Mister Spex

1.0

Generated by Doxygen 1.8.4

Wed Jan 22 2014 11:39:00

Contents

Chapter 1

Documentation of Mister Spex

This is the documentation of the software 'Mister Spex'. It was developed for the university course 'Praktische Aspekte der Informatik' in WS 13/14.

The proposal was declared as following:

++++++++++++++++++++ GERMAN ++++++++++++++++++++++++++++++

Student Sven Frank, 4442***

Projektname/working title: MisterSpecs

Kurze Zusammenfassung

Ein kleines Programm zur Erkennung von bestimmten Körperpartien/Gegenständen aus einem Video oder einer Live-Aufnahme. Anschließend können verschiedene Elemente auf die getrackten Objekte aufgesetzt werden. (Beispiel: digitales Aufsetzen einer Brille [richtige Position im Gesicht])

Detaillierte Beschreibung/Auflistung der Funktionalität

Materialinput:

- importieren von vorheraufgenommenen Videos (Standard: mp4-Format, h.264 Codec)
- Echtzeit-Videos über Webcam

Analysefunktionen:

- Gesichtserkennung bzw. Erkennung von verschiedenen Gesichtspartien, wie z.B. Nase

GUI

- Anzeige des Videos/Livebildes
- Auswahl für verschiedene Overlays

Optional:

- Tracking von weiteren Objekten abseits von Gesichtern
- Aufzeichnung von Webcamaufnahmen (inkl. Overlays) und Speichern in mp4-Format
- Unterstützung verschiedener Codes (Import/Export)
- Erstellen von eigenen Overlays
- passende Skalierung von Overlays
- passende GUI-Erweiterung für einzelne optinale Funktionen

Welche externen Bibliotheken/Tools werden benutzt und wofür?

OpenCV: Bildanalyse und -verarbeitung Bibliothek zur Endcodierung von Videodateien (Auswertung was sich am besten eignet)

Welche besonderen Anforderungen ergeben sich?

(z.B. komplexe Berechnungen in Echtzeit, effiziente Speicherverwaltung/Verarbeitung großer Datenmengen)

- verschiedene Funktionen
- Echtzeit-Tracking von Objekten
- Unterstützung von verschiedenen Dateitypen
- Verarbeitung der Datenmengen beim Aufzeichnen von Videos

++

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 Buffer< T > Class Template Reference

Buffer template class.

`#include <Buffer.h>`

Public Member Functions

- Buffer (int size)

  Template function to initialize a new buffer of type T with a specific size. Compareable to a constructor.
- void add (const T &data, bool dropIfFull=false)

  Template function add data of type T to the buffer. Data can be dropped if buffer is full.
- T get ()

  Template function get data of type T from the buffer.
- int size ()

  Template function to get actual buffer size.
- int maxSize ()

  Template function to get max buffer size.
- bool clear ()

  Template function to clear the actual buffer.
- bool isFull ()

  Template function check if buffer is full.
- bool isEmpty ()

  Template function check if buffer is empty.

Private Attributes

- QMutex queueProtect

  QMutex.
- QQueue< T > queue

  QQueue.
- QSemaphore * freeSlots

  QSemaphore - free Slots.
- QSemaphore * usedSlots

  QSemaphore - used Slots.
- QSemaphore * clearBuffer_add

QSemaphore - clear add Buffer.

- QSemaphore * clearBuffer_get

  QSemaphore - clear get Buffer.

- int bufferSize

  Buffer size.

5.1.1 Detailed Description

template<class T>class Buffer< T >

Buffer template class.

The buffer template class is the basis to save image temporarily to display them later.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 template<class T > Buffer< T >::Buffer (int *size*)

Template function to initialize a new buffer of type T with a specific size. Compareable to a constructor.

Parameters

size	an integer argument.

5.1.3 Member Function Documentation

5.1.3.1 template<class T > void Buffer< T >::add (const T & *data*, bool *dropIfFull* = `false`)

Template function add data of type T to the buffer. Data can be dropped if buffer is full.

Parameters

data	an argument of the specific type T.
dropIfFull	an boolean argument to set if an image should be dropped when buffer is full.

5.1.3.2 template<class T > bool Buffer< T >::clear ( )

Template function to clear the actual buffer.

Returns

True if buffer was released. In any other case the function return false.

5.1.3.3 template<class T > T Buffer< T >::get ( )

Template function get data of type T from the buffer.

Returns

Data with type T

5.1.3.4 template<class T > bool Buffer< T >::isEmpty ( )

Template function check if buffer is empty.

Returns

True if buffer is empty, else false.

5.1.3.5 template<class T > bool Buffer< T >::isFull ( )

Template function check if buffer is full.

Returns

True if buffer is full, else false.

5.1.3.6 template<class T > int Buffer< T >::maxSize ( )

Template function to get max buffer size.

Returns

Size of the buffer as an integer value.

5.1.3.7 template<class T > int Buffer< T >::size ( )

Template function to get actual buffer size.

Returns

Actual size of the buffer as an integer value.

5.1.4 Member Data Documentation

5.1.4.1 template<class T> int Buffer< T >::bufferSize [private]

Buffer size.

Contains the size of the buffer (Integer value).

5.1.4.2 template<class T> QSemaphore* Buffer< T >::clearBuffer_add [private]

QSemaphore - clear add Buffer.

Temporarily buffer to add the image to the queue.

5.1.4.3 template<class T> QSemaphore* Buffer< T >::clearBuffer_get [private]

QSemaphore - clear get Buffer.

Temporarily buffer to get the image from the queue.

5.1.4.4 template<class T> QSemaphore* Buffer< T >::freeSlots [private]

QSemaphore - free Slots.

Checks for for free slots to save the image.

5.1.4.5 template<class T> QQueue<T> Buffer< T >::queue [private]

QQueue.

Place where the actual image will be saved before displaying.

5.1.4.6 template<class T> QMutex Buffer< T >::queueProtect [private]

QMutex.

Mutex to protect the image queue while adding the latest image.

5.1.4.7 template<class T> QSemaphore* Buffer< T >::usedSlots [private]

QSemaphore - used Slots.

Returns which slots are actually in use.

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

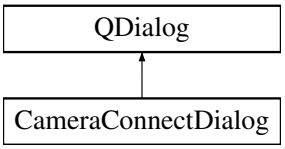
- Buffer.h

5.2 CameraConnectDialog Class Reference

Camera connect dialog class.

