CSI:PC2 — Frontend Reference Manual

0.3

Generated by Doxygen 1.5.3

Tue May 20 16:25:28 2008

Contents

1 CSI:PC2 — Frontend Namespace Index

1.1 CSI:PC2 — Frontend Namespace List

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

2 CSI:PC2 — Frontend Class Index

2.1 CSI:PC2 — Frontend Class List

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

3 CSI:PC2 — Frontend File Index

3.1 CSI:PC2 — Frontend File List

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

4 CSI:PC2 — Frontend Namespace Documentation

4.1 Iris Namespace Reference

Classes

- class IrisCode
- class IrisCodeMatcher

5 CSI:PC2 — Frontend Class Documentation

5.1 CAboutDlg Class Reference

Public Types

- enum { IDD = IDD_ABOUTBOX }

Public Member Functions

- CAboutDlg ()

Protected Member Functions

- virtual void DoDataExchange (CDataExchange *pDX)

5.1.1 Detailed Description

Definition at line 89 of file gui_control.cpp.

5.1.2 Member Enumeration Documentation

5.1.2.1 anonymous enum

Enumerator:

IDD

Definition at line 95 of file gui_control.cpp.

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

- gui_control.cpp

5.2 Capture Class Reference

Public Member Functions

- Capture ()
- Capture (CCltifa_apix1 *api)
- ~Capture (void)
- bool start (void)
- void stop (void)
- bool captureImage (void)
- bool isPreviewing (void)
- bool isUsingIrisGuardAPI ()
- void setUsingIrisGuardAPI (bool value)

5.2.1 Detailed Description

Definition at line 21 of file Capture.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 Capture::Capture ()

Constructor of Capture class, initializes all variables to capturing with ordinary camera

Definition at line 30 of file Capture.cpp.

5.2.2.2 Capture::Capture (CCltifa_apix1 * *api*)

Constructor of Capture class, initializes all variables to capturing with H100

Parameters:

* api pointer to the active x component of the H100-API

Definition at line 48 of file Capture.cpp.

5.2.2.3 Capture::~Capture (void)

Destructor of Capture class, releases all ressources and closes the preview windows

Definition at line 65 of file Capture.cpp.

References stop().

5.2.3 Member Function Documentation

5.2.3.1 bool Capture::start (void)

Starts previewing in a new window

Returns:

true if previewing is active, else false

Definition at line 128 of file Capture.cpp.

5.2.3.2 void Capture::stop (void)

Stops previewing, releases ressources

Definition at line 164 of file Capture.cpp.

References SAFE_RELEASE.

5.2.3.3 bool Capture::isUsingIrisGuardAPI ()

Inducates if using the IrisGuard API or not

Returns:

true if using IrisGuardAPI, else false

Definition at line 337 of file Capture.cpp.

5.2.3.4 void Capture::setUsingIrisGuardAPI (bool *value*)

Set the API, which is used

Parameters:

value true if using IrisGuard-API, else false

Definition at line 346 of file Capture.cpp.

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

- Capture.h
- Capture.cpp

5.3 cluster_centroid_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- angle_t orientation_vector [ORIENTATION_VECTOR_SIZE]
- uint8_t orientation_vector_mask [ORIENTATION_VECTOR_SIZE]

5.3.1 Detailed Description

Structure for a cluster centroid for the fingerprint clustering matching algorithm.

Definition at line 55 of file csi_struct.h.

5.3.2 Member Data Documentation

5.3.2.1 angle_t cluster_centroid_t::orientation_vector[ORIENTATION_VECTOR_SIZE]

orientation vector

Definition at line 56 of file csi_struct.h.

5.3.2.2 uint8_t cluster_centroid_t::orientation_vector_mask[ORIENTATION_VECTOR_SIZE]

orientation vectors mask

Definition at line 57 of file csi_struct.h.

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

- csi_struct.h

5.4 cluster_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- uint32_t nr_of_clusters
- uint32_t nr_of_bins
- cluster_centroid_t * centroids
- ard_vector_t ** ard_centers
- uint32_t * counts

5.4.1 Detailed Description

Structure for the clustering data after the offline process

Definition at line 62 of file csi_struct.h.

5.4.2 Member Data Documentation

5.4.2.1 uint32_t cluster_t::nr_of_clusters

of clusters

Definition at line 63 of file csi_struct.h.

5.4.2.2 uint32_t cluster_t::nr_of_bins

of bins

Definition at line 64 of file csi_struct.h.

5.4.2.3 cluster_centroid_t* cluster_t::centroids

cluster centroid

Definition at line 65 of file csi_struct.h.

5.4.2.4 ard_vector_t** cluster_t::ard_centers

ARD Vector

Definition at line 66 of file csi_struct.h.

5.4.2.5 uint32_t* cluster_t::counts

counts

Definition at line 67 of file csi_struct.h.

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

- csi_struct.h

5.5 coordinate_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- pixel_t x
- pixel_t y

5.5.1 Detailed Description

Structure for a x,y coordinate point.

Definition at line 75 of file csi_struct.h.

5.5.2 Member Data Documentation

5.5.2.1 pixel_t coordinate_t::x

coordinate in x direction

Definition at line 76 of file csi_struct.h.

5.5.2.2 pixel_t coordinate_t::y

coordinate in y direction

Definition at line 77 of file csi_struct.h.

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

- csi_struct.h

5.6 csi_timer_t Struct Reference

Public Attributes

- struct timeval * start
- struct timeval * stop

5.6.1 Detailed Description

Definition at line 40 of file csi_time.h.

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

- csi_time.h

5.7 CStateEvents Class Reference

Public Member Functions

- CStateEvents ()
- ~CStateEvents ()
- void AddStateEvent (Event *event)
- Event * GetStateEvent ()

5.7.1 Detailed Description

event queue class

Definition at line 74 of file fingerCapture.cpp.

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

- fingerCapture.cpp

5.8 database_managment Struct Reference

```
#include <csi_struct.h>
```

Public Attributes

- int mod_counter
- int counter
- char type [3]
- char path [300]

5.8.1 Detailed Description

Structure for Database Managment purposes

Definition at line 157 of file csi_struct.h.

5.8.2 Member Data Documentation

5.8.2.1 int database_managment::mod_counter

The number of headnodes.

Definition at line 158 of file csi_struct.h.

5.8.2.2 int database_managment::counter

The number of templates stored in the database.

Definition at line 159 of file csi_struct.h.

5.8.2.3 char database_managment::type[3]

Type of database templates: iriscodes (ir), filterbank (ff), clustering (fc) or minutiae (fm).

Definition at line 160 of file csi_struct.h.

5.8.2.4 char database_managment::path[300]

Path to the specific database.

Definition at line 161 of file csi_struct.h.

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

- csi_struct.h

5.9 db_init Struct Reference

`#include <csi_struct.h>`

Public Attributes

- triple_db fp_mng
- database_managment ir_mng
- FILE * fp_min
- FILE * fp_clu
- FILE * fp_fil
- FILE * fp_ir

5.9.1 Detailed Description

Structure for a initialising the database.

Definition at line 174 of file csi_struct.h.

5.9.2 Member Data Documentation

5.9.2.1 triple_db db_init::fp_mng

triple db management struct

Definition at line 175 of file csi_struct.h.

5.9.2.2 database_managment db_init::ir_mng

database management struct

Definition at line 176 of file csi_struct.h.

5.9.2.3 FILE* db_init::fp_min

filepointer for minutia db

Definition at line 177 of file csi_struct.h.

5.9.2.4 FILE* db_init::fp_clu

filepointer for cluster db

Definition at line 178 of file csi_struct.h.

5.9.2.5 FILE* db_init::fp_fil

filepointer for filterbank db

Definition at line 179 of file csi_struct.h.

5.9.2.6 FILE* db_init::fp_ir

filepointer for iris db

Definition at line 180 of file csi_struct.h.

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

- csi_struct.h

5.10 Event Class Reference

Public Member Functions

- Event (Event &aEvent)

- Event (char cmd)
- Event (TST_CallbackMessage *msg)

Public Attributes

- EventType type

Classes

- union ucontent

5.10.1 Detailed Description

Definition at line 43 of file fingerCapture.cpp.

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

- fingerCapture.cpp

5.11 Event::ucontent Union Reference

Public Attributes

- char cmd
- TST_CallbackMessage * msg

5.11.1 Detailed Description

Definition at line 47 of file fingerCapture.cpp.

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

- fingerCapture.cpp

5.12 feature_map_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- gray_value_t value [NUM_SECTORS][NUM_RINGS]

5.12.1 Detailed Description

Structure for the feature map of a filterbank based fingerprint template.

Definition at line 120 of file csi_struct.h.

5.12.2 Member Data Documentation

5.12.2.1 gray_value_t feature_map_t::value[NUM_SECTORS][NUM_RINGS]

Structure for the feature map of a filterbank based fingerprint template.

Definition at line 121 of file csi_struct.h.

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

- csi_struct.h

5.13 frontend_data_t Union Reference

`#include <csi_struct.h>`

Public Attributes

- frontend_iris_data_t iris
- frontend_finger_data_t finger

5.13.1 Detailed Description

This union defines a standard datapack in the request/message from the frontend/client. it inherites all data only for iris _OR_ for fingerprints.

Definition at line 241 of file csi_struct.h.

5.13.2 Member Data Documentation

5.13.2.1 frontend_iris_data_t frontend_data_t::iris

Structure of the frontend iris data package.

Definition at line 242 of file csi_struct.h.

5.13.2.2 frontend_finger_data_t frontend_data_t::finger

Structure of the frontend fingerprint package.

Definition at line 243 of file csi_struct.h.

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

- csi_struct.h

5.14 frontend_finger_data_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- jobmode_t job
- extractmode_t extractmode
- uint32_t raw_size
- template_filterbank_t tmpl_filterbank
- template_minutiae_t tmpl_minutiae
- template_cluster_t tmpl_cluster
- char * raw

5.14.1 Detailed Description

Type definition of the frontend data structure for fingerprints.

Definition at line 227 of file csi_struct.h.

5.14.2 Member Data Documentation

5.14.2.1 jobmode_t frontend_finger_data_t::job

ID of the job.

Definition at line 228 of file csi_struct.h.

5.14.2.2 extractmode_t frontend_finger_data_t::extractmode

Value of the extraction type.

Definition at line 229 of file csi_struct.h.

5.14.2.3 uint32_t frontend_finger_data_t::raw_size

Size of the attached picture stream.

Definition at line 230 of file csi_struct.h.

5.14.2.4 template_filterbank_t frontend_finger_data_t::tmpl_filterbank

Filterbank template structure.

Definition at line 231 of file csi_struct.h.

5.14.2.5 template_minutiae_t frontend_finger_data_t::tmpl_minutiae

Minutiae template structure.

Definition at line 232 of file csi_struct.h.

5.14.2.6 template_cluster_t frontend_finger_data_t::tmpl_cluster

Cluster template structure.

Definition at line 233 of file csi_struct.h.

5.14.2.7 char* frontend_finger_data_t::raw

Pointer to the raw picture stream.

Definition at line 234 of file csi_struct.h.

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

- csi_struct.h

5.15 frontend_iris_data_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- jobmode_t job
- extractmode_t extractmode
- uint32_t raw_size
- template_iris_t templat
- char * raw

5.15.1 Detailed Description

Type definition of the frontend data structure for iriscodes.

Definition at line 216 of file csi_struct.h.

5.15.2 Member Data Documentation

5.15.2.1 jobmode_t frontend_iris_data_t::job

ID of the job.

Definition at line 217 of file csi_struct.h.

5.15.2.2 extractmode_t frontend_iris_data_t::extractmode

Value of the extraction type.

Definition at line 218 of file csi_struct.h.

