Geko

Generated by Doxygen 1.8.6

Thu Dec 24 2015 17:13:04

Contents

1 Module Documentation

1.1 MagEngineAPI

Classes

- struct EventValue
- struct FilterParameter
- struct GekoWriterState

Macros

- #define UNAVAILABLE std::numeric_limits<int64_t>::min()

  minimum number of int64, used to indicate the video source access is unavailable
- #define NULLQUEUE -1

  error code for not set the maximum size of signal data queue
- #define DEFAULT_ALPHA 40

  Default alpha.
- #define DEFAULT_LOW_CUTOFF 0.4

Default low cutoff of the temporal filter.

- #define DEFAULT_HIGH_CUTOFF 2

  Default high cutoff of the temporal filter.

- #define DEFAULT_LAMBDA 72

  Default spatial frequency cutoff.

- #define DEFAULT_LEVEL 4

  Default level of Gaussian pyramid.

- #define DEFAULT_RADIAL 2

  Default number of octave in a spatial bands (used in steerable motion only)

- #define DEFAULT_ANGULAR 2

  Default number of orientation in steerable pyramid (used in steerable motion only)

- #define DEFAULT_SIGMA 5

  Default standard deviation (used in steerable motion only)

- #define DEFAULT_CUTOFF_BOUND 20

  Default filter cutoff maximum.

- #define DEFAULT_LAMBDA_BOUND 100

  Default spatial frequency maximum.

Typedefs

- typedef void(∗ CallBackFunction)(struct EventValue)

Enumerations

- enum MagEngineEvent {
  NONE_EVENT = -1, ORIGINAL_IMAGE = 0, PROCESSED_IMAGE, FACE_IMAGE,
  NO_FACE_IMAGE, FREQUENCY_NUMBER, SIGNAL_DATA, ROI_RESET,
  ROI_CHANGED, FILTER_CHANGED, VALUE_ALPHA, VALUE_LOWCUT,
  VALUE_HIGHCUT, VALUE_LAMBDA, VALUE_LEVEL, VALUE_RADIAL,
  VALUE_ANGULAR, VALUE_SIGMA, VALUE_SOURCE_FRAME_RATE, VALUE_PROCESS_FRAME_RATE,
  VALUE_FILE_DURATION, VALUE_FILE_PROGRESS, VALUE_CAMERA_COUNT, AUTOFOCUS_AVAILABLE,
  AUTOFOCUS_ENABLED, OPEN_CAMERA_SUCCESS, OPEN_CAMERA_FAIL, OPEN_FILE_SUCCESS,
  OPEN_FILE_FAIL, OPEN_EXTERNAL_SOURCE_SUCCESS, OPEN_EXTERNAL_SOURCE_FAIL, CLOSE_SOURCE_SUCCESS,
  FILE_ARCHIVED, END_OF_FILE }
- enum ProcessMethod { LINEAR_MOTION = 0, LINEAR_COLOR, STEERABLE_MOTION, RIESZ_MOTION }
- enum SourceOperation { CLOSE_SOURCE, OPEN_CAMERA_SOURCE, OPEN_FILE_SOURCE, OPEN_EXTERNAL_SOURCE }
- enum RoiMode { MANUAL_ROI = 0, FACE_DETECT, FACE_TRACK, MARKER_DETECT }
- enum SourceAccess {
  CAMERA_EXPOSURE = 0, CAMERA_AUTO_FOCUS, CAMERA_FRAME_RATE, CAMERA_BRIGHTNESS,
  CAMERA_CONTRAST, CAMERA_SATURATION, CAMERA_AUTO_WHITEBALANCE, CAMERA_BACKLIGHT_COMPENSATION,
  CAMERA_ABSOLUTE_FOCUS, CAMERA_ZOOM_IN, CAMERA_ZOOM_OUT, CAMERA_ZOOM_RESET,
  CAMERA_PAN_LEFT, CAMERA_PAN_RIGHT, CAMERA_PAN_RESET, CAMERA_TILT_UP,
  CAMERA_TILT_DOWN, CAMERA_TILT_RESET, CAMERA_PTZ_RESET, FILE_SEEK = 100,
  FILE_PROGRESS, FILE_DURATION, FILE_PLAY, FILE_PAUSE,
  FILE_FORWARD, FILE_BACKWARD, FILE_STOP, FILE_NORNAL_PLAYBACK,
  FILE_TURBO_PLAYBACK, IMAGE_FLIP = 200 }