`#include <CameraConnectDialog.h>`

Inheritance diagram for CameraConnectDialog:



Public Member Functions

- CameraConnectDialog (QWidget *parent=0)
  Constructor of the dialog to set up the UI from the Qt file.
- ~CameraConnectDialog ()
  Destructor of the dialog to delete the created UI.
- int getMode ()
  Function which return the user selected mode.

Private Attributes

- Ui::CameraConnectDialog * ui
  CameraConnectDialog UI pointer.

5.2.1 Detailed Description

Camera connect dialog class.

Dialog class to select the mode in which the software should run.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 CameraConnectDialog::CameraConnectDialog (QWidget * *parent* = 0) [explicit]

Constructor of the dialog to set up the UI from the Qt file.

Parameters

parent	an QWidget argument. If no parameter is give parent = 0.

5.2.3 Member Function Documentation

5.2.3.1 int CameraConnectDialog::getMode ( )

Function which return the user selected mode.

Returns

An integer value which identifies the selected mode.

5.2.4 Member Data Documentation

5.2.4.1 Ui::CameraConnectDialog* CameraConnectDialog::ui [private]

CameraConnectDialog UI pointer.

Pointer the the user interface file of the dialog.

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

- CameraConnectDialog.h
- CameraConnectDialog.cpp

5.3 FaceDetection Class Reference

FaceDetection class.

`#include <FaceDetection.h>`

Public Member Functions

- FaceDetection ()
  Class constructor which creates a new handler.
- cv::Mat detectObjects (cv::Mat &, ImageProcessingSettings, ImageProcessingFlags)
  Function to detect objects in a given image. In this case face objects.
- void cvCreateOverlay (QString, const cv::Mat &, cv::Mat &, cv::Point2i, int)
  Function to draw a specific overlay on the given image.
- void cvCreateOverlayGlasses (QString, const cv::Mat &, cv::Mat &, cv::Point2i, int)
  Special draw function for the glasses overlay because the special anker point behaviour.

Private Attributes

- double **tem_cropfactor**
- cv::Mat **temp_glass_overlay**

5.3.1 Detailed Description

FaceDetection class.

Class has functions for object detection and overlay composing.

5.3.2 Member Function Documentation

5.3.2.1 void FaceDetection::cvCreateOverlay (QString *fn*, const cv::Mat & *src*, cv::Mat & *output*, cv::Point2i *location*, int *scale*)

Function to draw a specific overlay on the given image.

Parameters

fn	an QString name of the overlay file to load.
src	an Mat image (OpenCV type) file.
output	an Mat image (OpenCV type) file which also contains the overlay.
location	an Point2i (OpenCV type for single point with to integers coordinates) which is the anker point of the overlay.
scale	an integer value for scaling the overlay according to the image size.

5.3.2.2 void FaceDetection::cvCreateOverlayGlasses (QString *fn*, const cv::Mat & *src*, cv::Mat & *output*, cv::Point2i *location*, int *scale*)

Special draw function for the glasses overlay because the special anker point behaviour.

Parameters

fn	an QString name of the overlay file to load.
src	an Mat image (OpenCV type) file.
output	an Mat image (OpenCV type) file which also contains the overlay.
location	an Point2i (OpenCV type for single point with to integers coordinates) which is the anker point of the overlay.
scale	an integer value for scaling the overlay according to the image size.

5.3.2.3 cv::Mat FaceDetection::detectObjects (cv::Mat & *inMat*, ImageProcessingSettings *settings*, ImageProcessingFlags *drawFlags*)

Function to detect objects in a given image. In this case face objects.

Parameters

inMat	an Mat image (OpenCV type) file.
settings	an ImageProcessingSettings struct which contains the settings for the area which should be detected

drawFlags	an ImageProcessingFlags struct which contains bool values about overlay showing and detection drawing

Returns

Image in Mat type with in image drawn detection and/or overlays

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

- FaceDetection.h
- FaceDetection.cpp

5.4 ImageHandler Class Reference

Image handler class.

`#include <ImageHandler.h>`

Public Member Functions

- ImageHandler ()
  Class constructor which creates a new handler.
- ~ImageHandler ()
  Class destructor.
- QPixmap loadImageFromFile (QString, struct ImageProcessingSettings, struct ImageProcessingFlags, bool)
  Function to load a single image from a file.
- QPixmap applyChangesToImageProc (struct ImageProcessingSettings, struct ImageProcessingFlags)
  Function to apply changes on the selected image.
- bool saveImageToFile (QString, QString)
  Function to save the display image to file.

Private Attributes

- FaceDetection * faceDetection
  FaceDetection.
- Mat image
  Mat (OpenCV image format).
- Mat orig_image
  Mat (OpenCV image format).

5.4.1 Detailed Description

Image handler class.

Class to handle static images for the software.

5.4.2 Member Function Documentation

5.4.2.1 QPixmap ImageHandler::applyChangesToImageProc (struct ImageProcessingSettings *imgProcSettings*, struct ImageProcessingFlags *drawFlags*)

Function to apply changes on the selected image.

Parameters

imgProcSettings	an ImageProcessingSettings struct which contains the settings for the area which should be detected
drawFlags	an ImageProcessingFlags struct which contains bool values about overlay showing and detection drawing

Returns

The function returns the selected image converted to QPixmap format with applied detection

5.4.2.2 QPixmap ImageHandler::loadImageFromFile (QString *fn*, struct ImageProcessingSettings *imgProcSettings*, struct ImageProcessingFlags *drawFlags*, bool *imageDetection*)

Function to load a single image from a file.

Parameters

fn	a QString argument which contains the name of the file to load.
imgProcSettings	an ImageProcessingSettings struct which contains the settings for the area which should be detected
drawFlags	an ImageProcessingFlags struct which contains bool values about overlay showing and detection drawing
imageDetection	a boolean value to say if image detection should be done.

Returns

The function returns the selected image converted to QPixmap format with applied detection

5.4.2.3 bool ImageHandler::saveImageToFile (QString *fn*, QString *dir*)

Function to save the display image to file.

Parameters

fn	a QString argument which contains the name to save the image.
dir	a QString argument which contains the directory name where the file should be saved.

Returns

Boolean value which says if the process was successful or not.

5.4.3 Member Data Documentation

5.4.3.1 FaceDetection* ImageHandler::faceDetection [private]

FaceDetection.

Pointer to instance of FaceDetection class.

5.4.3.2 Mat ImageHandler::image [private]

Mat (OpenCV image format).

Image which will be display in the main window after conversion.

5.4.3.3 Mat ImageHandler::orig_image [private]

Mat (OpenCV image format).

Original image.