5.15.2.3 uint32_t frontend_iris_data_t::raw_size

Size of the attached picture stream.

Definition at line 219 of file csi_struct.h.

5.15.2.4 template_iris_t frontend_iris_data_t::templat

Structure for the iris template.

Definition at line 220 of file csi_struct.h.

5.15.2.5 char* frontend_iris_data_t::raw

Pointer to the raw picture stream.

Definition at line 221 of file csi_struct.h.

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

- csi_struct.h

5.16 gui_control Class Reference

`#include <gui_control.h>`

Public Types

- enum { IDD = IDD_GUI_DIALOG }

Public Member Functions

- gui_control (CWnd *pParent=NULL)
- void showHelpBalloons ()
- afx_msg void OnEnChangeEdit1 ()
- afx_msg void OnBnClickedButton1 ()
- afx_msg void OnBnClickedButton2 ()
- afx_msg void OnBnClickedButton3 ()
- afx_msg void OnBnClickedButton4 ()
- afx_msg void OnBnClickedButton5 ()
- afx_msg void OnBnClickedButton6 ()
- afx_msg void OnBnClickedButton7 ()
- afx_msg void OnBnClickedButton8 ()
- afx_msg void OnLButtonDown (UINT nFlags, CPoint point)
- afx_msg void OnBnClickedCheck1 ()

Public Attributes

- CButton m_cbButton1
- CButton m_cbButton2
- CButton m_cbButton3
- CButton m_cbButton4
- CButton m_cbButton5
- CButton m_cbButton6
- CButton m_cbButton7
- CButton m_cbButton8
- CString m_csEdit1
- CPicture m_Picture
- CButton m_cbCheck1
- CString scantype
- CString tmpPicture

Protected Member Functions

- virtual void DoDataExchange (CDataExchange *pDX)
- virtual BOOL OnInitDialog ()
- afx_msg void OnSysCommand (UINT nID, LPARAM lParam)
- afx_msg void OnPaint ()
- afx_msg HCURSOR OnQueryDragIcon ()

Protected Attributes

- HICON m_hIcon

5.16.1 Detailed Description

The gui-control class used for the gui man-machine interaction, frontend-cluster communication and integration of frontend-modules.

Definition at line 24 of file gui_control.h.

5.16.2 Member Enumeration Documentation

5.16.2.1 anonymous enum

Enumerator:

IDD

Definition at line 31 of file gui_control.h.

5.16.3 Member Function Documentation

5.16.3.1 void gui_control::showHelpBalloons ()

Method for positioning and handling the tooltips. Implemented as a state-machine. Every state contains different textinformation and is connected to different buttons

Definition at line 365 of file gui_control.cpp.

References help_on, help_status, lf, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, m_pBalloonTip, m_pBalloonTip2, and m_pBalloonTip3.

5.16.3.2 void gui_control::DoDataExchange (CDataExchange * *pDX*) [protected, virtual]

Standard MFC-Method. Elements of the GUI (e.g. Buttons) must be initialized here

Definition at line 129 of file gui_control.cpp.

References apix, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, m_cbCheck1, and m_csEdit1.

5.16.3.3 virtual BOOL gui_control::OnInitDialog () [protected, virtual]

Method is called when the program starts. Symbols and menues be initalized. Resize-informations are set. Ini-configuration variables are set. Iriscamera-mode is selected and temporary pictures are set.

5.16.3.4 void gui_control::OnSysCommand (UINT *nID*, LPARAM *lParam*) [protected]

Standard MFC-Method to handle system-calls (e.g. showing the about-dialog).

Definition at line 280 of file gui_control.cpp.

5.16.3.5 void gui_control::OnPaint () [protected]

Standard MFC-Method for painting the dialog and symbols. Additional implemeted the preview-pictures and reactivating tooltips.

Definition at line 297 of file gui_control.cpp.

References help_on, imgShow, m_hIcon, m_Picture, preview, rect, and showHelpBalloons().

5.16.3.6 HCURSOR gui_control::OnQueryDragIcon () [protected]

Standard MFC-Method.

Definition at line 351 of file gui_control.cpp.

References m_hIcon.

5.16.3.7 void gui_control::OnEnChangeEdit1 ()

This Method is used when the textfield in the dialog is changed. In this program this method is not used.

Definition at line 358 of file gui_control.cpp.

5.16.3.8 void gui_control::OnBnClickedButton1 ()

Method called by clicking on Iris-Button. Iris-camera is called and saves an image. A variable to show the image is set.

Definition at line 627 of file gui_control.cpp.

References Capture::captureImage(), help_status, imgShow, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, OnPaint(), scantype, showHelpBalloons(), and tmpPicture.

5.16.3.9 void gui_control::OnBnClickedButton2 ()

Method called by clicking on Finger-Button. Fingerprintsensor is active and saves an image. A variable to show the image is set.

Definition at line 659 of file gui_control.cpp.

References finger_scanner_path, help_status, imgShow, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, OnPaint(), scantype, showHelpBalloons(), and tmpPicture.

5.16.3.10 void gui_control::OnBnClickedButton3 ()

Method called by clicking on Cancel-Button. Variables are reset.

Definition at line 688 of file gui_control.cpp.

References help_status, iris_capture_flag, iris_ext_flag, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, and showHelpBalloons().

5.16.3.11 void gui_control::OnBnClickedButton4 ()

Method called by clicking on Rescan-Button. Depending on scantype an iris or a finger is scanned.

Definition at line 706 of file gui_control.cpp.

References Capture::captureImage(), finger_scanner_path, help_status, imgShow, iris_ext_flag, m_cbButton1, OnPaint(), scantype, showHelpBalloons(), and tmpPicture.

5.16.3.12 void gui_control::OnBnClickedButton5 ()

Method called by clicking on OK-Button. Depending on scantype an iris or a finger feature-extraction is executed if the extraction is on frontend. For iris are two steps of

extraction: the preview and the code generation. If extraction_on_frontend is set to 0 in the ini-File the method does nothing.

Definition at line 733 of file gui_control.cpp.

References finger_extraction_on_frontend, finger_extraction_path, help_on, help_status, imgShow, iris_ext_flag, iris_extraction_on_frontend, iris_extraction_path, iris_extraction_path2, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, OnPaint(), preview, previewClose(), scantype, showHelpBalloons(), and tmpPicture.

5.16.3.13 void gui_control::OnBnClickedButton6 ()

Method called by clicking on Match-Button. Depending on scantype an iris or a finger matching is executed. If the extraction is on the frontend, the template-data is sent to the cluster. Otherwise the picture is sent. For finger-matching one of four methods can be selected in the ini-File.

Definition at line 846 of file gui_control.cpp.

References EXTRACT_FALSE, EXTRACT_TRUE, frontend_finger_data_t::extractmode, frontend_iris_data_t::extractmode, frontend_data_t::finger, finger_extraction_on_frontend, finger_match_mode, help_status, frontend_data_t::iris, iris_extraction_on_frontend, frontend_finger_data_t::job, frontend_iris_data_t::job, JOB_MATCH_FINGER, JOB_MATCH_FINGER_CLUSTER, JOB_MATCH_FINGER_FILTERBANK, JOB_MATCH_FINGER_MINUTIAE, JOB_MATCH_FINGER_MIXED, JOB_MATCH_IRIS, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, m_csEdit1, message, port, frontend_finger_data_t::raw_size, frontend_iris_data_t::raw_size, scantype, server, showHelpBalloons(), frontend_iris_data_t::templat, templates_fingerprint_t::template_cluster, templates_fingerprint_t::template_filterbank, templates_fingerprint_t::template_minutiae, frontend_finger_data_t::tmpl_cluster, frontend_finger_data_t::tmpl_filterbank, and frontend_finger_data_t::tmpl_minutiae.

5.16.3.14 void gui_control::OnBnClickedButton7 ()

Method called by clicking on Save-Button. Depending on scantype an iris or a finger save is executed. If the extraction is on the frontend, the template-data and the image is sent to the cluster. Otherwise only the picture is sent.

Definition at line 1158 of file gui_control.cpp.

References EXTRACT_FALSE, EXTRACT_TRUE, frontend_finger_data_t::extractmode, frontend_iris_data_t::extractmode, frontend_data_t::finger, finger_extraction_on_frontend, help_status, frontend_data_t::iris, iris_extraction_on_frontend, frontend_finger_data_t::job, frontend_iris_data_t::job, JOB_SAVE_FINGER, JOB_SAVE_IRIS, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, m_csEdit1, message, port, frontend_finger_data_t::raw_size, frontend_iris_data_t::raw_size, scantype, server, showHelpBalloons(), frontend_iris_data_t::templat, templates_fingerprint_t::template_cluster, templates_fingerprint_t::template_filterbank, templates_fingerprint_t::template_minutiae, frontend_finger_data_t::tmpl_cluster,

frontend_finger_data_t::tmpl_filterbank, and frontend_finger_data_t::tmpl_minutiae.

5.16.3.15 void gui_control::OnBnClickedButton8 ()

Method called by clicking on Reset-Button. Resets the variables and deletes the textfield.

Definition at line 1514 of file gui_control.cpp.

References help_status, iris_capture_flag, iris_ext_flag, m_cbButton1, m_cbButton2, m_cbButton3, m_cbButton4, m_cbButton5, m_cbButton6, m_cbButton7, m_cbButton8, m_csEdit1, and showHelpBalloons().

5.16.3.16 void gui_control::OnLButtonDown (UINT *nFlags*, CPoint *point*)

Close the preview-window of iris-feature-extraction and call the reactivation of the tooltips with the preview-variable. Click somewhere outsite the preview-window. For clicking inside the window the method previewClose is used.

Parameters:

* parameters set by the Mouse-Event

Definition at line 595 of file gui_control.cpp.

References preview.

5.16.3.17 void gui_control::OnBnClickedCheck1 ()

Method called by check or uncheck the Help-Box.Method enables or disables the tooltips.

Definition at line 1538 of file gui_control.cpp.

References help_on, m_cbCheck1, m_csEdit1, and showHelpBalloons().

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

- gui_control.h
- gui_control.cpp

5.17 Iris::IrisCode Class Reference

Public Member Functions

- IrisCode (char *irisCodeFile_, char *maskFile_, vigra::BImage img, char *imageName)
- int Quantisierung (std::complex< double > res, int i, bool fail)
- void saveIrisCodeToFile ()
- int gaborFilter (int r0, int phi0, int i, int frequNr)
- void newBuildIrisCode ()
- double bestIrisR (double phi, double bestRRight, double bestRLeft)

- double x_r_phi (double r, double phi, double bestR, double bestRLeft, double bestRRight, double bestX)
- double y_r_phi (double r, double phi, double bestR, double bestRLeft, double bestRRight, double bestY)
- double Phi (int bestX, int bestY, int x, int y, int bestR)
- double R (int bestX, int bestY, int x, int y, int bestR, int bestROut)
- bool isInCircle (int r, int x, int y, int x0, int y0)
- double bresenhamEliptic (int x0, int y0, int a, int b)
- void drawBresenhamIrisLeft (int R, int x0, int y0)
- void drawBresenhamIrisRight (int R, int x0, int y0)
- double bresenhamIrisRight (int R, int x0, int y0)
- double bresenhamIrisLeft (int R, int x0, int y0)
- double bresenhamPupil (int R, int x0, int y0)
- void drawBresenham (int R, int x0, int y0)
- void findPupil ()
- void findOuterBoundaryLeft ()
- void findOuterBoundaryRight ()
- void findEdgeLeft ()
- void drawIrisBoundaries ()
- void normalizeIris ()
- void drawIrisCode ()
- void printIrisCode ()
- void generateImage ()
- double bresenhamElipticAngleDOWN (int x0, int y0, int a, int b, double angle)
- double bresenhamElipticAngleUP (int x0, int y0, int a, int b, double angle)
- double drawBresenhamElipticAngleUP (int x0, int y0, int a, int b, double angle)
- double drawBresenhamElipticAngleDOWN (int x0, int y0, int a, int b, double angle)
- void findUpperBound ()
- void findLowerBound ()
- void drawUpperAndLowerBound ()
- void eyelidMaskUp (int x, int y)
- void eyelidMaskDown (int x, int y)
- void mask ()
- void generateMask ()
- void setMaskBits ()
- void saveMaskBitsToFile ()
- int * getIrisCode ()
- int * getMaskBits ()
- void findPupilArea ()
- void drawPupilSearchingArea ()
- void deleteArtefactsinPupil ()
- void deleteArtefactsinNormalisation ()
- void drawNormalisation ()
- void drawBresenhamLine (int x1, int y1, int x2, int y2, int **raw)
- void gaborFilter_preprocessing (double omega_, double alpha_, double beta_, int frequNr)

Public Attributes

- std::complex< double > *** preprocessing_exp_values
- double gabor_k
- double gabor_beta

5.17.1 Detailed Description

Definition at line 28 of file IrisCode.hpp.

