(), DetectionOptions::setParrec(), DetectionOptions::setShow(), DetectionOptions::setTiming(), and DetectionOptions::setVerbose().

Referenced by main().

6.9.2.5 void selectMenuOption (const char *option*)

Selector function for menu example

Definition at line 382 of file main.cpp.

References detectFromImage(), detectFromStream(), and in_path.

Referenced by showMenu().

6.9.2.6 void showMenu ()

Menu function of an application example

Definition at line 448 of file main.cpp.

References out_path, DetectionOptions::readOutputImagePath(), and selectMenuOption().

Referenced by main().

6.9.2.7 void writeDataToFile (string *file_name*, int *frame_no*)

Write the parameters to the output file

Parameters:

file_name - output file name

Definition at line 157 of file main.cpp.

References Facet::facesList.

Referenced by detectFromImage(), and detectFromStream().

6.9.3 Variable Documentation

6.9.3.1 CvCapture* capture

Camera image capturing object

Definition at line 85 of file main.cpp.

Referenced by detectFromStream().

6.9.3.2 int detection_runs_counter = 0

The number of features detection runs

Definition at line 70 of file main.cpp.

Referenced by detectFromImage(), and detectFromStream().

6.9.3.3 bool detrun = false

Detection in progress flag

Definition at line 73 of file main.cpp.

Referenced by detectFromStream().

6.9.3.4 std:: string in_path

Path name for input file

Definition at line 82 of file main.cpp.

Referenced by detectFromImage(), detectFromStream(), and selectMenuOption().

6.9.3.5 std:: string out_path

Path name for output file

Definition at line 79 of file main.cpp.

Referenced by detectFromImage(), detectFromStream(), and showMenu().

6.9.3.6 int quiet = 0

Verbosity suppresion flag

Definition at line 67 of file main.cpp.

Referenced by detectFromStream(), and main().

6.9.3.7 Timer uclock

Procedures execution time counter

Definition at line 76 of file main.cpp.

6.9.3.8 CvVideoWriter* writer

Resulting video file writer object

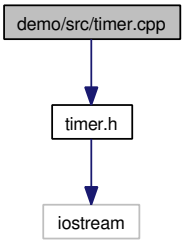
Definition at line 88 of file main.cpp.

Referenced by detectFromStream().

6.10 demo/src/timer.cpp File Reference

```
#include "timer.h"
```

Include dependency graph for timer.cpp:



6.10.1 Detailed Description

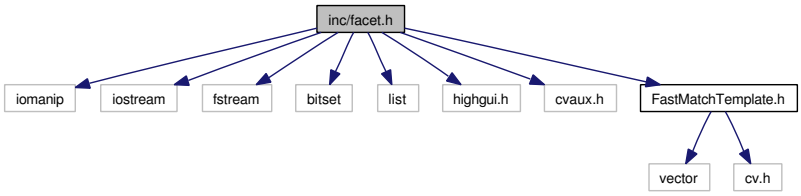
File contains Timer class and functions for code execution timing.

Definition in file timer.cpp.

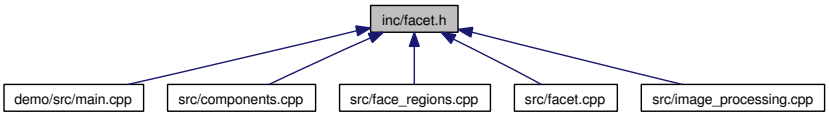
6.11 inc/facet.h File Reference

```
#include <iomanip>
#include <iostream>
#include <fstream>
#include <bitset>
#include <list>
#include <highgui.h>
#include <cvaux.h>
#include "FastMatchTemplate.h"
```

Include dependency graph for facet.h:



This graph shows which files directly or indirectly include this file:



Classes

- class FaceParameters
- class HaarParameters
- class FaceComponent
- class FFace
- class ImgProcMethods
- struct facepar_t
- class Facet