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

- ImageHandler.h
- ImageHandler.cpp

5.5 ImageProcessingFlags Struct Reference

Structure which contains information about the flags. Determine what process shall be done.

`#include <Structures.h>`

Public Attributes

- bool showDetectionOn

  Bool.
- bool showOverlaysOn

  Bool.

5.5.1 Detailed Description

Structure which contains information about the flags. Determine what process shall be done.

Structure with two boolean values.

5.5.2 Member Data Documentation

5.5.2.1 bool ImageProcessingFlags::showDetectionOn

Bool.

Boolean value if detection shall be drawn.

5.5.2.2 bool ImageProcessingFlags::showOverlaysOn

Bool.

Boolean value if overlays shall be drawn.

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

- Structures.h

5.6 ImageProcessingSettings Struct Reference

Structure which contains information about the overlays apply on the image.

`#include <Structures.h>`

Public Attributes

- bool overlayParam1

  Bool.
- bool overlayParam2

  Bool.
- bool overlayParam3

  Bool.

5.6.1 Detailed Description

Structure which contains information about the overlays apply on the image.

Structure with three boolean values.

5.6.2 Member Data Documentation

5.6.2.1 bool ImageProcessingSettings::overlayParam1

Bool.

Boolean value if glasses shall be displayed.

5.6.2.2 bool ImageProcessingSettings::overlayParam2

Bool.

Boolean value if funny nose shall be displayed.

5.6.2.3 bool ImageProcessingSettings::overlayParam3

Bool.

Boolean value if red lips shall be displayed.

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

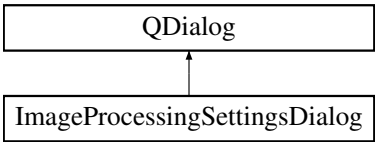
- Structures.h

5.7 ImageProcessingSettingsDialog Class Reference

Image processing dialog class.

`#include <ImageProcessingSettingsDialog.h>`

Inheritance diagram for ImageProcessingSettingsDialog:



Public Slots

- void updateStoredSettingsFromDialog ()

  Public slot to updated the stored settings from the dialog ones.

Signals

- void newImageProcessingSettings (struct ImageProcessingSettings p_settings)

  Signal: New ImageProcessingSettings are set.

Public Member Functions

- ImageProcessingSettingsDialog (QWidget *parent=0)

  Constructor of the dialog to set up the UI from the Qt file.
- ~ImageProcessingSettingsDialog ()

  Destructor of the dialog to delete the created UI.
- void updateDialogSettingsFromStored ()

  Function which updates the dialog settings from the stored ones.
- ImageProcessingSettings getImgProcSettingsForImg ()

  Function to get actual ImageProcessingSettings struct.
- ImageProcessingFlags ImgProcFlagsForImg ()

  Function to get actual ImageProcessingFlags struct.

Private Slots

- void resetOverlays ()

  Private slot to reset the display settings.

Private Attributes

- Ui::ImageProcessingSettingsDialog * ui

  ImageProcessingSettingsDialog UI pointer.
- ImageProcessingSettings imageProcessingSettings

  ImageProcessingSettings struct.

5.7.1 Detailed Description

Image processing dialog class.

Dialog to set specifc parameters which will be applied on the displayed image.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 ImageProcessingSettingsDialog::ImageProcessingSettingsDialog (QWidget * *parent* = 0) [explicit]

Constructor of the dialog to set up the UI from the Qt file.

Parameters

parent	an QWidget argument. If no parameter is give parent = 0.

5.7.3 Member Function Documentation

5.7.3.1 ImageProcessingSettings ImageProcessingSettingsDialog::getImgProcSettingsForImg ( )

Function to get actual ImageProcessingSettings struct.

Returns

ImageProcessingSettings struct which contains the acutal settings.

5.7.3.2 ImageProcessingFlags ImageProcessingSettingsDialog::ImgProcFlagsForImg ( )

Function to get actual ImageProcessingFlags struct.

Returns

ImageProcessingFlags struct which contains the acutal flags.

5.7.3.3 void ImageProcessingSettingsDialog::newImageProcessingSettings (struct ImageProcessingSettings *p_settings*) [signal]

Signal: New ImageProcessingSettings are set.

Parameters

p_settings	ImageProcessingSettings struct as argument.

5.7.3.4 void ImageProcessingSettingsDialog::resetOverlays ( ) [private], [slot]

Private slot to reset the display settings.

Private slot tocall function to reset settings. Value from Config.h are used.

5.7.3.5 void ImageProcessingSettingsDialog::updateStoredSettingsFromDialog ( ) [slot]

Public slot to updated the stored settings from the dialog ones.

Public slot to connect with a specific action in program that should call this function

5.7.4 Member Data Documentation

5.7.4.1 ImageProcessingSettings ImageProcessingSettingsDialog::imageProcessingSettings [private]

ImageProcessingSettings struct.

Structure of several settings.

5.7.4.2 Ui::ImageProcessingSettingsDialog* ImageProcessingSettingsDialog::ui [private]

ImageProcessingSettingsDialog UI pointer.

Pointer the the user interface file of the dialog.

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

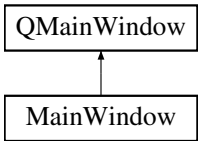
- ImageProcessingSettingsDialog.h
- ImageProcessingSettingsDialog.cpp

5.8 MainWindow Class Reference

MainWindow class creates the main application window and builds the basis.

`#include <MainWindow.h>`

Inheritance diagram for MainWindow:



Public Slots

- void selectMode ()

  Public slot to select mode.
- void connectToCamera ()

  Public slot to connect to camera.
- void disconnectCamera ()

  Public slot to disconnect camera.
- void showAboutDialog ()

  Public slot to show about dialog.
- void setDetection (bool)

  Public slot to set detection flag.
- void setImageProcessingSettings ()

  Public slot to set ImageProcessingSettings.
- void setRecDir ()

  Public slot to set recording directory.
- void takeSnapshot ()

  Public slot to take a snapshot of the actual displayed image.
- void recordVideo ()

  Public slot to record a video in LiveView mode.

Signals

- void newImageProcessingFlags (struct ImageProcessingFlags imageProcessingFlags)

  Signal: Send new ImageProcessingFlags.
- void changeDetectionState (bool detection)

  Signal: Change detection state flag.
- void setROI (QRect roi)

Signal: Set Region of interest.

- void setSaveParams (QString dir, QString fn)

  Signal: Set save parameter.
- void setLiveViewSaveImgFlag (bool sFlag)

  Signal: Set LiveView snapshot flag.
- void setRecordFlag (bool rFlag)

  Signal: Set record flag.