5.17.2 Member Function Documentation

5.17.2.1 void Iris::IrisCode::saveIrisCodeToFile ()

Saves the derived iriscode bits to a .txt file.

Definition at line 166 of file IrisCode.cpp.

5.17.2.2 int Iris::IrisCode::gaborFilter (int *r0*, int *phi0*, int *i*, int *frequNr*)

Applies a gabor filter to the coordinates (r0, phi0) of the normalized iris.

Parameters:

r0 radial polar coordinate.

phi0 angular polar coordinate.

i index of the current bit.

frequNr number of frequence used

Returns:

bit-coded quantification of the filter output.

Definition at line 227 of file IrisCode.cpp.

References preprocessing_exp_values, and Quantisierung().

5.17.2.3 void Iris::IrisCode::newBuildIrisCode ()

Generates the iriscode for the normalized iris.

Definition at line 308 of file IrisCode.cpp.

References gaborFilter(), and gaborFilter_preprocessing().

5.17.2.4 double Iris::IrisCode::bestIrisR (double *phi*, double *bestRRight*, double *bestRLeft*)

Returns the radius of the iris for the angle phi.

Parameters:

phi angle in radian measure.
bestRRight the calculated radius of the right half of the iris.
bestRLeft the calculated radius of the left half of the iris.

Returns:

the radius of the iris for the angle phi.

Definition at line 410 of file IrisCode.cpp.

5.17.2.5 double Iris::IrisCode::x_r_phi (double *r*, double *phi*, double *bestR*, double *bestRLeft*, double *bestRRight*, double *bestX*)

Calculates the x-coordinate for the pixel defined by radius r and angle phi.

Parameters:

r radius in pixels.
phi angle in radian measure.
bestR radius of the pupil.
bestRLeft radius of the left half of the iris.
bestRRight radius of the right half of the iris.
bestX x-Coordinate of the pupil.

Returns:

x-coordinate for the pixel defined by radius r and angle phi.

Definition at line 429 of file IrisCode.cpp.

5.17.2.6 double Iris::IrisCode::y_r_phi (double *r*, double *phi*, double *bestR*, double *bestRLeft*, double *bestRRight*, double *bestY*)

Calculates the y-coordinate for the pixel defined by radius r and angle phi.

Parameters:

r radius in pixels.
phi angle in radian measure.
bestR radius of the pupil.
bestRLeft radius of the left half of the iris.
bestRRight radius of the right half of the iris.
bestY y-Coordinate of the pupil.

Returns:

y-coordinate for the pixel defined by radius r and angle phi.

Definition at line 459 of file IrisCode.cpp.

References bestIrisR().

5.17.2.7 double Iris::IrisCode::Phi (int *bestX*, int *bestY*, int *x*, int *y*, int *bestR*)

Calculates the angle of the straight line across the center of pupil and the coordinates (x, y).

Parameters:

bestX x-coordinate of the pupil.

bestY y-coordinate of the pupil.

x an x-coordinate.

y an y-coordinate.

bestR radius of the pupil.

Returns:

angle of the straight line across the center of pupil and the coordinates (x, y).

Definition at line 489 of file IrisCode.cpp.

5.17.2.8 double Iris::IrisCode::R (int *bestX*, int *bestY*, int *x*, int *y*, int *bestR*, int *bestROut*)

Calculates the normalized distance between the outer boundary of the pupil and the coordinates (x, y).

Parameters:

bestX x-coordinate of the pupil.

bestY y-coordinate of the pupil.

x an x-coordinate.

y an y-coordinate.

bestR radius of the pupil.

bestROut radius of the iris.

Returns:

normalized distance between the outer boundary of the pupil and the coordinates (x, y).

Definition at line 519 of file IrisCode.cpp.

5.17.2.9 bool Iris::IrisCode::isInCircle (int *r*, int *x*, int *y*, int *x0*, int *y0*)

Calculates if coordinates (x, y) lie in the circle defined by the coordinates (x0, y0) and the radius r.

Parameters:

r radius in pixels.

x an x-coordinate.

y an y-coordinate.

x0 x-coordinate of the center of circle.

y0 y-coordinate of the center of circle.

Returns:

'true' if the coordinates are in the circle, else 'false'.

Definition at line 548 of file IrisCode.cpp.

5.17.2.10 void Iris::IrisCode::drawBresenhamIrisLeft (int *R*, int *x0*, int *y0*)

Sets the pixels of the left half of the circle, defined by center (x0, y0) and radius R to white. Image ist stored in the array rgbValuesOut.

Parameters:

R radius of the circle.

x0 x-coordinate of the circle center.

y0 y-coordinate of the circle center

Definition at line 625 of file IrisCode.cpp.

5.17.2.11 void Iris::IrisCode::drawBresenhamIrisRight (int *R*, int *x0*, int *y0*)

Sets the pixels of the right half of the circle, defined by center (x0, y0) and radius R to white. Image ist stored in the array rgbValuesOut.

Parameters:

R radius of the circle.

x0 x-coordinate of the circle center.

y0 y-coordinate of the circle center

Definition at line 669 of file IrisCode.cpp.

5.17.2.12 void Iris::IrisCode::drawBresenham (int *R*, int *x0*, int *y0*)

Sets the pixels of the circle, defined by center (x0, y0) and radius R to white. Image ist stored in the array rgbValuesOut.

Parameters:

R radius of the circle.

x0 x-coordinate of the circle center.

y0 y-coordinate of the circle center

Definition at line 866 of file IrisCode.cpp.

5.17.2.13 void Iris::IrisCode::findPupil ()

Determines the coordinates of the center of the pupil and the radius of the pupil. Saves the values to bestX, bestY and bestR.

Definition at line 907 of file IrisCode.cpp.

References bresenhamPupil().

5.17.2.14 void Iris::IrisCode::findOuterBoundaryLeft ()

Determines radius of left half of the iris. Saves the value to bestRLeft.

Definition at line 982 of file IrisCode.cpp.

References bresenhamIrisLeft().

5.17.2.15 void Iris::IrisCode::findOuterBoundaryRight ()

Determines radius of right half of the iris. Saves the value to bestRRight.

Definition at line 1089 of file IrisCode.cpp.

References bresenhamIrisRight().

5.17.2.16 void Iris::IrisCode::drawIrisBoundaries ()

Draws the determined circles of pupil, left and right iris boundaries to the image Iris.bmp.

Definition at line 1122 of file IrisCode.cpp.

References drawBresenham(), drawBresenhamIrisLeft(), and drawBresenhamIrisRight().

5.17.2.17 void Iris::IrisCode::normalizeIris ()

Transforms the localized iris into a normalized representation (pseudo-polar coordinates). Saves the image of the normalized Iris to Normalization.bmp.

Definition at line 1141 of file IrisCode.cpp.

References x_r_phi(), and y_r_phi().

5.17.2.18 void Iris::IrisCode::drawIrisCode ()

Draws a graphical representation of the derived iriscode and saves it to IrisCode.bmp.

Definition at line 1282 of file IrisCode.cpp.

5.17.2.19 void Iris::IrisCode::printIrisCode ()

Command line output of the iriscode.

Definition at line 1317 of file IrisCode.cpp.

5.17.2.20 void Iris::IrisCode::generateImage ()

Transforms the current content of rgbValuesOut to an image and saves it to Iris.bmp.

Definition at line 1328 of file IrisCode.cpp.

5.17.2.21 void Iris::IrisCode::findUpperBound ()

Detects the upper boundary of the eyelid.

Definition at line 2965 of file IrisCode.cpp.

References bresenhamElipticAngleUP().

5.17.2.22 void Iris::IrisCode::findLowerBound ()

Detects the lower boundary of the eyelid.

Definition at line 2866 of file IrisCode.cpp.

References bresenhamElipticAngleDOWN().

5.17.2.23 void Iris::IrisCode::drawUpperAndLowerBound ()

Draws the upper and lower boundary of the eyelid to the output image.

Definition at line 3066 of file IrisCode.cpp.

References drawBresenhamElipticAngleDOWN(), and drawBresenhamElipticAngleUP().

5.17.2.24 void Iris::IrisCode::eyelidMaskUp (int *x*, int *y*)

Sets the black mask for the upper eyelid coordinate (x, y).

Parameters:

- ***x*** an x-coordinate of the upper eyelid.
- ***y*** an y-coordinate of the upper eyelid.

Definition at line 3077 of file IrisCode.cpp.

5.17.2.25 void Iris::IrisCode::eyelidMaskDown (int *x*, int *y*)

Sets the black mask for the lower eyelid coordinate (x, y).

Parameters:

- ***x*** an x-coordinate of the lower eyelid.
- ***y*** an y-coordinate of the lower eyelid.

Definition at line 3092 of file IrisCode.cpp.

5.17.2.26 void Iris::IrisCode::mask ()

Sets the color values for the mask array. Masked coordinates are set to black, valid coordinates are set to white.

Definition at line 3104 of file IrisCode.cpp.

5.17.2.27 void Iris::IrisCode::generateMask ()

Generates a normalized representation of the mask array. Saves the image of the normalized mask to Mask.bmp.

Definition at line 3118 of file IrisCode.cpp.

References mask(), x_r_phi(), and y_r_phi().

5.17.2.28 void Iris::IrisCode::setMaskBits ()

Sets the bits of the maskBits array.

Definition at line 3151 of file IrisCode.cpp.

5.17.2.29 void Iris::IrisCode::saveMaskBitsToFile ()

Saves the mask bits to a .txt file.

Definition at line 3174 of file IrisCode.cpp.

5.17.2.30 void Iris::IrisCode::findPupilArea ()

searches for the pupil area

Definition at line 3199 of file IrisCode.cpp.

5.17.2.31 void Iris::IrisCode::drawPupilSearchingArea ()

draws the pupil area in image

Definition at line 3278 of file IrisCode.cpp.

5.17.2.32 void Iris::IrisCode::deleteArtefactsinPupil ()

deletes the artificats in pupil

Definition at line 3302 of file IrisCode.cpp.

5.17.2.33 void Iris::IrisCode::deleteArtefactsinNormalisation ()

Deletes artifacts in nomalisation

Definition at line 1180 of file IrisCode.cpp.

5.17.2.34 void Iris::IrisCode::drawNormalisation ()

Draws a graphical representation of the derived iriscode and saves it to IrisCode.bmp.

Definition at line 1245 of file IrisCode.cpp.

5.17.2.35 void Iris::IrisCode::gaborFilter_preprocessing (double *omega_*, double *alpha_*, double *beta_*, int *frequNr*)

Saves the results of the gabor function in an array.