Defines

- #define MAX_SIZE 2048

Enumerations

- enum mean_type
- enum proj_type

- enum count_mode
- enum select_mode

Functions

- void counting (CvMat *matrix, int tab[MAX_SIZE], proj_type type, count_mode mode, int start, int end)
- void drawProjection (IplImage *img, int tab[MAX_SIZE], int start, int end, proj_type type, CvScalar color)
- int horizEyePosition (int tab[MAX_SIZE], int upperb, int lowerb)
- int horizMouthPosition (int tab[MAX_SIZE], int img_height, int upperb)
- void vertEyeBorder (int tab[MAX_SIZE], int img_width, int &ind_b, int &ind_e)
- int vertEyeCenter (int tab[MAX_SIZE], int img_width, int offset)
- void vertMouthBorder (int tab[MAX_SIZE], int width, int start, int end, int &lower_border, int &upper_border)
- double eyeMatching (IplImage *img, IplImage *templ, CvPoint &topLeft)
- void detectFacialRegions (IplImage *src, FaceComponent &eye_left, FaceComponent &eye_right, FaceComponent &mouth, int &width, bool detect_mouth)
- double eyeDetection (IplImage *eye_img, IplImage *templ, int correction, FaceComponent &iris)
- void findFaceRegions (IplImage *source_img, FFace &face, bool detect_mouth=false, bool detect_eyeballs=false)
- void example_usage (char *face)
- void drawStrLine (CvPoint p1, CvPoint p2, double &a, double &b)
- void drawStrLine (CvPoint p1, CvPoint p2, double &A, double &B, double &C)
- double calcAngleOX (double A, double B)
- double calcTwoLinesAngle (double A1, double B1, double A2, double B2)
- double calcTwoLinesAngle (double a1, double a2)
- double rad2deg (double rad)
- double deg2rad (double deg)
- double pointsDistance (CvPoint p1, CvPoint p2)

6.11.1 Detailed Description

File defines the classes for face features detection, parameterisation and displaying them in the image.

Definition in file facet.h.

6.11.2 Define Documentation

6.11.2.1 #define MAX_SIZE 2048

Max vertical and horizontal size of face image

Definition at line 225 of file facet.h.

6.11.3 Enumeration Type Documentation

6.11.3.1 enum count_mode

Selects counting mode

Definition at line 234 of file facet.h.

6.11.3.2 enum mean_type

Selects data averaging type

Definition at line 228 of file facet.h.

6.11.3.3 enum proj_type

Selects projection direction

Definition at line 231 of file facet.h.

6.11.3.4 enum select_mode

Selects option mode in selectRegion() function

Definition at line 237 of file facet.h.

6.11.4 Function Documentation

6.11.4.1 double calcAngleOX (double *A*, double *B*)

Calculates line slope angle

a=tg(alpha); where alpha - line slope angle a= -A/B; => tg(alfa)= -A/B => alfa = arctan(-A/B)

Parameters:

A,B - line equation parameters Ax + By + C = 0

Returns:

line slope angle value

Definition at line 65 of file components.cpp.

6.11.4.2 double calcTwoLinesAngle (double *a1*, double *a2*)

Calculates angle between two lines

Lines are defined by: y = a1x + b y = a2x + b

Parameters:

a1 - first line equation parameters

a2 - second line equation parameters

Returns:

value of the angle between two lines

Definition at line 78 of file components.cpp.

6.11.4.3 double calcTwoLinesAngle (double *A1*, double *B1*, double *A2*, double *B2*)

Calculates angle between two lines Lines are defined by: A1x + B1y + C1 = 0 A2x + B2y + C2 = 0

Parameters:

A1,B1 - first line equation parameters

A2,B2 - second line equation parameters

Returns:

value of the angle between two lines

Definition at line 70 of file components.cpp.

6.11.4.4 void counting (CvMat * *matrix*, int *tab*[MAX_SIZE], proj_type *type*, count_mode *mode*, int *start*, int *end*)

Counts the elements of the input matrix

Parameters:

matrix - input image matrix

tab - data array

type - direction of projection counting (horizontal/vertical)

mode - counting mode (normal/binary)

start - starting point of counting

end - ending point of counting

Definition at line 67 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.11.4.5 double deg2rad (double *deg*)

Converts degrees to radians

Parameters:

deg - value in degrees

Returns:

value in radians

Definition at line 89 of file components.cpp.

6.11.4.6 void detectFacialRegions (IplImage * *src*, FaceComponent & *eye_left*, FaceComponent & *eye_right*, FaceComponent & *mouth*, int & *width*, bool *detect_mouth*)

Detects eyes and mouth regions in face image

Parameters:

src - input (face) image

eye_left - left eye face component info

eye_right - right eye face component info

mouth - mouth face component info

width - face width

detect_mouth - mouth detection flag

Definition at line 297 of file face_regions.cpp.