Public Member Functions

- MainWindow (QWidget *parent=0)

  Constructor of the main window to set up the UI from the Qt file.
- ~MainWindow ()

  Destructor of the dialog to delete the created UI.

Private Slots

- void updateFrame (const QPixmap &frame)

  Private slot to update the shown image in MainWindow.

Private Member Functions

- void showImage (QString)

  Function to show a static image from an image file.
- void applyChangesOnSingleImage ()

  Function to apply settings and flags on a loaded image.
- void stopVideoThread ()

  Function to stop the VideoThread.
- void stopProcessingThread ()

  Function to stop the ProcessingThread.
- void startPlayback (QString)

  Function to start displaying a video from a file.
- void stopPlayback ()

  Function to stop the Playback.
- void stopPlaybackThread ()

  Function to stop the PlaybackThread.

Private Attributes

- Ui::MainWindow * ui

  MainWindow UI pointer.
- ImageProcessingSettingsDialog * imageProcessingSettingsDialog

  ImageProcessingSettingsDialog pointer.
- VideoThread * videoThread

  VideoThread pointer.
- VideoImageBuffer * videoImageBuffer

  VideoImageBuffer pointer.
- ProcessingThread * processingThread

  ProcessingThread pointer.
- PlaybackThread * playbackThread

PlaybackThread *pointer.*

- RecordVideoThread * recordThread

  RecordVideoThread *pointer.*

- ImageHandler * imageHandler

  ImageHandler *pointer.*

- ImageProcessingFlags imageProcessingFlags

  ImageProcessingFlags *struct.*

- ImageProcessingSettings imageProcessingSettings

  ImageProcessingSettings *struct.*

- bool imageDetection

  Bool.

- bool detection

  Bool.

- QString saveDirectory

  QString.

- bool recording

  Bool.

- bool liveViewActive

  Bool.

- bool playbackActive

  Bool.

- int mode

  Integer.

- int deviceNumber

  Integer.

- QString file

  QString.

- int rec_width

  Integer.

- int rec_height

  Integer.

5.8.1 Detailed Description

MainWindow class creates the main application window and builds the basis.

Main windows of the application which does most of the work, like connection functions and buttons as well as the initilization of the different mode. Also function for recording and saving an image are provided here.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

Constructor of the main window to set up the UI from the Qt file.

Parameters

parent	an QWidget argument. If no parameter is give parent = 0.

5.8.3 Member Function Documentation

5.8.3.1 void MainWindow::changeDetectionState (bool *detection*) [signal]

Signal: Change detection state flag.

This signal emits the actual detection flag to the connected slot.

Parameters

detection	a bool argument.

5.8.3.2 void MainWindow::newImageProcessingFlags (struct ImageProcessingFlags *imageProcessingFlags*) [signal]

Signal: Send new ImageProcessingFlags.

This signal emits the actual ImageProcessingFlags to the connected slot.

Parameters

image-ProcessingFlags	a ImageProcessingFlags structure.

5.8.3.3 void MainWindow::setLiveViewSaveImgFlag (bool *sFlag*) [signal]

Signal: Set LiveView snapshot flag.

This signal emits save sflag to the connected slot in ProcessingThread.

Parameters

sFlag	a bool argument.

5.8.3.4 void MainWindow::setRecordFlag (bool *rFlag*) [signal]

Signal: Set record flag.

This signal emits record rflag to the connected slot in ProcessingThread.

Parameters

rFlag	a bool argument.

5.8.3.5 void MainWindow::setROI (QRect *roi*) [signal]

Signal: Set Region of interest.

This signal emits the desired ROI to the connected slot.

Parameters

roi	a QRect argument.

5.8.3.6 void MainWindow::setSaveParams (QString *dir,* QString *fn*) [signal]

Signal: Set save parameter.

This signal emits the save parameters to the connected slot.

Parameters

dir	a QString argument with Directory name.
fn	a QString argument with filename.

5.8.3.7 void MainWindow::showImage (QString *fn*) [private]

Function to show a static image from an image file.

Parameters

fn	a QString with the filename to load.

5.8.3.8 void MainWindow::startPlayback (QString *filename*) [private]

Function to start displaying a video from a file.

Parameters

filename	a QString with the filename to load.

5.8.3.9 void MainWindow::updateFrame (const QPixmap & *frame*) [private], [slot]

Private slot to update the shown image in MainWindow.

Parameters

frame	a QPixmap to load in specific label.

5.8.4 Member Data Documentation

5.8.4.1 bool MainWindow::detection [private]

Bool.

Boolean value if detection (in video or liveview shall be done.

5.8.4.2 int MainWindow::deviceNumber [private]

Integer.

Integer value for device number.

5.8.4.3 QString MainWindow::file [private]

QString.

Name of a file.

5.8.4.4 bool MainWindow::imageDetection [private]

Bool.

Boolean value if image detection shall be done.

5.8.4.5 ImageHandler* MainWindow::imageHandler [private]

ImageHandler pointer.

Pointer to a imageHandler.

5.8.4.6 ImageProcessingFlags MainWindow::imageProcessingFlags [private]

ImageProcessingFlags struct.

Structure of several flags.

5.8.4.7 ImageProcessingSettings MainWindow::imageProcessingSettings [private]

ImageProcessingSettings struct.

Structure of several settings.

5.8.4.8 ImageProcessingSettingsDialog* MainWindow::imageProcessingSettingsDialog [private]

ImageProcessingSettingsDialog pointer.

Pointer to a imageProcessingSettingsDialog.

5.8.4.9 bool MainWindow::liveViewActive [private]

Bool.

Boolean value to determine if liveview mode is on.

5.8.4.10 int MainWindow::mode [private]

Integer.

Integer value to determine the mode of the program.

5.8.4.11 bool MainWindow::playbackActive [private]

Bool.

Boolean value to determine if playback mode is on.

5.8.4.12 PlaybackThread* MainWindow::playbackThread [private]

PlaybackThread pointer.

Pointer to a playbackThread.

5.8.4.13 ProcessingThread* MainWindow::processingThread [private]

ProcessingThread pointer.

Pointer to a processingThread.

5.8.4.14 int MainWindow::rec_height [private]

Integer.

Integer value with recording height.

5.8.4.15 int MainWindow::rec_width [private]

Integer.

Integer value with recording width.

5.8.4.16 bool MainWindow::recording [private]

Bool.

Boolean value to determine if recording is on.

5.8.4.17 RecordVideoThread* MainWindow::recordThread [private]

RecordVideoThread pointer.