- enum ParameterType {
  ALPHA = 0, HIGHCUT, LOWCUT, LEVEL,
  LEVEL_BOUND, LAMBDA, LAMBDA_BOUND, RADIAL_OCTAVE,
  ANGULAR_ORDER, SIGMA, FRAMERATE, CUTOFF_BOUND }
- enum LicenseResponse {
  LICENSE_OFFLINE_WARNING = -2, LICENSE_OFFLINE_SUCCESS = -1, LICENSE_SUCCESS = 0, LICENSE_OFFLINE_FORBIDDEN,
  LICENSE_MISSING, LICENSE_INCOMPLETE, LICENSE_EXPIRED }
- enum RecordType { RECORD_NONE = 0, RECORD_ORIGINAL_ONLY, RECORD_PROCESSED_ONLY, RECORD_BOTH }

Functions

- enum LicenseResponse initMagEngine (CallBackFunction cb)

  Initialize function.
- void destroyMagEngine ()

  Release function.
- void setProcessMethod (enum ProcessMethod processMethod)

  Algorithm setting function.
- void setSource (enum SourceOperation operation, const char *source)

  Video source setting.
- void setOutput (const char *output)

  Output filename setting.
- void setWebBackendMode (bool webBackendFlag)

  Toggle web backend mode.
- bool controlSource (enum SourceAccess type, int value=0)

  Camera or file control.
- int getCameraCount ()

  Get the number of connected cameras.
- int64_t querySource (enum SourceAccess type)

  Camera or file query.
- void setROIMode (enum RoiMode mode)

  Region of interest method setting.
- enum RoiMode getROIMode ()

  Get the current ROI mode.
- void setROIView (uint16_t x, uint16_t y, uint16_t width, uint16_t height)

  Set Region of interest.
- void setFilterParameter (enum ParameterType type, float value)

  Filter parameter setting.
- void setFaceDectionXML (std::string xmlFile)

  Face detection XML selection.
- std::string getFaceDectionXML ()

  Get the name of the selected XML file.
- void setFileOutput (enum RecordType recordType)

  Enable or disable file output.
- enum ProcessMethod getProcessMethod ()

  Get the current process method.
- std::string getSourceName ()

  Get the video source name.
- struct FilterParameter getFilterParameter ()

  Get the FilterParameter structure.
- struct GekoWriterState getFileOutputState ()

Get the current info about writer.

- void setDataQueueLength (size_t len)

  set the miximum data queue length for rate calculation

- size_t getDataQueueLength ()

  get the currently maximum data queue length

- time_t **getExpiredDate** ()
- int **getVerificatioErrorCode** ()
- std::string **MAG_ENGINE_VERSION** ()
- const char * **MAG_ENGINE_VERSION_DATE** ()

1.1.1 Detailed Description

Author

Nathan Chen

Date

23 Sep 2015

The API header file, define API functions, callback event and operations

1.1.2 Typedef Documentation

1.1.2.1 typedef void(* CallBackFunction)(struct EventValue)

Callback function

1.1.3 Enumeration Type Documentation

1.1.3.1 enum MagEngineEvent

Here is the list of event that will be evoked in Geko, to catch the value and image you are interested in, implement the event handler like we did in xxx.cpp

See Also

xxx.cpp

Enumerator

NONE_EVENT none event

ORIGINAL_IMAGE Evoked when source image is captured

PROCESSED_IMAGE Evoked when processed image is done

FACE_IMAGE Evoked when face detection success, return the captured face image

NO_FACE_IMAGE Evoked when face detection fail, return a "no face" image

FREQUENCY_NUMBER Evoked when frequency result had been estimated, NOTE: won't evoked in default, see setDataQueueLength

SIGNAL_DATA Evoked when signal extracted, return the signal queue, NOTE: won't evoked in default, see setDataQueueLength

ROI_RESET Evoked when region of interest had been changed

ROI_CHANGED Evoked when ROI MODE had been changed

FILTER_CHANGED Evoked when filter method had been changed

VALUE_ALPHA Evoked when magnification value had been changed

VALUE_LOWCUT Evoked when filter low cutoff had been changed

VALUE_HIGHCUT Evoked when filter high cutoff had been changed

VALUE_LAMBDA Evoked when spatial frequency cutoff had been changed (used in linear motion only)