References counting(), horizEyePosition(), horizMouthPosition(), FaceComponent::p1, FaceComponent::p2, vertEyeBorder(), vertEyeCenter(), and vertMouthBorder().

Referenced by findFaceRegions().

6.11.4.7 void drawProjection (IplImage * *img*, int *tab*[MAX_SIZE], int *start*, int *end*, proj_type *type*, CvScalar *color*)

Draws vertical or horizontal projection.

Parameters:

img - output image

tab - input data array

start - starting point

end - ending point

type - direction of projection (horizontal/vertical)

color - line colour

Definition at line 97 of file face_regions.cpp.

6.11.4.8 void drawStrLine (CvPoint *p1*, CvPoint *p2*, double & *A*, double & *B*, double & *C*)

Calculates equation parameters of straight-line through two points (p1.x, p1.y),(p2.x, p2.y). Variables A, B, C are set to values of parameters for line equation Ax + By + C = 0

Parameters:

p1,p2 - points

A,B,C - line equation parameters

Definition at line 54 of file components.cpp.

6.11.4.9 void drawStrLine (CvPoint *p1*, CvPoint *p2*, double & *a*, double & *b*)

Calculates equation parameters of straight-line through two points (p1.x, p1.y),(p2.x, p2.y).

Variables a, b are set to values of parameters for line equation y = ax + b

Parameters:

p1,p2 - points

a,b - line equation parameters

Definition at line 48 of file components.cpp.

6.11.4.10 void example_usage (char * *face*)

Example usage of function for coarse face subregions detection

Parameters:

face - face image file name

6.11.4.11 double eyeDetection (IplImage * *eye_img*, IplImage * *templ*, int *correction*, FaceComponent & *iris*)

Detects fine eye subregion in coarse eye region

Parameters:

eye_img - input (coarse eye region) image

templ - eye template image

correction - Y coordinate offset value (to be subtracted from the found pupil center position)

iris - component for the pupil position in coarse eye region

Returns:

template match accuracy ([0,100] range)

Definition at line 466 of file face_regions.cpp.

References eyeMatching(), and FaceComponent::FaceComponent().

Referenced by findFaceRegions().

6.11.4.12 double eyeMatching (IplImage * *img*, IplImage * *templ*, CvPoint & *topLeft*)

Detects eyes in coarse eye region by template matching

Parameters:

img - input (coarse eye region) image

templ - eye template image

topLeft - upper left corner of the matched eye subregion

Returns:

template match accuracy ([0,100] range)

Definition at line 252 of file face_regions.cpp.

Referenced by eyeDetection().

6.11.4.13 void findFaceRegions (IplImage * *source_img*, FFace & *face*, bool *detect_mouth* = false, bool *detect_eyeballs* = false)

Finds locations of face components subregions in face image

Parameters:

source_img - input (face) image

face - face describing structure to be filled

detect_mouth - mouth detection flag

detect_eyeballs - eyeballs detection flag

Definition at line 556 of file face_regions.cpp.

References detectFacialRegions(), eyeDetection(), FFace::left_eye, FFace::left_iris, FFace::mouth, FaceComponent::p1, FaceComponent::p2, FFace::right_eye, FFace::right_iris, and FFace::width.

Referenced by Facet::detectFeat().

6.11.4.14 int horizEyePosition (int *tab*[MAX_SIZE], int *upperb*, int *lowerb*)

Calculate horizontal eyes position

Parameters:

tab - input data array

upperb - upper limit for maximum searching

lowerb - lower limit for maximum searching

Returns:

- horizontal eyes position value

Definition at line 120 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.11.4.15 int horizMouthPosition (int *tab*[MAX_SIZE], int *img_height*, int *upperb*)

Calculate horizontal mouth position

Parameters:

tab - input data array

img_height - image height

upperb - upper limit for maximum searching

Returns:

horizontal mouth position value

Definition at line 146 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.11.4.16 double pointsDistance (CvPoint *p1*, CvPoint *p2*)

Calculates distance between two points

Parameters:

p1,p2 - points

Returns:

distance between the points

Definition at line 94 of file components.cpp.