Pointer to a recordThread.

5.8.4.18 QString MainWindow::saveDirectory [private]

QString.

Directory to record a video.

5.8.4.19 Ui::MainWindow* MainWindow::ui [private]

MainWindow UI pointer.

Pointer the the user interface file of the dialog.

5.8.4.20 VideoImageBuffer* MainWindow::videoImageBuffer [private]

VideoImageBuffer pointer.

Pointer to a videoImageBuffer.

5.8.4.21 VideoThread* MainWindow::videoThread [private]

VideoThread pointer.

Pointer to a videoThread.

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

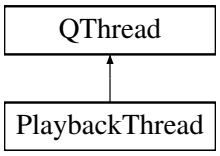
- MainWindow.h
- MainWindow.cpp

5.9 PlaybackThread Class Reference

Playback class which creates a thread to play a video.

`#include <PlaybackThread.h>`

Inheritance diagram for PlaybackThread:



Signals

- void nextFrame (const QPixmap &frame)

  Signal: Next frame.

Public Member Functions

- PlaybackThread (QString filename)

  Thread constructor class.
- void stop ()

  Set stop flag for thread worker.
- bool loadPlayback ()

  Function to load video file to VideoCapture.

Protected Member Functions

- void run ()

  Thread function which runs till doStop is set to true.

Private Slots

- void updateImageProcessingFlags (struct ImageProcessingFlags)

  Private slot to update the ImageProcessingFlags.
- void updateImageProcessingSettings (struct ImageProcessingSettings)

  Private slot to update the ImageProcessingSettings.
- void updateDetState (bool)

  Private slot to update the detection state.

Private Attributes

- FaceDetection * faceDetection

  FaceDetection.
- QMutex doStopMutex

  QMutex.
- QMutex processingMutex

  QMutex.
- volatile bool doStop

Bool.

- VideoCapture cap

  VideoCapture.
- Mat grabbedFrame

  Mat.
- QPixmap frame

  QPixmap.
- QString fn

  QString.
- int fps

  Integer.
- int frames

  Integer.
- int counter

  Integer.
- bool detectionState

  Boolean.
- struct ImageProcessingFlags imgProcFlags

  ImageProcessingFlags struct.
- struct ImageProcessingSettings imgProcSettings

  ImageProcessingSettings struct.

5.9.1 Detailed Description

Playback class which creates a thread to play a video.

Thread creating class to playback a video from file.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 PlaybackThread::PlaybackThread (QString *filename*)

Thread constructor class.

Parameters

filename	a QString argument which contains the video filename.

5.9.3 Member Function Documentation

5.9.3.1 void PlaybackThread::nextFrame (const QPixmap & *frame*) [signal]

Signal: Next frame.

This signale emits the next frame to the main window to display it.

Parameters

frame	a QPixmap image.

5.9.3.2 void PlaybackThread::run ( ) [protected]

Thread function which runs till doStop is set to true.

To function is the worker of the thread. It does all the calls to detection functions and emits new images.

5.9.3.3 void PlaybackThread::updateDetState (bool *flag*) [private], [slot]

Private slot to update the detection state.

Parameters

flag	a bool value.

5.9.3.4 void PlaybackThread::updateImageProcessingFlags (struct ImageProcessingFlags *imgProcFlags*) [private], [slot]

Private slot to update the ImageProcessingFlags.

Parameters

imgProcFlags	a ImageProcessingFlags struct.

5.9.3.5 void PlaybackThread::updateImageProcessingSettings (struct ImageProcessingSettings *imgProcSettings*) [private], [slot]

Private slot to update the ImageProcessingSettings.

Parameters

imgProcSettings	a ImageProcessingSettings struct.

5.9.4 Member Data Documentation

5.9.4.1 VideoCapture PlaybackThread::cap [private]

VideoCapture.

VideoCapture reads in the images from the given file.

5.9.4.2 int PlaybackThread::counter [private]

Integer.

Integer counter value.

5.9.4.3 bool PlaybackThread::detectionState [private]

Boolean.

Boolean value to make detection or not.

5.9.4.4 volatile bool PlaybackThread::doStop [private]

Bool.

Boolean value to stop the thread.

5.9.4.5 QMutex PlaybackThread::doStopMutex [private]

QMutex.

Mutex to stop the thread.

5.9.4.6 FaceDetection* PlaybackThread::faceDetection [private]

FaceDetection.

Pointer to instance of FaceDetection class.

5.9.4.7 QString PlaybackThread::fn [private]

QString.

Filename of the video file.

5.9.4.8 int PlaybackThread::fps [private]

Integer.

Integer value which contains the framerate.

5.9.4.9 QPixmap PlaybackThread::frame [private]

QPixmap.

Converted image in QPixmap format which will be emitted to main window to display

5.9.4.10 int PlaybackThread::frames [private]

Integer.

Integer value which contains the number of grabbed frames.

5.9.4.11 Mat PlaybackThread::grabbedFrame [private]

Mat.

Grabbed frame from the video file.

5.9.4.12 struct ImageProcessingFlags PlaybackThread::imgProcFlags [private]

ImageProcessingFlags struct.

Structure which contains information about the flags.

5.9.4.13 struct ImageProcessingSettings PlaybackThread::imgProcSettings [private]

ImageProcessingSettings struct.

Structure which contains information about the settings.

5.9.4.14 QMutex PlaybackThread::processingMutex [private]

QMutex.

Mutex which is set while processing the image.

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

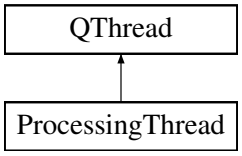
- PlaybackThread.h
- PlaybackThread.cpp

5.10 ProcessingThread Class Reference

Processing class which does image detection in webcam mode.

`#include <ProcessingThread.h>`

Inheritance diagram for ProcessingThread:



Signals

- void newFrame (const QPixmap &frame)

  Signal: New frame.
- void recFrame (const Mat &img)

  Signal: Record frame.

Public Member Functions

- ProcessingThread (VideoImageBuffer *videoImageBuffer, int deviceNumber)

  Thread constructor class.
- QRect getCurrentROI ()

  Function to get the current region of interest.
- void stop ()

  Set stop flag for thread worker.

Protected Member Functions

- void run ()

  Thread function which runs till doStop is set to true.