Parameters:

omega_ frequence of the gabor filter.

alpha_ horizontal scale of the gabor filter.

beta_ vertical scale of the gabor filter.

frequNr number of frequence used

Definition at line 186 of file IrisCode.cpp.

References preprocessing_exp_values.

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

- IrisCode.hpp
- IrisCode.cpp

5.18 Iris::IrisCodeMatcher Class Reference

Public Member Functions

- IrisCodeMatcher (char *IrisCodeFile1, char *IrisCodeFile2, char *maskBitsFile1, char *maskBitsFile2, int nOB)
- IrisCodeMatcher (char *IrisCodeFile1, char *IrisCodeFile2, char *maskBitsFile1, char *maskBitsFile2)
- void readIrisCodeFromFile ()
- void readMaskBitsFromFile ()
- void calcHammingDistance ()
- IrisCodeMatcher (char *IrisCodeFile1, char *IrisCodeFile2, int nOB)
- IrisCodeMatcher (char *IrisCodeFile1, char *IrisCodeFile2)
- double calcHammingDistance (int shift)
- void HammingDistance ()

5.18.1 Detailed Description

Definition at line 11 of file IrisCodeGenerator/Matcher/IrisCodeMatcher.hpp.

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

- IrisCodeGenerator/Matcher/IrisCodeMatcher.hpp
- IrisCodeMatcher/IrisCodeMatcher.hpp
- IrisCodeMatcher.hpp
- NewIrisCodeMatcher/IrisCodeMatcher.hpp
- IrisCodeGenerator/Matcher/IrisCodeMatcher.cpp
- IrisCodeMatcher/IrisCodeMatcher.cpp
- NewIrisCodeMatcher/IrisCodeMatcher.cpp

5.19 matching_result_item_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- template_id_t id
- score_t score

5.19.1 Detailed Description

Structure that holds a single result of a template matching.

Definition at line 250 of file csi_struct.h.

5.19.2 Member Data Documentation

5.19.2.1 template_id_t matching_result_item_t::id

The template ID of the result.

Definition at line 251 of file csi_struct.h.

5.19.2.2 score_t matching_result_item_t::score

The matching score of the result.

Definition at line 252 of file csi_struct.h.

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

- csi_struct.h

5.20 matching_result_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- matching_result_item_t results [MAX_MATCHING_RESULTS]
- int length

5.20.1 Detailed Description

Type definition of a structure to save the results of each node while matching.

Definition at line 258 of file csi_struct.h.

5.20.2 Member Data Documentation

5.20.2.1 matching_result_item_t matching_result_t::results[MAX_MATCHING_RESULTS]

A list of results with the maximum size of `MAX_MATCHING_RESULTS`.

Definition at line 259 of file csi_struct.h.

5.20.2.2 int matching_result_t::length

Number of results in the resultslist.

Definition at line 260 of file csi_struct.h.

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

- csi_struct.h

5.21 minutia_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- coordinate_t coordinate
- angle_t angle
- minutia_type_t type

5.21.1 Detailed Description

Structure for a single minutia point.

Definition at line 96 of file csi_struct.h.

5.21.2 Member Data Documentation

5.21.2.1 coordinate_t minutia_t::coordinate

coordinate of the minutia

Definition at line 97 of file csi_struct.h.

5.21.2.2 angle_t minutia_t::angle

angle of the minutia

Definition at line 98 of file csi_struct.h.

5.21.2.3 minutia_type_t minutia_t::type

type of the minutia

Definition at line 99 of file csi_struct.h.

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

- csi_struct.h

5.22 template_cluster_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- template_id_t id
- angle_t orientation_vector [ORIENTATION_VECTOR_SIZE]
- uint8_t orientation_vector_mask [ORIENTATION_VECTOR_SIZE]
- ard_vector_t ard_vector
- uint32_t cluster_id
- uint16_t ard_bin

5.22.1 Detailed Description

Structure for a template for the fingerprint clustering matching algorithm.

Definition at line 44 of file csi_struct.h.

5.22.2 Member Data Documentation

5.22.2.1 template_id_t template_cluster_t::id

templates ID

Definition at line 45 of file csi_struct.h.

5.22.2.2 angle_t template_cluster_t::orientation_vector[ORIENTATION_VECTOR_SIZE]

the orientation vector

Definition at line 46 of file csi_struct.h.

5.22.2.3 uint8_t template_cluster_t::orientation_vector_mask[ORIENTATION_VECTOR_SIZE]

the orientation vectors mask

Definition at line 47 of file csi_struct.h.

5.22.2.4 ard_vector_t template_cluster_t::ard_vector

average distance vector

Definition at line 48 of file csi_struct.h.

5.22.2.5 uint32_t template_cluster_t::cluster_id

clusters id

Definition at line 49 of file csi_struct.h.

5.22.2.6 uint16_t template_cluster_t::ard_bin

bin where the template gets sorted in

Definition at line 50 of file csi_struct.h.

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

- csi_struct.h

5.23 template_filterbank_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- template_id_t id
- feature_map_t map [NUM_SECTORS]

5.23.1 Detailed Description

Structure for a template for the filterbank based matching algorithm.

Definition at line 127 of file csi_struct.h.

5.23.2 Member Data Documentation

5.23.2.1 template_id_t template_filterbank_t::id

template id

Definition at line 128 of file csi_struct.h.

5.23.2.2 feature_map_t template_filterbank_t::map[NUM_SECTORS]

feature map

Definition at line 129 of file csi_struct.h.

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

- csi_struct.h

5.24 template_iris_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- char type
- template_id_t id
- iris_code_t code
- iris_mask_t mask

5.24.1 Detailed Description

Structure for an iris template.

Definition at line 148 of file csi_struct.h.

5.24.2 Member Data Documentation

5.24.2.1 char template_iris_t::type

type of tmeplate

Definition at line 149 of file csi_struct.h.

5.24.2.2 template_id_t template_iris_t::id

templates id

Definition at line 150 of file csi_struct.h.

5.24.2.3 iris_code_t template_iris_t::code

iris code

Definition at line 151 of file csi_struct.h.

5.24.2.4 iris_mask_t template_iris_t::mask

iris code mask

Definition at line 152 of file csi_struct.h.

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

- csi_struct.h

5.25 template_minutiae_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- template_id_t id
- core_t core
- minutia_t minutiae [MAX_NUM_MINUTIAE]
- int32_t minutiae_length

5.25.1 Detailed Description

Structure for a template for the fingerprint clustering matching algorithm.

Definition at line 105 of file csi_struct.h.

5.25.2 Member Data Documentation

5.25.2.1 template_id_t template_minutiae_t::id

the id of the template

Definition at line 106 of file csi_struct.h.

5.25.2.2 core_t template_minutiae_t::core

the core of the fingerprint

Definition at line 107 of file csi_struct.h.

5.25.2.3 minutia_t template_minutiae_t::minutiae[MAX_NUM_MINUTIAE]

an array of minuta_t points

Definition at line 108 of file csi_struct.h.

5.25.2.4 int32_t template_minutiae_t::minutiae_length

the number of minutae points in the minutiae[] array

Definition at line 109 of file csi_struct.h.

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

- csi_struct.h

5.26 templates_fingerprint_t Struct Reference

`#include <csi_struct.h>`

Public Attributes

- template_cluster_t template_cluster
- template_minutiae_t template_minutiae
- template_filterbank_t template_filterbank

5.26.1 Detailed Description

Structure for the fingerprint templates.

Definition at line 135 of file csi_struct.h.

5.26.2 Member Data Documentation

5.26.2.1 template_cluster_t templates_fingerprint_t::template_cluster

related cluster template

Definition at line 136 of file csi_struct.h.

5.26.2.2 template_minutiae_t templates_fingerprint_t::template_minutiae

related minutia template

Definition at line 137 of file csi_struct.h.

5.26.2.3 template_filterbank_t templates_fingerprint_t::template_filterbank

related filterbank template

Definition at line 138 of file csi_struct.h.

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

- csi_struct.h

5.27 triple_db Struct Reference

`#include <csi_struct.h>`

Public Attributes

- database_managment mng_minutia
- database_managment mng_filter
- database_managment mng_cluster

5.27.1 Detailed Description

Structure for a triple fingerprint db.

Definition at line 166 of file csi_struct.h.

5.27.2 Member Data Documentation

5.27.2.1 database_managment triple_db::mng_minutia

Database management for minutiae templates.

Definition at line 167 of file csi_struct.h.

5.27.2.2 database_managment triple_db::mng_filter

Database management for filterbank templates.

Definition at line 168 of file csi_struct.h.

5.27.2.3 database_managment triple_db::mng_cluster

Database management for cluster templates.

Definition at line 169 of file csi_struct.h.

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

- csi_struct.h

6 CSI:PC2 — Frontend File Documentation

6.1 Capture.cpp File Reference

Implementation file to capture a video stream from the camera, preview it on the screen and save it as a jpg and bmp.

```
#include "StdAfx.h"
#include "Capture.h"
```

Functions

- void on_mouse (int event, int x, int y, int flags, void *param)

6.1.1 Detailed Description

Implementation file to capture a video stream from the camera, preview it on the screen and save it as a jpg and bmp.

Author:

Klaus Herbold <klaus.herbold@csipc2.de>

Definition in file Capture.cpp.

6.1.2 Function Documentation

6.1.2.1 void on_mouse (int *event*, int *x*, int *y*, int *flags*, void * *param*)

Callbackfunction to close the window with the captured image here, only the first paramter is nedded, the others are unneccessary

Parameters:

event what kind of event is triggered, 1 for left mouse click

x horizontal coordinate of the mouse

y vertical coordinate of the mouse

flags a combination of some events like CTRL-key is pressed

**param* user defined parameter passed to the callback function

Definition at line 20 of file Capture.cpp.

6.2 Capture.h File Reference

Header file to capture a video stream from the camera, preview it on the screen and save it as a jpg and bmp.

```
#include <dshow.h>
#include <cv.h>
#include <highgui.h>
#include "cltifa_apix1.h"
```

Classes

- class Capture

Defines

- #define SAFE_RELEASE(x) { if (x) x → Release(); x = NULL; }

6.2.1 Detailed Description

Header file to capture a video stream from the camera, preview it on the screen and save it as a jpg and bmp.

Author:

Klaus Herbold <klaus.herbold@csipc2.de>

Definition in file Capture.h.

6.2.2 Define Documentation

6.2.2.1 #define SAFE_RELEASE(x) { if (x) x → Release(); x = NULL; }

Macro to release the Direct Show elements

Definition at line 19 of file Capture.h.

6.3 csi_config.h File Reference

This file contains common configurations used by several parts of the CSI:PC^2 project.

Defines

- #define MAX_LENGTH_FILENAME 255
- #define MAX_MATCHING_RESULTS 10
- #define MAX_NUM_MINUTIAE 70
- #define NUM_SECTORS 8
- #define NUM_RINGS 5
- #define ORIENTATION_VECTOR_SIZE 156

6.3.1 Detailed Description

This file contains common configurations used by several parts of the CSI:PC^2 project.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_config.h.

6.3.2 Define Documentation

6.3.2.1 #define MAX_LENGTH_FILENAME 255

The maximum length of a filename (useful for allocation and checks).

TODO: check how to integrate this into autotools to make this OS independant.

Definition at line 18 of file csi_config.h.

6.3.2.2 #define MAX_MATCHING_RESULTS 10

This constant specifies the maximum number of matching results a single client should return.

Definition at line 24 of file csi_config.h.

6.3.2.3 #define MAX_NUM_MINUTIAE 70

Maximum number of minutiae points in a fingerprint template

Definition at line 29 of file csi_config.h.