6.11.4.17 double rad2deg (double *rad*)

Converts radians to degrees

Parameters:

rad - value in radians

Returns:

value in degrees

Definition at line 84 of file components.cpp.

6.11.4.18 void vertEyeBorder (int *tab*[MAX_SIZE], int *img_width*, int & *ind_b*, int & *ind_e*)

Calculate external vertical eyes boundaries

Parameters:

tab - input data array
img_width - image width
ind_b - variable for left vertical eyes boundary
ind_e - variable for right vertical eyes boundary

Definition at line 160 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.11.4.19 int vertEyeCenter (int *tab*[MAX_SIZE], int *img_width*, int *offset*)

Calculate vertical center of eyes region

Parameters:

tab - input data array
img_width - image width
offset - offset

Returns:

position of vertical center of eyes region

Definition at line 193 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.11.4.20 void vertMouthBorder (int *tab*[MAX_SIZE], int *width*, int *start*, int *end*, int & *lower_border*, int & *upper_border*)

Calculates vertical mouth region boundaries

Left boundary is set on the left side of the leftmost projection range of sufficient length, right boundary - on the right side of the rightmost projection range of the sufficient length.

Parameters:

tab - input data array

width - image width

start - starting point

end - ending point

lower_border - variable for left vertical mouth boundary

upper_border - variable for right vertical mouth boundary

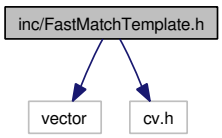
Definition at line 211 of file face_regions.cpp.

Referenced by detectFacialRegions().

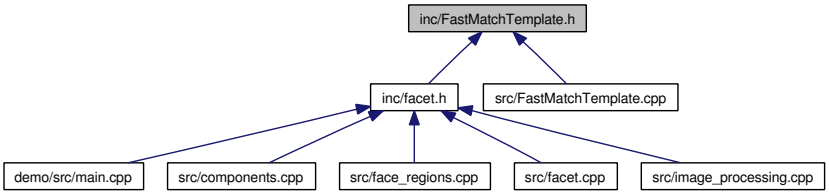
6.12 inc/FastMatchTemplate.h File Reference

```
#include <vector>
#include "cv.h"
```

Include dependency graph for FastMatchTemplate.h:



This graph shows which files directly or indirectly include this file:



6.12.1 Detailed Description

Id

FastMatchTemplate.h 5 2009-03-12 22:30:56Z mw

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Author:

Tristen Georgiou (Copyright © 2005 Tristen Georgiou)

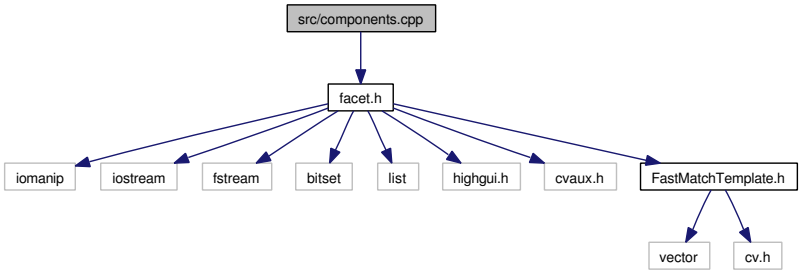
minor changes by Marcin Namysl 2008.06.28

Definition in file FastMatchTemplate.h.

6.13 src/components.cpp File Reference

```
#include "facet.h"
```

Include dependency graph for components.cpp:



Functions

- void drawStrLine (CvPoint p1, CvPoint p2, double &a, double &b)
- void drawStrLine (CvPoint p1, CvPoint p2, double &A, double &B, double &C)
- double calcAngleOX (double A, double B)
- double calcTwoLinesAngle (double A1, double B1, double A2, double B2)
- double calcTwoLinesAngle (double a1, double a2)
- double rad2deg (double rad)
- double deg2rad (double deg)
- double pointsDistance (CvPoint p1, CvPoint p2)

6.13.1 Detailed Description

File contains auxiliary functions for equation of straight-line through two points, calculation of angle between two lines, distance between two points and rad/deg conversion.

Definition in file components.cpp.