Private Slots

- void updateImageProcessingFlags (struct ImageProcessingFlags)

  Private slot to update the ImageProcessingFlags.
- void updateImageProcessingSettings (struct ImageProcessingSettings)

  Private slot to update the ImageProcessingSettings.
- void setROI (QRect roi)

  Private slot to set region of interest.
- void setSaveParams (QString dir, QString fn)

  Private slot to set save parameters.
- void setSaveImgFlag (bool sFlag)

  Private slot to set save image sFlag.
- void setFrameProcessing (bool flag)

  Private slot to set frame processing flag.
- void setRecFlag (bool rFlag)

  Private slot to set recording rFlag.

Private Member Functions

- void setROI ()

  Function to set ROI.
- void resetROI ()

  Function to reset ROI.
- bool saveCurrentImage (Mat)

  Function to save current image.

Private Attributes

- FaceDetection * faceDetection

  FaceDetection.
- VideoImageBuffer * videoImageBuffer

  VideoImageBuffer.
- Mat currentFrame

  Mat image.
- Mat currentFrameGrayscale

  Mat image.
- QPixmap frame

  QPixmap image.
- QMutex doStopMutex

  QMutex.
- QMutex processingMutex

  QMutex.
- Size frameSize

  Size.
- Point framePoint

  Point.
- Rect currentROI

  Rect.
- struct ImageProcessingFlags imgProcFlags

  ImageProcessingFlags struct.
- struct ImageProcessingSettings imgProcSettings

  ImageProcessingSettings struct.
- volatile bool doStop

  Bool.
- int deviceNumber

  Integer.
- bool enableFrameProcessing

  Bool.
- bool saveImage

  Bool.
- QString recDir

  QString.
- QString recFn

  QString.
- bool recFlag

  Bool.

5.10.1 Detailed Description

Processing class which does image detection in webcam mode.

Thread processing captured images.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 ProcessingThread::ProcessingThread (VideoImageBuffer * *videoImageBuffer,* int *deviceNumber*)

Thread constructor class.

Parameters

videoImage-Buffer	pointer to a buffer which contains the captured images.
deviceNumber	an integer argument.

5.10.3 Member Function Documentation

5.10.3.1 QRect ProcessingThread::getCurrentROI ( )

Function to get the current region of interest.

Returns

QRect - The current ROI

5.10.3.2 void ProcessingThread::newFrame (const QPixmap & *frame*) [signal]

Signal: New frame.

This signale emits the next frame to the main window to display it.

Parameters

frame	a QPixmap image.

5.10.3.3 void ProcessingThread::recFrame (const Mat & *img*) [signal]

Signal: Record frame.

This signale emits the next frame for recording.

Parameters

img	a Mat image.

5.10.3.4 void ProcessingThread::run ( ) [protected]

Thread function which runs till doStop is set to true.

To function is the worker of the thread. It does all the calls to detection functions and emits new images.

5.10.3.5 bool ProcessingThread::saveCurrentImage (Mat *img*) [private]

Function to save current image.

Parameters

img	a Mat image argument.

Returns

Boolean value if process was successful.

5.10.3.6 void ProcessingThread::setFrameProcessing (bool *flag*) [private], [slot]

Private slot to set frame processing flag.

Parameters

flag	a bool argument.

5.10.3.7 void ProcessingThread::setRecFlag (bool *rFlag*) [private], [slot]

Private slot to set recording rFlag.

Parameters

rFlag	a bool argument.

5.10.3.8 void ProcessingThread::setROI (QRect *roi*) [private], [slot]

Private slot to set region of interest.

Parameters

roi	a QRect argument.

5.10.3.9 void ProcessingThread::setSaveImgFlag (bool *sFlag*) [private], [slot]

Private slot to set save image sFlag.

Parameters

sFlag	a bool argument.

5.10.3.10 void ProcessingThread::setSaveParams (QString *dir,* QString *fn*) [private], [slot]

Private slot to set save parameters.

Parameters

dir	a QString argument which contains the Directory name.
fn	a QString argument which contains the file name.

5.10.3.11 void ProcessingThread::updateImageProcessingFlags (struct ImageProcessingFlags *imgProcFlags*) [private], [slot]

Private slot to update the ImageProcessingFlags.

Parameters

imgProcFlags	a ImageProcessingFlags struct.

5.10.3.12 void ProcessingThread::updateImageProcessingSettings (struct ImageProcessingSettings *imgProcSettings*) [private], [slot]

Private slot to update the ImageProcessingSettings.

Parameters

imgProcSettings	a ImageProcessingSettings struct.

5.10.4 Member Data Documentation

5.10.4.1 Mat ProcessingThread::currentFrame [private]

Mat image.

Current image.

5.10.4.2 Mat ProcessingThread::currentFrameGrayscale [private]

Mat image.

Current image in grayscale.

5.10.4.3 Rect ProcessingThread::currentROI [private]

Rect.

Current region of interest.

5.10.4.4 int ProcessingThread::deviceNumber [private]

Integer.

Integer value for device number.

5.10.4.5 volatile bool ProcessingThread::doStop [private]

Bool.

Boolean value to stop the thread.

5.10.4.6 QMutex ProcessingThread::doStopMutex [private]

QMutex.

Mutex to stop the thread.

5.10.4.7 bool ProcessingThread::enableFrameProcessing [private]

Bool.

Boolean value if frame processing is enabled.

5.10.4.8 FaceDetection* ProcessingThread::faceDetection [private]

FaceDetection.

Pointer to class instance for FaceDetection.

5.10.4.9 QPixmap ProcessingThread::frame [private]

QPixmap image.

Current image converted to QPixmap format.

5.10.4.10 Point ProcessingThread::framePoint [private]

Point.

Frame point.

5.10.4.11 Size ProcessingThread::frameSize [private]

Size.

Size of the frame.

5.10.4.12 struct ImageProcessingFlags ProcessingThread::imgProcFlags [private]

ImageProcessingFlags struct.

Structure which contains information about the flags.

5.10.4.13 struct ImageProcessingSettings ProcessingThread::imgProcSettings [private]

ImageProcessingSettings struct.

Structure which contains information about the settings.

5.10.4.14 QMutex ProcessingThread::processingMutex [private]

QMutex.

Mutex which is set while processing the image.

5.10.4.15 QString ProcessingThread::recDir [private]

QString.

Directory to record a video.

5.10.4.16 bool ProcessingThread::recFlag [private]

Bool.

State if video shall be recorded.

5.10.4.17 QString ProcessingThread::recFn [private]

QString.

Filename of the recorded video.

5.10.4.18 bool ProcessingThread::saveImage [private]

Bool.

Boolean value if image shall be saved.

5.10.4.19 VideoImageBuffer* ProcessingThread::videoImageBuffer [private]

VideoImageBuffer.