6.3.2.4 #define NUM_RINGS 5

Number of rings used by the filterbank based fingerprint matching algorithm.

Definition at line 39 of file csi_config.h.

6.3.2.5 #define NUM_SECTORS 8

Number of sectors used by the filterbank based fingerprint matching algorithm.

Definition at line 34 of file csi_config.h.

6.3.2.6 #define ORIENTATION_VECTOR_SIZE 156

Size of the orientation Vector (how many regions of the circular tesselation are stored)

Definition at line 44 of file csi_config.h.

6.4 csi_math.c File Reference

This file contains math functions specifically designed for this project.

Functions

- int maxi (int a, int b)
- int mini (int a, int b)

6.4.1 Detailed Description

This file contains math functions specifically designed for this project.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_math.c.

6.4.2 Function Documentation

6.4.2.1 int maxi (int *a*, int *b*) [inline]

Returns the maximum of the two supplied integer values.

Parameters:

a an integer value

b an integer value.

Returns:

the absolute bigger value of the two.

Definition at line 18 of file csi_math.c.

6.4.2.2 int mini (int *a*, int *b*) [inline]

Returns the minimum of the two supplied integer values.

Parameters:

a an integer value

b an integer value.

Returns:

the absolute smaller value of the two.

Definition at line 31 of file csi_math.c.

6.5 csi_math.h File Reference

Header file for the math library. All external functions are exposed here.

Functions

- int maxi (int a, int b)
- int mini (int a, int b)

6.5.1 Detailed Description

Header file for the math library. All external functions are exposed here.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_math.h.

6.5.2 Function Documentation

6.5.2.1 int maxi (int *a*, int *b*) [inline]

Returns the maximum of the two supplied integer values.

Parameters:

a an integer value

b an integer value.

Returns:

the absolute bigger value of the two.

Definition at line 18 of file csi_math.c.

6.5.2.2 int mini (int *a*, int *b*) [inline]

Returns the minimum of the two supplied integer values.

Parameters:

a an integer value

b an integer value.

Returns:

the absolute smaller value of the two.

Definition at line 31 of file csi_math.c.

6.6 csi_results.c File Reference

This file contains utility functions to process matching results. Results are represented by the matching_result_t structure which is a sorted list of template IDs and their corresponding score.

```
#include "csi_struct.h"
```

```
#include "csi_results.h"
```

Functions

- void insert_new_result (matching_result_t *results, matching_result_item_t *new_result)

6.6.1 Detailed Description

This file contains utility functions to process matching results. Results are represented by the matching_result_t structure which is a sorted list of template IDs and their corresponding score.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_results.c.

6.6.2 Function Documentation

6.6.2.1 void insert_new_result (matching_result_t * *results*, matching_result_item_t * *new_result*)

Inserts the specified matching_result_t into the list of results. The result list's elements are sorted in ascending order, i.e. the element with the highest score is at index 0.

Parameters:

results a pointer to the matching_result_t.

new_result the new result to insert into the specified list of matching results.

Definition at line 26 of file csi_results.c.

6.7 csi_results.h File Reference

Header file that defines some utility functions to process matching results. Results are represented by the matching_result_t structure which is a sorted list of template IDs and their corresponding score.

```
#include "csi_struct.h"
```

Functions

- void insert_new_result (matching_result_t *results, matching_result_item_t *new_result)

6.7.1 Detailed Description

Header file that defines some utility functions to process matching results. Results are represented by the `matching_result_t` structure which is a sorted list of template IDs and their corresponding score.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_results.h.

6.7.2 Function Documentation

6.7.2.1 void insert_new_result (matching_result_t * *results*, matching_result_item_t * *new_result*)

Inserts the specified matching_result_t into the list of results. The result list's elements are sorted in ascending order, i.e. the element with the highest score is at index 0.

Parameters:

results a pointer to the matching_result_t.

new_result the new result to insert into the specified list of matching results.

Definition at line 26 of file csi_results.c.

References MAX_MATCHING_RESULTS, matching_result_t::results, and matching_result_item_t::score.

6.8 csi_serialization.c File Reference

Utility functions to serialize and deserialize data structures.

```
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stddef.h>
#include "csi_serialization.h"
```

Functions

- int write_fingerprint_template (const char *path, templates_fingerprint_t *templates, const int count)
- int read_fingerprint_template (const char *path, templates_fingerprint_t *templates, const int count)

6.8.1 Detailed Description

Utility functions to serialize and deserialize data structures.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_serialization.c.

6.8.2 Function Documentation

6.8.2.1 int read_fingerprint_template (const char * *path*, templates_fingerprint_t * *templates*, const int *count*)

De-Serializes the fingerprint templates data structure.

Parameters:

path the file to read from.

templates the templates data structures to fill, must be initialized.

count the number of templates.

Returns:

0 if everything works fine, otherwise a non-zero value using standard error codes.

Definition at line 65 of file csi_serialization.c.

6.8.2.2 int write_fingerprint_template (const char * *path*, templates_fingerprint_t * *templates*, const int *count*)

Serializes the fingerprint templates data structure.

Parameters:

path the file to serialize to.

templates the template data structures to serialize.

count the number of templates.

Returns:

0 if everything works fine, otherwise a non-zero value using standard error codes.

Definition at line 27 of file csi_serialization.c.

6.9 csi_serialization.h File Reference

Header file for the utility functions to serialize and deserialize data structures.

```
#include "csi_struct.h"
```

Functions

- int write_fingerprint_template (const char *path, templates_fingerprint_t *templates, const int count)
- int read_fingerprint_template (const char *path, templates_fingerprint_t *templates, const int count)

6.9.1 Detailed Description

Header file for the utility functions to serialize and deserialize data structures.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_serialization.h.

6.9.2 Function Documentation

6.9.2.1 int read_fingerprint_template (const char * *path*, templates_fingerprint_t * *templates*, const int *count*)

De-Serializes the fingerprint templates data structure.

Parameters:

path the file to read from.

templates the templates data structures to fill, must be initialized.

count the number of templates.

Returns:

0 if everything works fine, otherwise a non-zero value using standard error codes.

Definition at line 65 of file csi_serialization.c.

6.9.2.2 int write_fingerprint_template (const char * *path*, templates_fingerprint_t * *templates*, const int *count*)

Serializes the fingerprint templates data structure.

Parameters:

path the file to serialize to.

*templates the template data structures to serialize.

count the number of templates.

Returns:

0 if everything works fine, otherwise a non-zero value using standard error codes.

Definition at line 27 of file csi_serialization.c.

6.10 csi_struct.h File Reference

This file contains common structures used by several parts of the CSI:PC2 project.

```
#include <stdint.h>
#include <stdio.h>
#include "csi_config.h"
```

Classes

- struct template_cluster_t
- struct cluster_centroid_t
- struct cluster_t
- struct coordinate_t
- struct minutia_t
- struct template_minutiae_t
- struct feature_map_t
- struct template_filterbank_t
- struct templates_fingerprint_t
- struct template_iris_t
- struct database_managment
- struct triple_db
- struct db_init
- struct frontend_iris_data_t
- struct frontend_finger_data_t
- union frontend_data_t
- struct matching_result_item_t
- struct matching_result_t

Defines

- #define BUFFSIZE 100

Typedefs

- typedef float score_t
- typedef uint32_t template_id_t
- typedef uint16_t angle_t
- typedef unsigned char ard_vector_t
- typedef double ov_dist_t
- typedef uint16_t pixel_t
- typedef coordinate_t core_t
- typedef unsigned char gray_value_t
- typedef uint64_t iris_vector_t [32]
- typedef iris_vector_t iris_code_t
- typedef iris_vector_t iris_mask_t
- typedef struct db_init db_init_t

Enumerations

- enum minutia_type_t { CSI_BIFURCATION = 0, CSI_RIDGE_ENDING }
- enum jobmode_t {
  JOB_NULL = 1, JOB_STATUS, JOB_MATCH_FINGER, JOB_MATCH_-FINGER_MINUTIAE,
  JOB_MATCH_FINGER_CLUSTER, JOB_MATCH_FINGER_-FILTERBANK, JOB_MATCH_FINGER_MIXED, JOB_MATCH_IRIS,
  JOB_SAVE_FINGER, JOB_SAVE_IRIS, JOB_SHUTDOWN, JOB_-EXTRACT_ALL }
- enum extractmode_t {
  EXTRACT_FALSE = 1, EXTRACT_TRUE, EXTRACT_TST_RAW, EXTRACT_TST_BIN,
  EXTRACT_SFINGE }

6.10.1 Detailed Description

This file contains common structures used by several parts of the CSI:PC2 project.

Author:

Christoph Konersmann <christoph.konersmann@csipc2.de>
Dominic Eschweiler <Dominic.Eschweiler@csipc2.de>
Elmar Weber <university@elmarweber.org>

Definition in file csi_struct.h.

6.10.2 Define Documentation

6.10.2.1 #define BUFFSIZE 100

Definition of the buffersize for an amount of templates. Buffersize for the database

Definition at line 30 of file csi_struct.h.

6.10.3 Typedef Documentation

6.10.3.1 typedef uint16_t angle_t

Datatype for the representation of an angle between 0 and 360 without any decimals.

Definition at line 36 of file csi_struct.h.

6.10.3.2 typedef unsigned char ard_vector_t

Datatype for the ARD vector. Used only by the clustering algorithm.

Definition at line 38 of file csi_struct.h.

6.10.3.3 typedef coordinate_t core_t

Datatype for the core of a fingerprint. coordinate of the corepoint

Definition at line 83 of file csi_struct.h.

6.10.3.4 typedef struct db_init db_init_t

Structure for a initialising the database. Structure for a initialising the database.

6.10.3.5 typedef unsigned char gray_value_t

Datatype that represents a gray value (8bits, 256 gray values). Datatype that represents a gray value (8bits, 256 gray values).

Definition at line 115 of file csi_struct.h.

6.10.3.6 typedef iris_vector_t iris_code_t

iris code

Definition at line 143 of file csi_struct.h.

6.10.3.7 typedef iris_vector_t iris_mask_t

iris mask

Definition at line 144 of file csi_struct.h.

6.10.3.8 typedef uint64_t iris_vector_t[32]

iris vector

Definition at line 142 of file csi_struct.h.

6.10.3.9 typedef double ov_dist_t

Datatype for the distance between to orientation vectors

Definition at line 40 of file csi_struct.h.

6.10.3.10 typedef uint16_t pixel_t

Datatype for the representation of a pixel (as in bitmap).

Definition at line 70 of file csi_struct.h.

6.10.3.11 typedef float score_t

Datatype used by matching scores.

Definition at line 32 of file csi_struct.h.

6.10.3.12 typedef uint32_t template_id_t

Datatype for the id of a template.

Definition at line 34 of file csi_struct.h.

6.10.4 Enumeration Type Documentation

6.10.4.1 enum extractmode_t

Enumerate different modes of extraction

Enumerator:

EXTRACT_FALSE Do not extract.

EXTRACT_TRUE Extract templates with default mode.

EXTRACT_TST_RAW Extract templates with mode TST_RAW.

EXTRACT_TST_BIN Extract templates with mode TST_BIN.

EXTRACT_SFINGE Extract templates with mode TST_SFINGE.

Definition at line 205 of file csi_struct.h.

6.10.4.2 enum jobmode_t

Enumerate the jobs which could be done by the CCD.

Enumerator:

JOB_NULL The dummy job ID, used to test the connection.

JOB_STATUS Job ID for getting the CCD status.

JOB_MATCH_FINGER Job ID for default fingerprint matching.

JOB_MATCH_FINGER_MINUTIAE Job ID for minutiae based fingerprint matching.

JOB_MATCH_FINGER_CLUSTER Job ID for clustering based fingerprint matching.