6.13.2 Function Documentation

6.13.2.1 double calcAngleOX (double *A*, double *B*)

Calculates line slope angle

a=tg(alpha); where alpha - line slope angle a= -A/B; => tg(alfa)= -A/B => alfa = arctan(-A/B)

Parameters:

A,B - line equation parameters Ax + By + C = 0

Returns:

line slope angle value

Definition at line 65 of file components.cpp.

6.13.2.2 double calcTwoLinesAngle (double *a1*, double *a2*)

Calculates angle between two lines

Lines are defined by: y = a1x + b y = a2x + b

Parameters:

a1 - first line equation parameters
a2 - second line equation parameters

Returns:

value of the angle between two lines

Definition at line 78 of file components.cpp.

6.13.2.3 double calcTwoLinesAngle (double *A1*, double *B1*, double *A2*, double *B2*)

Calculates angle between two lines Lines are defined by: A1x + B1y + C1 = 0 A2x + B2y + C2 = 0

Parameters:

A1,B1 - first line equation parameters
A2,B2 - second line equation parameters

Returns:

value of the angle between two lines

Definition at line 70 of file components.cpp.

6.13.2.4 double deg2rad (double *deg*)

Converts degrees to radians

Parameters:

deg - value in degrees

Returns:

value in radians

Definition at line 89 of file components.cpp.

6.13.2.5 void drawStrLine (CvPoint *p1*, CvPoint *p2*, double & *A*, double & *B*, double & *C*)

Calculates equation parameters of straight-line through two points (p1.x, p1.y),(p2.x, p2.y). Variables A, B, C are set to values of parameters for line equation Ax + By + C = 0

Parameters:

p1,p2 - points
A,B,C - line equation parameters

Definition at line 54 of file components.cpp.

6.13.2.6 void drawStrLine (CvPoint *p1*, CvPoint *p2*, double & *a*, double & *b*)

Calculates equation parameters of straight-line through two points (p1.x, p1.y),(p2.x, p2.y).

Variables a, b are set to values of parameters for line equation y = ax + b

Parameters:

p1,p2 - points

a,b - line equation parameters

Definition at line 48 of file components.cpp.

6.13.2.7 double pointsDistance (CvPoint *p1*, CvPoint *p2*)

Calculates distance between two points

Parameters:

p1,p2 - points

Returns:

distance between the points

Definition at line 94 of file components.cpp.

6.13.2.8 double rad2deg (double *rad*)

Converts radians to degrees

Parameters:

rad - value in radians

Returns:

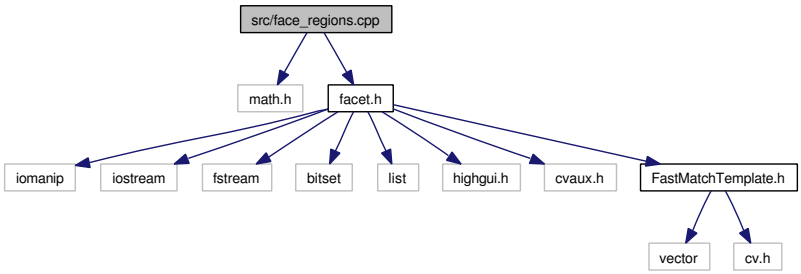
value in degrees

Definition at line 84 of file components.cpp.

6.14 src/face_regions.cpp File Reference

```
#include <math.h>
#include "facet.h"
```

Include dependency graph for face_regions.cpp:



Functions

- void counting (CvMat *matrix, int tab[MAX_SIZE], proj_type type, count_mode mode, int start, int end)
- void drawProjection (IplImage *img, int tab[MAX_SIZE], int start, int end, proj_type type, CvScalar color)
- int horizEyePosition (int tab[MAX_SIZE], int upperb, int lowerb)
- int horizMouthPosition (int tab[MAX_SIZE], int img_height, int upperb)
- void vertEyeBorder (int tab[MAX_SIZE], int img_width, int &ind_b, int &ind_e)
- int vertEyeCenter (int tab[MAX_SIZE], int img_width, int offset)
- void vertMouthBorder (int tab[MAX_SIZE], int width, int start, int end, int &lower_border, int &upper_border)
- double eyeMatching (IplImage *img, IplImage *templ, CvPoint &topLeft)
- void detectFacialRegions (IplImage *src, FaceComponent &eye_left, FaceComponent &eye_right, FaceComponent &mouth, int &width, bool detect_mouth)
- double eyeDetection (IplImage *eye_img, IplImage *templ, int correction, FaceComponent &iris)
- void findFaceRegions (IplImage *source_img, FFace &face, bool detect_mouth, bool detect_eyeballs)

6.14.1 Detailed Description

The file defines a set of functions for face subregions (mouth and eyes) detection using gradient projection method.