VALUE_LEVEL Evoked when Gaussian pyramid level had been changed (used in linear color only)

VALUE_RADIAL Evoked when radial octave had been changed (used in steerable motion only)

VALUE_ANGULAR Evoked when angular order had been changed (used in steerable motion only)

VALUE_SIGMA Evoked when Gaussian standard deviation had been changed (used in steerable motion only)

VALUE_SOURCE_FRAME_RATE Evoked when source frame rate had been changed

VALUE_PROCESS_FRAME_RATE Evoked when processing frame rate had been changed

VALUE_FILE_DURATION Evoked when get video file length

VALUE_FILE_PROGRESS Evoked during file processing, return the timestamp in video file

VALUE_CAMERA_COUNT Evoked when camera count had been changed

AUTOFOCUS_AVAILABLE Evoked after source opened, indicate whether the source can do auto focus

AUTOFOCUS_ENABLED Evoked after source opened, indicate whether auto focus is enabled

OPEN_CAMERA_SUCCESS Evoked when webcam opened successful

OPEN_CAMERA_FAIL Evoked when webcam opened fail

OPEN_FILE_SUCCESS Evoked when file opened successful

OPEN_FILE_FAIL Evoked when file opened fail

OPEN_EXTERNAL_SOURCE_SUCCESS Evoked when external source opened successful

OPEN_EXTERNAL_SOURCE_FAIL Evoked when external source opened fail

CLOSE_SOURCE_SUCCESS Evoked when source closed successful

FILE_ARCHIVED Evoked when file archived

END_OF_FILE Evoked when reach to end of file

1.1.3.2 enum ProcessMethod

Process Algorithm

Enumerator

LINEAR_MOTION Eulerian linear motion amplification method

LINEAR_COLOR Eulerian linear color amplification method

STEERABLE_MOTION Eulerian phase based motion amplification method, process in frequency domain

RIESZ_MOTION Eulerian phase based motion amplification method, process in time domain

1.1.3.3 enum SourceOperation

Video Source

Enumerator

CLOSE_SOURCE Close video source

OPEN_CAMERA_SOURCE Open webcam source

OPEN_FILE_SOURCE Open file source

OPEN_EXTERNAL_SOURCE Open user define source

1.1.3.4 enum RoiMode

Region of Interest Mode Different way to extract ROI from a image

Enumerator

MANUAL_ROI User can assign a ROI in the image

FACE_DETECT Doing face detection first, then locate the fore head region as ROI

FACE_TRACK Same as FACE_DETECT, plus tracking technique

MARKER_DETECT Doing marker detection, then locate the bottom of marker as ROI
The marker is rotation invariant, please refer to ArUco website

See Also

http://www.uco.es/investiga/grupos/ava/node/26

1.1.3.5 enum SourceAccess

Video source control/query type, only some of them are available

Enumerator

CAMERA_EXPOSURE camera exposure access

CAMERA_AUTO_FOCUS camera auto focus access

CAMERA_FRAME_RATE camera frame rate access

CAMERA_BRIGHTNESS camera brightness access

CAMERA_CONTRAST camera contrast access

CAMERA_SATURATION camera saturation access

CAMERA_AUTO_WHITEBALANCE camera whitebalance access

CAMERA_BACKLIGHT_COMPENSATION camera backlight compensation access

CAMERA_ABSOLUTE_FOCUS camera absolute focus access

CAMERA_ZOOM_IN camera zoom in

CAMERA_ZOOM_OUT camera zoom out

CAMERA_ZOOM_RESET camera zoom reset

CAMERA_PAN_LEFT camera pan left

CAMERA_PAN_RIGHT camera pan right

CAMERA_PAN_RESET camera pan reset

CAMERA_TILT_UP camera tilt up

CAMERA_TILT_DOWN camera tilt down

CAMERA_TILT_RESET camera tilt reset

CAMERA_PTZ_RESET reset pan, tilt and zoom

FILE_SEEK Video file seek to some timestamp

FILE_PROGRESS Video file current progress

FILE_DURATION Video file length

FILE_PLAY Play video file

FILE_PAUSE Pause video file

FILE_FORWARD Foward video file

FILE_BACKWARD Backward video file

FILE_STOP Stop video file

FILE_NORNAL_PLAYBACK Video file normal playback

FILE_TURBO_PLAYBACK Video file turbo playback

IMAGE_FLIP Flip the input image

1.1.3.6 enum ParameterType

Algorithm parameters type

Enumerator

ALPHA the amplification factor

HIGHCUT high cutoff of temporal filter

LOWCUT low cutoff of temporal filter

LEVEL Gaussian pyramid level (used in linear color only)