JOB_MATCH_FINGER_FILTERBANK Job ID for filterbank based fingerprint matching.

JOB_MATCH_FINGER_MIXED Job ID for special mixed fingerprint matching.

JOB_MATCH_IRIS Job ID for iriscode matching.

JOB_SAVE_FINGER Job ID for saving iris data.

JOB_SAVE_IRIS Job ID for saving fingerprint data.

JOB_SHUTDOWN Job ID for shutdown.

JOB_EXTRACT_ALL Extract data of all images of a database.

Definition at line 187 of file csi_struct.h.

6.10.4.3 enum minutia_type_t

Datatype for the type of a minutia point.

Enumerator:

CSI_BIFURCATION Bifurcation

CSI_RIDGE_ENDING Ending

Definition at line 88 of file csi_struct.h.

6.11 csi_time.c File Reference

CSI utility functions to measure time for benchmarking. It supports to report the overall time and the current time of a timer. Furthermore several functions help with calculating the time differences.

```
#include <stdlib.h>
#include <stdio.h>
#include <sys/time.h>
#include "csi_time.h"
```

Functions

- csi_timer_t * free_timer (csi_timer_t *timer)
- long diff_s (struct timeval *start, struct timeval *stop)
- long diff_ms (struct timeval *start, struct timeval *stop)
- long diff_us (struct timeval *start, struct timeval *stop)
- csi_timer_t * create_timer ()
- void start_timer (csi_timer_t *timer)
- void stop_timer (csi_timer_t *timer)
- long get_current_ms (csi_timer_t *timer)
- long get_current_us (csi_timer_t *timer)

- long get_diff_s (csi_timer_t *timer)
- long get_diff_ms (csi_timer_t *timer)
- long get_diff_us (csi_timer_t *timer)
- char * to_time_str (csi_timer_t *timer)
- void printf_current_time (char *prefix, csi_timer_t *timer)
- void printf_diff_time (char *prefix, csi_timer_t *timer)

6.11.1 Detailed Description

CSI utility functions to measure time for benchmarking. It supports to report the overall time and the current time of a timer. Furthermore several functions help with calculating the time differences.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_time.c.

6.11.2 Function Documentation

6.11.2.1 csi_timer_t* create_timer ()

Creates and allocates a new csi_timer_t.

Returns:

an allocated csi_timer_c, NULL if the allocation fails.

Definition at line 32 of file csi_time.c.

6.11.2.2 csi_timer_t * free_timer (csi_timer_t * *timer*)

Frees the specified timer.

Parameters:

**timer* the timer to free.

Returns:

a NULL pointer.

Definition at line 61 of file csi_time.c.

References csi_timer_t::start, and csi_timer_t::stop.

6.11.2.3 long get_current_ms (csi_timer_t * *timer*)

Calculates the milliseconds the timer is running, i.e. the time that passed since start_timer has been called.

Parameters:

timer the timer to use.

Returns:

the milliseconds since the start of the timer.

Definition at line 122 of file csi_time.c.

6.11.2.4 long get_current_us (csi_timer_t * *timer*)

Calculates the microseconds the timer is running, i.e. the time that passed since start_timer has been called.

Parameters:

timer the timer to use.

Returns:

the microseconds since the start of the timer.

Definition at line 138 of file csi_time.c.

6.11.2.5 long get_diff_ms (csi_timer_t * *timer*)

Calculates the milliseconds between the start and stop of the specified timer. Please be aware that if either one of the start_timer or stop_timer method has not been called this function will produce an undefined behaviour.

Parameters:

timer the timer to use.

Returns:

the milliseconds between the start and stop of the timer.

Definition at line 170 of file csi_time.c.

6.11.2.6 long get_diff_s (csi_timer_t * *timer*)

Calculates the seconds between the start and stop of the specified timer. Please be aware that if either one of the start_timer or stop_timer method has not been called this function will produce an undefined behaviour.

Parameters:

∗***timer*** the timer to use.

Returns:

the seconds between the start and stop of the timer.

Definition at line 156 of file csi_time.c.

6.11.2.7 long get_diff_us (csi_timer_t ∗ *timer*)

Calculates the microseconds between the start and stop of the specified timer. Please be aware that if either one of the start_timer or stop_timer method has not been called this function will produce an undefined behaviour.

Parameters:

∗***timer*** the timer to use.

Returns:

the microseconds between the start and stop of the timer.

Definition at line 185 of file csi_time.c.

6.11.2.8 void printf_current_time (char ∗ *prefix*, csi_timer_t ∗ *timer*)

Prints the current time of the timer to stdout (`prefix`: minutes:seconds.milliseconds).

Parameters:

∗***prefix*** the prefix to print before the timer output.

∗***timer*** the timer.

Definition at line 215 of file csi_time.c.

6.11.2.9 void printf_diff_time (char ∗ *prefix*, csi_timer_t ∗ *timer*)

Prints the difference time of the timer (end - start) to stdout (`prefix`: minutes:seconds.milliseconds).

Parameters:

∗***prefix*** the prefix to print before the timer output.

∗***timer*** the timer.

Definition at line 230 of file csi_time.c.

6.11.2.10 void start_timer (csi_timer_t * *timer*)

Starts the specified timer.

Parameters:

**timer* the timer to use.

Definition at line 82 of file csi_time.c.

6.11.2.11 void stop_timer (csi_timer_t * *timer*)

Stops the specified timer.

Parameters:

**timer* the timer to use.

Definition at line 92 of file csi_time.c.

6.11.2.12 char* to_time_str (csi_timer_t * *timer*)

Creates a string representation as seconds.milliseconds of the specified csi_timer_t. Please note that the caller must free the result.

Parameters:

**timer* the timer.

Returns:

a string seconds:milliseconds.

Definition at line 198 of file csi_time.c.

6.12 csi_time.h File Reference

Header file for CSI utility functions to measure time for benchmarking. It supports to report the overall time and the current time of a timer. Furthermore several functions help with calculating the time differences.

```
#include <sys/time.h>
```

Classes

- struct csi_timer_t

Functions

- csi_timer_t * create_timer ()
- csi_timer_t * free_timer (csi_timer_t *timer)
- void start_timer (csi_timer_t *timer)
- void stop_timer (csi_timer_t *timer)
- long get_current_ms (csi_timer_t *timer)
- long get_current_us (csi_timer_t *timer)
- long get_diff_s (csi_timer_t *timer)
- long get_diff_ms (csi_timer_t *timer)
- long get_diff_us (csi_timer_t *timer)
- char * to_time_str (csi_timer_t *timer)
- void printf_current_time (char *prefix, csi_timer_t *timer)
- void printf_diff_time (char *prefix, csi_timer_t *timer)

6.12.1 Detailed Description

Header file for CSI utility functions to measure time for benchmarking. It supports to report the overall time and the current time of a timer. Furthermore several functions help with calculating the time differences.

An example use could look like this:

```
csi_timer_t *timer;

timer = create_timer();

start_timer(timer);

// do your thing printf("timer is current running:  %ld
ms\n", get_current_ms(timer));

stop_timer(timer);

printf("milliseconds:  %ld\n", get_diff_ms(timer));
printf("microseconds:  %ld\n", get_diff_us(timer));

free_timer(timer);
```

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_time.h.

6.12.2 Function Documentation

6.12.2.1 csi_timer_t* create_timer ()

Creates and allocates a new csi_timer_t.

Returns:

an allocated csi_timer_c, NULL if the allocation fails.

Definition at line 32 of file csi_time.c.

References free_timer(), csi_timer_t::start, and csi_timer_t::stop.

6.12.2.2 csi_timer_t* free_timer (csi_timer_t * *timer*)

Frees the specified timer.

Parameters:

**timer* the timer to free.

Returns:

a NULL pointer.

6.12.2.3 long get_current_ms (csi_timer_t * *timer*)

Calculates the milliseconds the timer is running, i.e. the time that passed since start_timer has been called.

Parameters:

**timer* the timer to use.

Returns:

the milliseconds since the start of the timer.

Definition at line 122 of file csi_time.c.

References diff_ms(), and csi_timer_t::start.

6.12.2.4 long get_current_us (csi_timer_t * *timer*)

Calculates the microseconds the timer is running, i.e. the time that passed since start_timer has been called.

Parameters:

**timer* the timer to use.

Returns:

the microseconds since the start of the timer.

Definition at line 138 of file csi_time.c.

References diff_us(), and csi_timer_t::start.

6.12.2.5 long get_diff_ms (csi_timer_t * *timer*)

Calculates the milliseconds between the start and stop of the specified timer. Please be aware that if either one of the start_timer or stop_timer method has not been called this function will produce an undefined behaviour.

Parameters:

timer the timer to use.

Returns:

the milliseconds between the start and stop of the timer.

Definition at line 170 of file csi_time.c.

References diff_ms(), csi_timer_t::start, and csi_timer_t::stop.

6.12.2.6 long get_diff_s (csi_timer_t * *timer*)

Calculates the seconds between the start and stop of the specified timer. Please be aware that if either one of the start_timer or stop_timer method has not been called this function will produce an undefined behaviour.

Parameters:

timer the timer to use.

Returns:

the seconds between the start and stop of the timer.

Definition at line 156 of file csi_time.c.

References diff_s(), csi_timer_t::start, and csi_timer_t::stop.

6.12.2.7 long get_diff_us (csi_timer_t * *timer*)

Calculates the microseconds between the start and stop of the specified timer. Please be aware that if either one of the start_timer or stop_timer method has not been called this function will produce an undefined behaviour.

Parameters:

timer the timer to use.

Returns:

the microseconds between the start and stop of the timer.

Definition at line 185 of file csi_time.c.

References diff_us(), csi_timer_t::start, and csi_timer_t::stop.

6.12.2.8 void printf_current_time (char * *prefix*, csi_timer_t * *timer*)

Prints the current time of the timer to stdout (`prefix`: minutes:seconds.milliseconds).

Parameters:

prefix the prefix to print before the timer output.

timer the timer.

Definition at line 215 of file csi_time.c.

References get_current_ms().

6.12.2.9 void printf_diff_time (char * *prefix*, csi_timer_t * *timer*)

Prints the difference time of the timer (end - start) to stdout (`prefix`: minutes:seconds.milliseconds).

Parameters:

prefix the prefix to print before the timer output.

timer the timer.

Definition at line 230 of file csi_time.c.

References get_current_ms(), and get_diff_s().

6.12.2.10 void start_timer (csi_timer_t * *timer*)

Starts the specified timer.

Parameters:

timer the timer to use.

Definition at line 82 of file csi_time.c.

References csi_timer_t::start.

6.12.2.11 void stop_timer (csi_timer_t * *timer*)

Stops the specified timer.

Parameters:

timer the timer to use.

Definition at line 92 of file csi_time.c.

References csi_timer_t::stop.

6.12.2.12 char* to_time_str (csi_timer_t * *timer*)

Creates a string representation as seconds.milliseconds of the specified csi_timer_t. Please note that the caller must free the result.

Parameters:

timer the timer.

Returns:

a string seconds:milliseconds.

Definition at line 198 of file csi_time.c.

References get_diff_ms(), and get_diff_s().

6.13 csi_tools.c File Reference

CSI utility functions of a general nature.