It contains FaceComponent class definition, which can be used to describe a single face component (mouth, eyes, eyebrows, nose, forehead). The FFace class is defined, which contains objects of type FaceComponent (mouth, eyes, eyebrows, forehead, eyelids).

Definition in file face_regions.cpp.

6.14.2 Function Documentation

6.14.2.1 void counting (CvMat * *matrix*, int *tab*[MAX_SIZE], proj_type *type*, count_mode *mode*, int *start*, int *end*)

Counts the elements of the input matrix

Parameters:

matrix - input image matrix

tab - data array

type - direction of projection counting (horizontal/vertical)

mode - counting mode (normal/binary)

start - starting point of counting

end - ending point of counting

Definition at line 67 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.14.2.2 void detectFacialRegions (IplImage * *src*, FaceComponent & *eye_left*, FaceComponent & *eye_right*, FaceComponent & *mouth*, int & *width*, bool *detect_mouth*)

Detects eyes and mouth regions in face image

Parameters:

src - input (face) image

eye_left - left eye face component info

eye_right - right eye face component info

mouth - mouth face component info

width - face width

detect_mouth - mouth detection flag

Definition at line 297 of file face_regions.cpp.

References counting(), horizEyePosition(), horizMouthPosition(), FaceComponent::p1, FaceComponent::p2, vertEyeBorder(), vertEyeCenter(), and vertMouthBorder().

Referenced by findFaceRegions().

6.14.2.3 void drawProjection (IplImage * *img*, int *tab*[MAX_SIZE], int *start*, int *end*, proj_type *type*, CvScalar *color*)

Draws vertical or horizontal projection.

Parameters:

img - output image

tab - input data array

start - starting point

end - ending point

type - direction of projection (horizontal/vertical)

color - line colour

Definition at line 97 of file face_regions.cpp.

6.14.2.4 double eyeDetection (IplImage * *eye_img*, IplImage * *templ*, int *correction*, FaceComponent & *iris*)

Detects fine eye subregion in coarse eye region

Parameters:

eye_img - input (coarse eye region) image

templ - eye template image

correction - Y coordinate offset value (to be subtracted from the found pupil center position)

iris - component for the pupil position in coarse eye region

Returns:

template match accuracy ([0,100] range)

Definition at line 466 of file face_regions.cpp.

References eyeMatching(), and FaceComponent::FaceComponent().

Referenced by findFaceRegions().

6.14.2.5 double eyeMatching (IplImage * *img*, IplImage * *templ*, CvPoint & *topLeft*)

Detects eyes in coarse eye region by template matching

Parameters:

img - input (coarse eye region) image

templ - eye template image

topLeft - upper left corner of the matched eye subregion

Returns:

template match accuracy ([0,100] range)

Definition at line 252 of file face_regions.cpp.

Referenced by eyeDetection().

6.14.2.6 void findFaceRegions (IplImage * *source_img*, FFace & *face*, bool *detect_mouth* = `false`, bool *detect_eyeballs* = `false`)

Finds locations of face components subregions in face image

Parameters:

source_img - input (face) image

face - face describing structure to be filled

detect_mouth - mouth detection flag

detect_eyeballs - eyeballs detection flag

Definition at line 556 of file face_regions.cpp.

References detectFacialRegions(), eyeDetection(), FFace::left_eye, FFace::left_iris, FFace::mouth, FaceComponent::p1, FaceComponent::p2, FFace::right_eye, FFace::right_iris, and FFace::width.

Referenced by Facet::detectFeat().