LEVEL_BOUND Maximum value for level

LAMBDA spatial frequency cutoff (used in linear motion only)

LAMBDA_BOUND Maximum value for lambda

RADIAL_OCTAVE number of octave in a spatial bands (used in steerable motion only)

ANGULAR_ORDER number of orientation in the steerable pyramid (used in steerable motion only)

SIGMA the Gaussian standard deviation (used in steerable motion only)

FRAMERATE Processing frame rate

CUTOFF_BOUND Maximum value for temporal filter cutoff

1.1.3.7 enum LicenseResponse

License verification response type

Enumerator

LICENSE_OFFLINE_WARNING license passed offline verification, but it has to finish an online verification in one month

LICENSE_OFFLINE_SUCCESS license passed offline verification

LICENSE_SUCCESS license is valid

LICENSE_OFFLINE_FORBIDDEN license must take an online verification

LICENSE_MISSING can not find any license

LICENSE_INCOMPLETE incomplete license

LICENSE_EXPIRED license is invalid

1.1.3.8 enum RecordType

Video record type

Enumerator

RECORD_NONE Don't record video

RECORD_ORIGINAL_ONLY Only record original video

RECORD_PROCESSED_ONLY Only record processed video

RECORD_BOTH Record both videos

1.1.4 Function Documentation

1.1.4.1 enum LicenseResponse initMagEngine (CallBackFunction *cb*)

Initialize function.

be sure to call this function before doing anything

Parameters

cb	pass your CallBackFunction as a parameter

1.1.4.2 void destroyMagEngine ( )

Release function.

be sure to call this function at the end of program to release the pointer

1.1.4.3 void setProcessMethod (enum **ProcessMethod** *processMethod*)

Algorithm setting function.

set the processing algorithm you want to use

Parameters

processMethod	the process algorithm

1.1.4.4 void setSource (enum **SourceOperation** *operation,* const char * *source*)

Video source setting.

set the video source from a file or webcam

if the device is a webcam, use videoX, X is the camera index

Parameters

operation	specify the SourceOperation
source	the name of the source (file name or device name)

1.1.4.5 void setOutput (const char * *output*)

Output filename setting.

set the filename of output file

Parameters

output	specify the output filename

1.1.4.6 void setWebBackendMode (bool *webBackendFlag*)

Toggle web backend mode.

use this function to toggle file output mode

Parameters

webBackend-Flag	specify whether to run as web backend mode

1.1.4.7 bool controlSource (enum **SourceAccess** *type,* int *value* = 0)

Camera or file control.

use this function to control camera or file source

Parameters

type	specify control type SourceAccess
value	specify the value

Returns

whether the control is successful

1.1.4.8 int getCameraCount ( )

Get the number of connected cameras.

Returns

the number of connected cameras

1.1.4.9 int64_t querySource (enum SourceAccess *type*)

Camera or file query.

query statics from video source

Parameters

type	specify query type

Returns

the current value of the query type is return, if the type is unaccessible than UNAVAILABLE is return

1.1.4.10 void setROIMode (enum RoiMode *mode*)

Region of interest method setting.

set to the specified ROI method

Parameters

mode	the ROI method

1.1.4.11 enum RoiMode getROIMode ( )

Get the current ROI mode.

Returns

the current chosen ROI mode

1.1.4.12 void setROIView (uint16_t *x,* uint16_t *y,* uint16_t *width,* uint16_t *height*)

Set Region of interest.

only available in MANUAL_ROI mode

Parameters

x	the x coordinate of the top-left corner of ROI
y	the y coordinate of the top-left corner of ROI
width	the width of ROI
height	the height of ROI

1.1.4.13 void setFilterParameter (enum **ParameterType** *type,* float *value*)

Filter parameter setting.

select a filter parameter to set to target value

Parameters

type	specify parameter type
value	target value

1.1.4.14 void setFaceDectionXML (std::string *xmlFile*)

Face detection XML selection.