Pointer to buffer which contains the images.

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

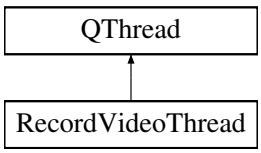
- ProcessingThread.h
- ProcessingThread.cpp

5.11 RecordVideoThread Class Reference

Record Video class which records the processed images from the webcam to a video file.

`#include <RecordVideoThread.h>`

Inheritance diagram for RecordVideoThread:



Public Member Functions

- RecordVideoThread (QString filename, int height, int width)
  Thread constructor class.
- void stop ()
  Set stop flag for thread worker.
- bool createVideoWriter ()
  Function to create a new video writer.

Protected Member Functions

- void run ()
  Thread function which runs till doStop is set to true.

Private Slots

- void addRecFrame (const Mat &frame)
  Private slot to add next image to video file.

Private Attributes

- QMutex doStopMutex

  QMutex.
- QMutex recordingMutex

  QMutex.
- volatile bool doStop

  Bool.
- VideoWriter vid_writer

  VideoWriter.
- QString fn

  QString.
- int rec_width

  Integer.
- int rec_height

  Integer.
- int rec_fps

  Integer.

5.11.1 Detailed Description

Record Video class which records the processed images from the webcam to a video file.

Thread to record processed images.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 RecordVideoThread::RecordVideoThread (QString *filename,* int *height,* int *width*)

Thread constructor class.

Parameters

filename	a QString argument - Filename for recording.
height	an integer argument with the height of the video.
width	an integer argument with the width of the video.

5.11.3 Member Function Documentation

5.11.3.1 void RecordVideoThread::addRecFrame (const Mat & *frame*) [private],[slot]

Private slot to add next image to video file.

Parameters

frame	a Mat image to add to recorded vide file.

5.11.3.2 bool RecordVideoThread::createVideoWriter ( )

Function to create a new video writer.

Returns

bool - True if successfull.

5.11.3.3 void RecordVideoThread::run ( ) [protected]

Thread function which runs till doStop is set to true.

To function is the worker of the thread. It does all the calls to detection functions and emits new images.

5.11.4 Member Data Documentation

5.11.4.1 volatile bool RecordVideoThread::doStop [private]

Bool.

Boolean value to stop the thread.

5.11.4.2 QMutex RecordVideoThread::doStopMutex [private]

QMutex.

Mutex to stop the thread.

5.11.4.3 QString RecordVideoThread::fn [private]

QString.

Filename of the recorded video.

5.11.4.4 int RecordVideoThread::rec_fps [private]

Integer.

Integer value with recording framerate.

5.11.4.5 int RecordVideoThread::rec_height [private]

Integer.

Integer value with recording height.

5.11.4.6 int RecordVideoThread::rec_width [private]

Integer.

Integer value with recording width.

5.11.4.7 QMutex RecordVideoThread::recordingMutex [private]

QMutex.

Mutex which is set while recording the image.

5.11.4.8 VideoWriter RecordVideoThread::vid_writer [private]

VideoWriter.

VideoWriter from OpenCV to record video to file.

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

- RecordVideoThread.h
- RecordVideoThread.cpp

5.12 VideoImageBuffer Class Reference

Video buffer class.

`#include <VideoImageBuffer.h>`

Public Member Functions

- VideoImageBuffer ()

  Constructor of VideoImageBuffer.
- void add (int deviceNumber, Buffer< Mat > *imageBuffer)

  Function to add a Image buffer.
- Buffer< Mat > * getByDeviceNumber (int deviceNumber)

  Get the buffer for specific device number.
- void removeByDeviceNumber (int deviceNumber)

  Remove a buffer for a specifc device number.
- void wakeAll ()

  Wake all QWaitConditions.
- bool containsImageBufferForDeviceNumber (int deviceNumber)

  Function to check if the Video image buffer contains a buffer for a device number.

Private Attributes

- QHash< int, Buffer< Mat > * > imageBufferMap

  QHash<int, Buffer<Mat>>*
- QSet< int > syncSet

  QSet<int>
- QWaitCondition wc

  QWaitCondition.
- QMutex mutex

  QMutex.

5.12.1 Detailed Description

Video buffer class.

Video buffer class to share images between two or more threads.

5.12.2 Member Function Documentation

5.12.2.1 void VideoImageBuffer::add (int *deviceNumber,* Buffer< Mat > * *imageBuffer*)

Function to add a Image buffer.

Parameters

deviceNumber	an integer argument with the device number.
imageBuffer	a pointer to the image buffer to add (Type Buffer<Mat>).

5.12.2.2 bool VideoImageBuffer::containsImageBufferForDeviceNumber (int *deviceNumber*)

Function to check if the Video image buffer contains a buffer for a device number.

Parameters

deviceNumber	an integer argument with the device number.

Returns

True if VideoImageBuffer contains buffer for specific device number

5.12.2.3 Buffer< Mat > * VideoImageBuffer::getByDeviceNumber (int *deviceNumber*)

Get the buffer for specific device number.

Parameters

deviceNumber	an integer argument with the device number.

Returns

Pointer the requested buffer of type Mat.

5.12.2.4 void VideoImageBuffer::removeByDeviceNumber (int *deviceNumber*)

Remove a buffer for a specifc device number.

Parameters

deviceNumber	an integer argument with the device number.

5.12.3 Member Data Documentation

5.12.3.1 QHash<int, Buffer<Mat>*> VideoImageBuffer::imageBufferMap [private]

QHash<int, Buffer<Mat>*>

A map which contains all pointers to the added buffers

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

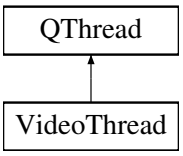
- VideoImageBuffer.h
- VideoImageBuffer.cpp

5.13 VideoThread Class Reference

Video class which cpatures an image from a webcam or specific device.

`#include <VideoThread.h>`

Inheritance diagram for VideoThread:



Public Member Functions

- VideoThread (VideoImageBuffer *videoImageBuffer, int deviceNumber, bool dropFrameIfBufferFull)
  Thread constructor class.
- void stop ()
  Set stop flag for thread worker.
- bool connectToCamera ()
  Function to create a camera connection.
- bool disconnectCamera ()
  Function to disconnect a camera connection.
- bool isCameraConnected ()
  Function to check if a camera is connected.
- int getInputSourceWidth ()
  Function to get the camera input width of an image.
- int getInputSourceHeight ()
  Function to get the camera input height of an image.

Protected Member Functions

- void run ()
  Thread function which runs till doStop is set to true.