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <stdarg.h>
#include "csi_tools.h"
```

Functions

- void csi_fatal (const char *error,...)
- void csi_fatal_malloc ()
- void csi_fopen_error (const char *file, const char *operation)
- void csi_fwrite_error (const char *file)
- void csi_fread_error (const char *file)
- void csi_fclose_error (const char *file)

6.13.1 Detailed Description

CSI utility functions of a general nature.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_tools.c.

6.13.2 Function Documentation

6.13.2.1 void csi_fatal (const char * *error*, ...)

Convenience function to exit a program on a fatal error.

Parameters:

error the error message to print before exiting (may contain `printf` format expressions).

... parameter to pass to the `error` format string.

Definition at line 26 of file csi_tools.c.

6.13.2.2 void csi_fatal_malloc ()

Convenience function to exit a program on a fatal error with regard to memory allocation. It prints the default message CSI_FATAL_MALLOC to stdout before exiting.

Definition at line 45 of file csi_tools.c.

6.13.2.3 void csi_fclose_error (const char * *file*)

Prints an error message to stdout in case a close operation on a file fails. Uses `errno` to formulate the message.

Parameters:

file the file the close operation failed on.

Definition at line 99 of file csi_tools.c.

6.13.2.4 void csi_fopen_error (const char * *file*, const char * *operation*)

Prints an error message to stdout in case an open operation on a file fails. Uses `errno` to formulate the message.

Parameters:

file the file the open operation failed on.

operation the operation used. One of `CSI_OPEN_READ`, `CSI_OPEN_WRITE` or `CSI_OPEN_READ_WRITE`.

Definition at line 60 of file csi_tools.c.

6.13.2.5 void csi_fread_error (const char * *file*)

Prints an error message to stdout in case a read operation on a file fails. Uses `errno` to formulate the message.

Parameters:

∗file the file the read operation failed on.

Definition at line 86 of file csi_tools.c.

6.13.2.6 void csi_fwrite_error (const char ∗ *file*)

Prints an error message to stdout in case a write operation on a file fails. Uses `errno` to formulate the message.

Parameters:

∗file the file the write operation failed on.

Definition at line 73 of file csi_tools.c.

6.14 csi_tools.h File Reference

Header file for the tools library. All external functions are exposed here.

Defines

- #define CSI_FATAL_MALLOC "Error allocating memory"
- #define CSI_OPEN_READ "reading"
- #define CSI_OPEN_WRITE "writing"
- #define CSI_OPEN_READ_WRITE "reading/writing"

Functions

- void csi_fatal (const char ∗error,...)
- void csi_fatal_malloc ()
- void csi_fopen_error (const char ∗file, const char ∗operation)
- void csi_fwrite_error (const char ∗file)
- void csi_fread_error (const char ∗file)
- void csi_fclose_error (const char ∗file)

6.14.1 Detailed Description

Header file for the tools library. All external functions are exposed here.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file csi_tools.h.

6.14.2 Function Documentation

6.14.2.1 void csi_fatal (const char * *error*, ...)

Convenience function to exit a program on a fatal error.

Parameters:

error the error message to print before exiting (may contain `printf` format expressions).

... parameter to pass to the `error` format string.

Definition at line 26 of file csi_tools.c.

6.14.2.2 void csi_fatal_malloc ()

Convenience function to exit a program on a fatal error with regard to memory allocation. It prints the default message CSI_FATAL_MALLOC to stdout before exiting.

Definition at line 45 of file csi_tools.c.

References csi_fatal(), and CSI_FATAL_MALLOC.

6.14.2.3 void csi_fclose_error (const char * *file*)

Prints an error message to stdout in case a close operation on a file fails. Uses `errno` to formulate the message.

Parameters:

file the file the close operation failed on.

Definition at line 99 of file csi_tools.c.

6.14.2.4 void csi_fopen_error (const char * *file*, const char * *operation*)

Prints an error message to stdout in case an open operation on a file fails. Uses `errno` to formulate the message.

Parameters:

file the file the open operation failed on.

operation the operation used. One of `CSI_OPEN_READ`, `CSI_OPEN_WRITE` or `CSI_OPEN_READ_WRITE`.

Definition at line 60 of file csi_tools.c.

6.14.2.5 void csi_fread_error (const char * *file*)

Prints an error message to stdout in case a read operation on a file fails. Uses `errno` to formulate the message.

Parameters:

∗file the file the read operation failed on.

Definition at line 86 of file csi_tools.c.

6.14.2.6 void csi_fwrite_error (const char ∗ *file*)

Prints an error message to stdout in case a write operation on a file fails. Uses `errno` to formulate the message.

Parameters:

∗file the file the write operation failed on.

Definition at line 73 of file csi_tools.c.

6.15 fingerCapture.cpp File Reference

Implementation file to capture a picture from the camera, and save it as a jpg.

```
#include <Magick++.h>
#include "windows.h"
#include <conio.h>
#include <iostream>
#include "..\..\..\..\ext\tstapi\TSTBasicAPI.h"
#include <queue>
#include "fingerCapture.h"
#include <stdio.h>
#include <cv.h>
#include <highgui.h>
```

Namespaces

- namespace **Magick**

Classes

- class Event
- union Event::ucontent
- class CStateEvents

Enumerations

- enum EventType { keyboard, message }

Functions

- bool CheckKeyboard ()
- void __stdcall CallbackFn (TST_CallbackMessage *msg, LPVOID param)
- void SaveBitmap (TST_CallbackMessage *msg)
- void takePicture (void)
- void startFPSensor (void)
- void stopFPSensor (void)
- void waitforFPSensor2 (void)
- void waitforFPSensor (void)
- int scannFinger (void)
- int main (int argc, char *argv[ ])

Variables

- TST_CBFunc pFnCallback = (TST_CBFunc)CallbackFn
- CStateEvents stateEvents
- bool fExit = false
- bool fStarted = false
- bool fAsync = false
- vector< string > devices
- string currentDevice

6.15.1 Detailed Description

Implementation file to capture a picture from the camera, and save it as a jpg.

Maintainer: Samira Brulic <Samira.Brulic@csipc2.de>

Date: 2007.10.30

Definition in file fingerCapture.cpp.

6.15.2 Enumeration Type Documentation

6.15.2.1 enum EventType

Enumerator:

keyboard

message

Definition at line 36 of file fingerCapture.cpp.

6.15.3 Function Documentation

6.15.3.1 void __stdcall CallbackFn (TST_CallbackMessage * *msg*, LPVOID *param*)

generate a message event when the callback function was called

Definition at line 133 of file fingerCapture.cpp.

6.15.3.2 bool CheckKeyboard ()

create a keyboard event when a key was pressed

Definition at line 118 of file fingerCapture.cpp.

6.15.3.3 void SaveBitmap (TST_CallbackMessage * *msg*)

Save Bitmap to file

Parameters:

msg pointer on message that contains data from the scanner

Definition at line 143 of file fingerCapture.cpp.

6.15.3.4 int scannFinger (void)

scannFinger is the interface for the gui: it starts the scanner, take the picture and save into a file

Definition at line 407 of file fingerCapture.cpp.

6.15.3.5 void startFPSensor (void)

load the driver of the scanner

Definition at line 201 of file fingerCapture.cpp.

6.15.3.6 void stopFPSensor (void)

unload the driver of the scanner

Definition at line 211 of file fingerCapture.cpp.

6.15.3.7 void takePicture (void)

takePicture initiates the captiureoperation of the scanner

Definition at line 176 of file fingerCapture.cpp.

6.15.3.8 void waitforFPSensor (void)

waitforFPSensor processes the eventqueue of the scanner before initiating the capture-operation

Definition at line 315 of file fingerCapture.cpp.

6.15.3.9 void waitforFPSensor2 (void)

waitforFPSensor2 processes the eventqueue of the scanner after initiating the capture-operation

Definition at line 220 of file fingerCapture.cpp.

6.16 fingerCapture.h File Reference

Header file for the fingerprint scanner.

Functions

- int scannFinger (void)
- void waitforFPSensor (void)
- void waitforFPSensor2 (void)
- void startFPSensor (void)
- void stopFPSensor (void)
- void takePicture (void)
- void SaveBitmap (TST_CallbackMessage *msg)
- void __stdcall CallbackFn (TST_CallbackMessage *msg, LPVOID param)
- bool CheckKeyboard ()

6.16.1 Detailed Description

Header file for the fingerprint scanner.

Author:

Samira Brulic <Samira.Brulic@csipc2.de>

Definition in file fingerCapture.h.

6.16.2 Function Documentation

6.16.2.1 void __stdcall CallbackFn (TST_CallbackMessage * *msg*, LPVOID *param*)

generate a message event when the callback function was called

Definition at line 133 of file fingerCapture.cpp.

References CStateEvents::AddStateEvent().

6.16.2.2 bool CheckKeyboard ()

create a keyboard event when a key was pressed

Definition at line 118 of file fingerCapture.cpp.

References CStateEvents::AddStateEvent().

6.16.2.3 void SaveBitmap (TST_CallbackMessage * *msg*)

Save Bitmap to file

Parameters:

msg pointer on message that contains data from the scanner

Definition at line 143 of file fingerCapture.cpp.

6.16.2.4 int scannFinger (void)

scannFinger is the interface for the gui: it starts the scanner, take the picture and save into a file

Definition at line 407 of file fingerCapture.cpp.

References CStateEvents::AddStateEvent(), startFPSensor(), stopFPSensor(), takePicture(), waitforFPSensor(), and waitforFPSensor2().

6.16.2.5 void startFPSensor (void)

load the driver of the scanner

Definition at line 201 of file fingerCapture.cpp.

References fStarted, and pFnCallback.

6.16.2.6 void stopFPSensor (void)

unload the driver of the scanner

Definition at line 211 of file fingerCapture.cpp.

6.16.2.7 void takePicture (void)

takePicture initiates the captiureoperation of the scanner

Definition at line 176 of file fingerCapture.cpp.

References currentDevice, devices, and fAsync.

6.16.2.8 void waitforFPSensor (void)

waitforFPSensor processes the eventqueue of the scanner before initiating the capture-operation

Definition at line 315 of file fingerCapture.cpp.

References Event::content, currentDevice, devices, fAsync, CStateEvents::GetStateEvent(), message, Event::ucontent::msg, and SaveBitmap().

6.16.2.9 void waitforFPSensor2 (void)

waitforFPSensor2 processes the eventqueue of the scanner after initiating the capture-operation

Definition at line 220 of file fingerCapture.cpp.

References CheckKeyboard(), Event::content, currentDevice, devices, fAsync, CStateEvents::GetStateEvent(), message, Event::ucontent::msg, and SaveBitmap().

6.17 gui_control.cpp File Reference

Implementation file for the gui man-machine interaction, frontend-cluster communication and integration of frontend modules.