6.14.2.7 int horizEyePosition (int *tab*[MAX_SIZE], int *upperb*, int *lowerb*)

Calculate horizontal eyes position

Parameters:

tab - input data array

upperb - upper limit for maximum searching

lowerb - lower limit for maximum searching

Returns:

- horizontal eyes position value

Definition at line 120 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.14.2.8 int horizMouthPosition (int *tab*[MAX_SIZE], int *img_height*, int *upperb*)

Calculate horizontal mouth position

Parameters:

tab - input data array

img_height - image height

upperb - upper limit for maximum searching

Returns:

horizontal mouth position value

Definition at line 146 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.14.2.9 void vertEyeBorder (int *tab*[MAX_SIZE], int *img_width*, int & *ind_b*, int & *ind_e*)

Calculate external vertical eyes boundaries

Parameters:

tab - input data array

img_width - image width

ind_b - variable for left vertical eyes boundary

ind_e - variable for right vertical eyes boundary

Definition at line 160 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.14.2.10 int vertEyeCenter (int *tab*[MAX_SIZE], int *img_width*, int *offset*)

Calculate vertical center of eyes region

Parameters:

tab - input data array

img_width - image width

offset - offset

Returns:

position of vertical center of eyes region

Definition at line 193 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.14.2.11 void vertMouthBorder (int *tab*[MAX_SIZE], int *width*, int *start*, int *end*, int & *lower_border*, int & *upper_border*)

Calculates vertical mouth region boundaries

Left boundary is set on the left side of the leftmost projection range of sufficient length, right boundary - on the right side of the rightmost projection range of the sufficient length.

Parameters:

tab - input data array

width - image width

start - starting point

end - ending point

lower_border - variable for left vertical mouth boundary

upper_border - variable for right vertical mouth boundary

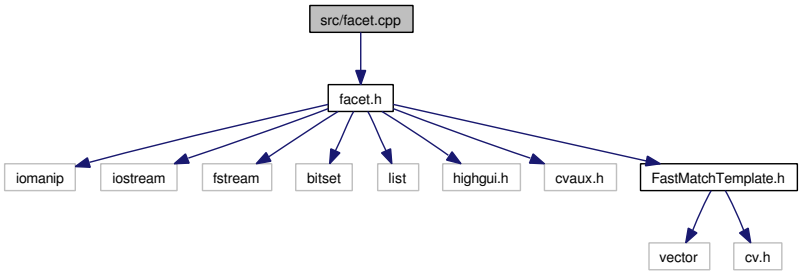
Definition at line 211 of file face_regions.cpp.

Referenced by detectFacialRegions().

6.15 src/facet.cpp File Reference

```
#include "facet.h"
```

Include dependency graph for facet.cpp:



6.15.1 Detailed Description

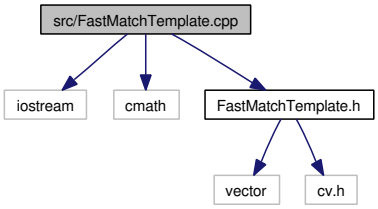
File defines the class for face features detection, parameterisation and displaying them in the image.

Definition in file facet.cpp.

6.16 src/FastMatchTemplate.cpp File Reference

```
#include <iostream>
#include <cmath>
#include "FastMatchTemplate.h"
```

Include dependency graph for FastMatchTemplate.cpp:



6.16.1 Detailed Description

Id

FastMatchTemplate.cpp 5 2009-03-12 22:30:56Z mw

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Author:

Tristen Georgiou (Copyright © 2005 Tristen Georgiou)

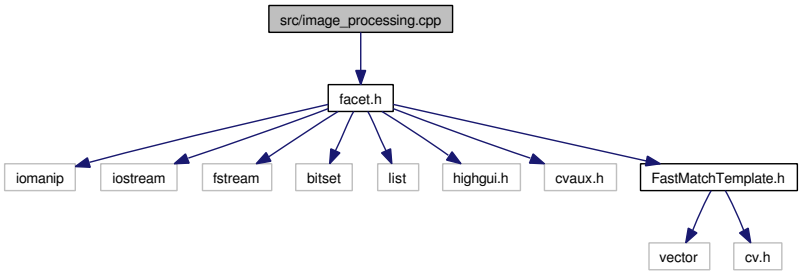
minor changes by Marcin Namysl 2008.06.28

Definition in file FastMatchTemplate.cpp.

6.17 src/image_processing.cpp File Reference

```
#include "facet.h"
```

Include dependency graph for image_processing.cpp:



6.17.1 Detailed Description

File defines class implementing image processing methods: hysteresis thresholding, silhouette outline creation, Hough transform, etc.

Definition in file image_processing.cpp.

Index