Private Attributes

- VideoImageBuffer * videoImageBuffer
  VideoImageBuffer.
- VideoCapture cap
  VideoCapture.
- Mat grabbedFrame
  Mat image.
- QMutex doStopMutex
  QMutex.
- volatile bool doStop
  Bool.
- int deviceNumber
  Integer.
- bool dropFrameIfBufferFull
  Bool.

5.13.1 Detailed Description

Video class which cpatures an image from a webcam or specific device.

Thread to capture images.

5.13.2 Constructor & Destructor Documentation

5.13.2.1 VideoThread::VideoThread (VideoImageBuffer * *videoImageBuffer,* int *deviceNumber,* bool *dropFrameIfBufferFull*)

Thread constructor class.

Parameters

videoImageBuffer	pointer to a buffer which contains the captured images.
deviceNumber	an integer argument.
dropFrameIfBufferFull	a bool value to set if images shall be dropped when the buffer is full.

5.13.3 Member Function Documentation

5.13.3.1 bool VideoThread::connectToCamera ( )

Function to create a camera connection.

Returns

Returns true when attempt was successful.

5.13.3.2 bool VideoThread::disconnectCamera ( )

Function to disconnect a camera connection.

Returns

Returns true when attempt was successful.

5.13.3.3 int VideoThread::getInputSourceHeight ( )

Function to get the camera input height of an image.

Returns

Returns the height as an integer value.

5.13.3.4 int VideoThread::getInputSourceWidth ( )

Function to get the camera input width of an image.

Returns

Returns the width as an integer value.

5.13.3.5 bool VideoThread::isCameraConnected ( )

Function to check if a camera is connected.

Returns

Returns true when a camera is connected.

5.13.3.6 void VideoThread::run ( ) [protected]

Thread function which runs till doStop is set to true.

To function is the worker of the thread. It does all the calls to detection functions and emits new images.

5.13.4 Member Data Documentation

5.13.4.1 VideoCapture VideoThread::cap [private]

VideoCapture.

Variable to store the camera connection. From OpenCV lib.

5.13.4.2 int VideoThread::deviceNumber [private]

Integer.

Integer value for device number.

5.13.4.3 volatile bool VideoThread::doStop [private]

Bool.

Boolean value to stop the thread.

5.13.4.4 QMutex VideoThread::doStopMutex [private]

QMutex.

Mutex to stop the thread.

5.13.4.5 bool VideoThread::dropFrameIfBufferFull [private]

Bool.

Boolean value if a frame shall be dropped when the buffer is full.

5.13.4.6 Mat VideoThread::grabbedFrame [private]

Mat image.

Grabbed frame from input source.

5.13.4.7 VideoImageBuffer* VideoThread::videoImageBuffer [private]

VideoImageBuffer.

Pointer to buffer which contains the images.

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

- VideoThread.h
- VideoThread.cpp

Chapter 6

File Documentation

6.1 Config.h File Reference

Structure file. It contains the general stored settings which are loaded at the beginning.

Macros

- #define DEFAULT_IMAGE_BUFFER_SIZE 1
- #define OVERLAY_PARAM_1 0
- #define OVERLAY_PARAM_2 0
- #define OVERLAY_PARAM_3 0

6.1.1 Detailed Description

Structure file. It contains the general stored settings which are loaded at the beginning.

6.1.2 Macro Definition Documentation

6.1.2.1 #define DEFAULT_IMAGE_BUFFER_SIZE 1

Default image buffer size for the software

6.1.2.2 #define OVERLAY_PARAM_1 0

Standard value for reset functions. This values defines the glasses as overlay

6.1.2.3 #define OVERLAY_PARAM_2 0

Standard value for reset functions. This values defines the red nose as overlay

6.1.2.4 #define OVERLAY_PARAM_3 0

Standard value for reset functions. This values defines the red lips as overlay

6.2 ImageConversion.h File Reference

Image conversion functions. Four function to convert images between the following three types: OpenCV Mat, QImage, QPixmap.

```
#include <iostream>
#include <QImage>
#include <QPixmap>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/imgproc/types_c.h>
```

Functions

- QImage cvMatToQImage (const cv::Mat &inMat)

  Image conversion function which converts an OpenCV Mat image to an Qt QImage.
- cv::Mat QImageToCvMat (const QImage &inImage, bool inCloneImageData)

  Image conversion function which converts an OpenCV Mat image to an Qt QImage.
- QPixmap cvMatToQPixmap (const cv::Mat &inMat)

  Inline function for direct conversion from Mat to QPixmap format.
- cv::Mat QPixmapToCvMat (const QPixmap &inPixmap, bool inCloneImageData=true)

  Inline function for direct conversion from QPixmap to Mat format.

6.2.1 Detailed Description

Image conversion functions. Four function to convert images between the following three types: OpenCV Mat, QImage, QPixmap.

6.2.2 Function Documentation

6.2.2.1 QImage cvMatToQImage (const cv::Mat & *inMat*)

Image conversion function which converts an OpenCV Mat image to an Qt QImage.

Parameters

inMat	a Mat image to convert.

Returns

The converted image as QImage (Qt format).

6.2.2.2 QPixmap cvMatToQPixmap (const cv::Mat & *inMat*) [inline]

Inline function for direct conversion from Mat to QPixmap format.

Parameters

inMat	a Mat image to convert.

Returns

The converted image as QPixmap (Qt format).

6.2.2.3 cv::Mat QImageToCvMat (const QImage & *inImage*, bool *inCloneImageData*)

Image conversion function which converts an OpenCV Mat image to an Qt QImage.

Parameters

inImage	a QImage image to convert.
inCloneImage-Data	a boolean value if image should be cloned for conversion.

Returns

The converted image as Mat (OpenCV format).

6.2.2.4 cv::Mat QPixmapToCvMat (const QPixmap & *inPixmap*, bool *inCloneImageData* = `true`) [inline]

Inline function for direct conversion from QPixmap to Mat format.

Parameters

inPixmap	a QPixmap image to convert.
inCloneImage-Data	a boolean value if image should be cloned for conversion.

Returns

The converted image as Mat (OpenCV format).

6.3 Structures.h File Reference

Structures of the program. This file contains to structures to share settings and flags programm wide.

Classes

- struct ImageProcessingSettings

  Structure which contains information about the overlays apply on the image.
- struct ImageProcessingFlags

  Structure which contains information about the flags. Determine what process shall be done.

6.3.1 Detailed Description

Structures of the program. This file contains to structures to share settings and flags programm wide.

Index