```
#include "stdafx.h"
#include "gui.h"
#include "gui_control.h"
#include <string>
#include <fstream>
#include "..\scanner\iris\irisCapture\irisCapture\Capture.h"
#include "..\..\..\ext\PracticalSocket.h"
#include "..\..\shared\csi_struct.h"
#include <cv.h>
#include <highgui.h>
```

Namespaces

- namespace **std**

Classes

- class CAboutDlg

Functions

- ON_WM_LBUTTONDOWN () BOOL gui_control
- void previewClose (int event, int x, int y, int flags, void *param)

Variables

- char * server
- int port
- char * iris_extraction_path
- char * iris_extraction_path2
- char * finger_extraction_path
- char * finger_scanner_path
- int iris_extraction_on_frontend
- int finger_extraction_on_frontend
- int finger_match_mode
- int iris_camera_mode
- string imgShow = ""
- int iris_capture_flag = 0
- int iris_ext_flag = 0
- CCltifa_apix1 apix
- Capture * cap = new Capture(&apix)
- int help_on = 0
- int help_status = 1
- int preview = 0
- CRect rect
- LOGFONT lf
- CBalloonTip * m_pBalloonTip
- CBalloonTip * m_pBalloonTip2
- CBalloonTip * m_pBalloonTip3

6.17.1 Detailed Description

Implementation file for the gui man-machine interaction, frontend-cluster communication and integration of frontend modules.

Author:

Andre Ueckermann <andre.ueckermann@csipc2.de>

Definition in file gui_control.cpp.

6.17.2 Function Documentation

6.17.2.1 void previewClose (int *event*, int *x*, int *y*, int *flags*, void * *param*)

Close the preview-window of iris-feature-extraction and call the reactivation of the tooltips with the preview-variable. Click inside the preview-window. For clicking somewhere outside the window the method OnLButtonDown is used.

Parameters:

event what kind of event is triggered, 1 for left mouse click

x horizontal coordinate of the mouse
y vertical coordinate of the mouse
flags a combination of some events like CTRL-key is pressed
param user defined parameter passed to the callback function

Definition at line 613 of file gui_control.cpp.

References preview.

6.17.3 Variable Documentation

6.17.3.1 CCltifa_apix1 apix

Object of the iriscamera api

Definition at line 63 of file gui_control.cpp.

6.17.3.2 Capture* cap = new Capture(&apix)

Object of the Capture class

Definition at line 65 of file gui_control.cpp.

6.17.3.3 int finger_extraction_on_frontend

Determines if the extraction for fingerprint is on frontend or on cluster (set up in ini-file)

Definition at line 47 of file gui_control.cpp.

6.17.3.4 char* finger_extraction_path

Path for fingerprint feature extraction (set up in ini-file)

Definition at line 41 of file gui_control.cpp.

6.17.3.5 int finger_match_mode

Determines which matching mode is used for fingerprints (set up in ini-file)

Definition at line 49 of file gui_control.cpp.

6.17.3.6 char* finger_scanner_path

Path for fingerprint scanner (set up in ini-file)

Definition at line 43 of file gui_control.cpp.

6.17.3.7 int help_on = 0

Determines if help is enabled or disabled

Definition at line 69 of file gui_control.cpp.

6.17.3.8 int help_status = 1

Determines which help tooltips are shown

Definition at line 71 of file gui_control.cpp.

6.17.3.9 string imgShow = ""

Saves the location of the preview image

Definition at line 55 of file gui_control.cpp.

6.17.3.10 int iris_camera_mode

Determines which type of camera is used (set up in ini-file)

Definition at line 51 of file gui_control.cpp.

6.17.3.11 int iris_capture_flag = 0

Used for manual shot of iris-pictures

Definition at line 57 of file gui_control.cpp.

6.17.3.12 int iris_ext_flag = 0

The feature extraction for iris is executed twice. One time for preview and one time for extraction

Definition at line 59 of file gui_control.cpp.

6.17.3.13 int iris_extraction_on_frontend

Determines if the extraction for iris is on frontend or on cluster (set up in ini-file)

Definition at line 45 of file gui_control.cpp.

6.17.3.14 char* iris_extraction_path

Path for iris extraction preview (set up in ini-file)

Definition at line 37 of file gui_control.cpp.

6.17.3.15 char* iris_extraction_path2

Path for iris feature extraction (set up in ini-file)

Definition at line 39 of file gui_control.cpp.

6.17.3.16 LOGFONT lf

The fontstyle of the tooltip

Definition at line 78 of file gui_control.cpp.

6.17.3.17 CBalloonTip* m_pBalloonTip

An object of the tooltip

Definition at line 82 of file gui_control.cpp.

6.17.3.18 CBalloonTip* m_pBalloonTip2

An object of the tooltip

Definition at line 84 of file gui_control.cpp.

6.17.3.19 CBalloonTip* m_pBalloonTip3

An object of the tooltip

Definition at line 86 of file gui_control.cpp.

6.17.3.20 int port

Serverport for Frontend-Cluster connection (set up in ini-file)

Definition at line 35 of file gui_control.cpp.

6.17.3.21 int preview = 0

If a preview-image is open, the tooltips become disabled

Definition at line 73 of file gui_control.cpp.

6.17.3.22 CRect rect

The rectangle for the tooltip

Definition at line 76 of file gui_control.cpp.

6.17.3.23 char* server

Servername for Frontend-Cluster connection (set up in ini-file)

Definition at line 33 of file gui_control.cpp.

6.18 gui_control.h File Reference

Header file for the gui man-machine interaction, frontend-cluster communication and integration of frontend modules.

```
#include "../../../ext/ResizeDlg.h"
#include "../../../ext/BalloonTip.h"
#include "../../../ext/Picture.h"
#include <string>
```

```
#include "..\scanner\iris\iriscapture\iriscapture\cltifa_-
apix1.h"
```

Classes

- class gui_control

6.18.1 Detailed Description

Header file for the gui man-machine interaction, frontend-cluster communication and integration of frontend modules.

Author:

Andre Ueckermann <andre.ueckermann@csipc2.de>

Definition in file gui_control.h.

6.19 image_utils.c File Reference

The image utilities provide some helper function with regard to image loading and saving.

```
#include <magick/api.h>
#include <wand/MagickWand.h>
#include <string.h>
#include <stdlib.h>
```

Functions

- int read_image (const char *ifile, unsigned char **odata, int *olen, int *ow, int *oh, int *od)
- int write_image (const char *ifile, const unsigned char *odata, const int ow, const int oh)
- int read_image_wand (MagickWand *wand, const char *ifile)

6.19.1 Detailed Description

The image utilities provide some helper function with regard to image loading and saving.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file image_utils.c.

6.19.2 Function Documentation

6.19.2.1 int read_image (const char * *ifile*, unsigned char ** *odata*, int * *olen*, int * *ow*, int * *oh*, int * *od*)

Reads an image from a file using the ImageMagick libary.

Parameters:

- ∗***ifile*** the file name of the image file.
- ∗∗***odata*** a pointer to a char array where the image data should be written to.
- ∗***olen*** a pointer to an integer where the length of the image data array (`**odata`) should be written to.
- ∗***ow*** a pointer to an integer where the image width should be written to.
- ∗***oh*** a pointer to an integer where the image height should be written to.
- ∗***od*** a pointer to an integer where the image depth (bits per pixel) should be written to.

Returns:

EXIT_SUCCESS if everything is OK, EXIT_FAILURE otherwise.

Definition at line 37 of file image_utils.c.

6.19.2.2 int read_image_wand (MagickWand * *wand*, const char * *ifile*)

Reads the specified image file into the `MagickWand`.

Parameters:

- ∗***wand*** a pointer to the MagickWand to fill.
- ∗***ifile*** the file name of the image file.

Returns:

EXIT_SUCCESS if everything is OK, EXIT_FAILURE otherwise.

Definition at line 145 of file image_utils.c.

6.19.2.3 int write_image (const char * *ifile*, const unsigned char * *odata*, const int *ow*, const int *oh*)

Writes an image to a file using the ImageMagick libary.

Parameters:

- ∗***ifile*** the file name of the image file.
- ∗***odata*** a pointer to the image data.
- ***ow*** the image width.

oh the image height.

Returns:

EXIT_SUCCESS if everything is OK, EXIT_FAILURE otherwise.

Definition at line 96 of file image_utils.c.

6.20 image_utils.h File Reference

Header for the image_utils. The image utilities provided some helper function with regard to image loading and processing.

`#include <wand/MagickWand.h>`

Functions

- int read_image (const char *ifile, unsigned char **odata, int *olen, int *ow, int *oh, int *od)
- int write_image (const char *ifile, const unsigned char *odata, const int ow, const int oh)
- int read_image_wand (MagickWand *wand, const char *ifile)

6.20.1 Detailed Description

Header for the image_utils. The image utilities provided some helper function with regard to image loading and processing.

Author:

Elmar Weber <university@elmarweber.org>

Definition in file image_utils.h.

6.20.2 Function Documentation

6.20.2.1 int read_image (const char * *ifile*, unsigned char ** *odata*, int * *olen*, int * *ow*, int * *oh*, int * *od*)

Reads an image from a file using the ImageMagick libary.

Parameters:

ifile the file name of the image file.

****odata*** a pointer to a char array where the image data should be written to.

olen a pointer to an integer where the length of the image data array (**odata) should be written to.

ow a pointer to an integer where the image width should be written to.

∗***oh*** a pointer to an integer where the image height should be written to.

∗***od*** a pointer to an integer where the image depth (bits per pixel) should be written to.

Returns:

EXIT_SUCCESS if everything is OK, EXIT_FAILURE otherwise.

Definition at line 37 of file image_utils.c.

6.20.2.2 int read_image_wand (MagickWand ∗ *wand*, const char ∗ *ifile*)

Reads the specified image file into the `MagickWand`.

Parameters:

∗***wand*** a pointer to the MagickWand to fill.

∗***ifile*** the file name of the image file.

Returns:

EXIT_SUCCESS if everything is OK, EXIT_FAILURE otherwise.

Definition at line 145 of file image_utils.c.

References read_image().

6.20.2.3 int write_image (const char ∗ *ifile*, const unsigned char ∗ *odata*, const int *ow*, const int *oh*)

Writes an image to a file using the ImageMagick libary.

Parameters:

∗***ifile*** the file name of the image file.

∗***odata*** a pointer to the image data.

ow the image width.

oh the image height.

Returns:

EXIT_SUCCESS if everything is OK, EXIT_FAILURE otherwise.

Definition at line 96 of file image_utils.c.

6.21 IrisCode.cpp File Reference

This file contains all necessary methods for generating the iriscode.

```
#include "IrisCode.hpp"
#include <iostream>
#include "vigra/gaborfilter.hxx"
#include "vigra/stdimage.hxx"
#include "vigra/edgedetection.hxx"
#include "vigra/impex.hxx"
#include "vigra/basicimage.hxx"
#include "vigra/rgbvalue.hxx"
#include "vigra/splineimageview.hxx"
#include <vector>
#include <math.h>
#include <complex>
#include <fstream>
#include <cstdlib>
#include <time.h>
```

Namespaces

- namespace Iris
- namespace **vigra**

6.21.1 Detailed Description

This file contains all necessary methods for generating the iriscode.

Author:

Christoph Scholz
Nils Timm

Definition in file IrisCode.cpp.

6.22 IrisCode.hpp File Reference

This is the header file for iriscode generation.

```
#include <iostream>
#include <string>
#include "vigra/gaborfilter.hxx"
#include "vigra/stdimage.hxx"
```

```
#include "vigra/edgedetection.hxx"
#include "vigra/impex.hxx"
#include "vigra/basicimage.hxx"
#include "vigra/rgbvalue.hxx"
#include <complex>
#include <vector>
```

Namespaces

- namespace Iris

Classes

- class Iris::IrisCode

6.22.1 Detailed Description

This is the header file for iriscode generation.

Author:

Christoph Scholz
Nils Timm

Definition in file IrisCode.hpp.

6.23 IrisCodeMatcher.cpp File Reference

This file contains all necessary methods for iriscode matching.

```
#include "IrisCodeMatcher.hpp"
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <time.h>
#include <math.h>
```

Namespaces

- namespace Iris

6.23.1 Detailed Description

This file contains all necessary methods for iriscode matching.

Author:

Nils Timm
Christoph Scholz

Definition in file NewIrisCodeMatcher/IrisCodeMatcher.cpp.

6.24 main.cpp File Reference

This file contains the main method for iriscode generation.

```
#include "IrisCode.hpp"
#include <iostream>
#include "vigra/gaborfilter.hxx"
#include "vigra/stdimage.hxx"
#include "vigra/edgedetection.hxx"
#include "vigra/impex.hxx"
#include "vigra/basicimage.hxx"
#include "vigra/rgbvalue.hxx"
#include <fstream>
```

Functions

- int main (int argc, char ∗argv[ ])

6.24.1 Detailed Description

This file contains the main method for iriscode generation.

Author:

Christoph Scholz
Nils Timm

Definition in file IrisCodeGenerator/main.cpp.

Index