Select which XML file you want to use for OpenCV face detection, they can be found in data/ folder

Parameters

xmlFile	the file name of the XML file

1.1.4.15 std::string getFaceDectionXML ( )

Get the name of the selected XML file.

Returns

the name of the current XML file

1.1.4.16 void setFileOutput (enum **RecordType** *recordType*)

Enable or disable file output.

set to RECORD_NONE if you don't want to store any file

set to RECORD_PROCESSED_ONLY if you want to record processed result only

set to RECORD_ORIGINAL_ONLY if you want to record original source only

set to RECORD_BOTH if you want to record both

Parameters

recordType	specify the record type

1.1.4.17 enum **ProcessMethod** getProcessMethod ( )

Get the current process method.

retrun the algoeithm Geko is using now

Returns

the current chosen algorithm

1.1.4.18 std::string getSourceName ( )

Get the video source name.

return file name when source is file

return videoX when source is webcam

Returns

current source name

1.1.4.19 struct FilterParameter getFilterParameter ( )

Get the FilterParameter structure.

the FilterParameter structure contains the current parameter values

Returns

the current filter parameter struct

1.1.4.20 struct GekoWriterState getFileOutputState ( )

Get the current info about writer.

Returns

the current writer state struct

1.1.4.21 void setDataQueueLength (size_t *len*)

set the miximum data queue length for rate calculation

set the maximum data length you want to use for rate calculation

NOTE 1: You have to use this function or the getDataQueueLength function at least once to enable the rate calculation function and callback

NOTE 2: Since we use DFT to calculate rate, the performance is best if len = 2^a∗3^b∗5^c...,

if your len didn't follow the rule, we will calculate an optimal size greater than or equal to len for you

Parameters

len	set the amount of data to calculate rate

1.1.4.22 size_t getDataQueueLength ( )

get the currently maximum data queue length

return the currently maximum data queue length

if you didn't use setDataQueueLength before, the function will return NULLQUEUE

otherwisw it will return the optimaized length 2^a∗3^b∗5^c...

Returns

the maximum data queue length

1.2 MediaSourceBase

Classes

- class MediaSourceBase

1.2.1 Detailed Description

Author

Vince

Date

27 Feb 2014

The base class for media source

please implement this class if you want to use your own source

2 Class Documentation

2.1 EventValue Struct Reference

`#include <MagEngineAPI.h>`

Public Types

- enum ValueType {
  NonValue, **IntegerValue**, **FloatValue**, **StringValue**,
  FrameValue, **FloatDataQueue** }

Public Member Functions

- EventValue (enum MagEngineEvent evt=NONE_EVENT)

  Constructor for no value event type.
- EventValue (enum MagEngineEvent evt, bool i)

  Constructor for boolean type.
- EventValue (enum MagEngineEvent evt, int i)

  Constructor for integer type.
- EventValue (enum MagEngineEvent evt, size_t i)

  Constructor for unsigned integer type.
- EventValue (enum MagEngineEvent evt, int64_t i)

  Constructor for 64bits integer type.
- EventValue (enum MagEngineEvent evt, float d)

  Constructor for floating point type.
- EventValue (enum MagEngineEvent evt, double d)

  Constructor for double floating point type.
- EventValue (enum MagEngineEvent evt, std::string s)

  Constructor for string type.
- EventValue (enum MagEngineEvent evt, char *data, uint32_t len, uint16_t w, uint16_t h)

  Constructor for image buffer type.
- EventValue (enum MagEngineEvent evt, float *data, uint32_t len)

  Constructor for signal data queue.

Public Attributes

- enum MagEngineEvent event
- enum EventValue::ValueType **valueType**
- std::string sValue
- union {
  double dValue
  int64_t iValue
  };
- void * dataBuffer
- uint16_t imageWidth
- uint16_t imageHeight

2.1.1 Detailed Description

This is the structure of callback event value

Callback event have different value types

You have to check the value type before any access

or you may get wrong value (e.g. the callback value is string but you accsee integer)

For implementation please refer to LambdaVueConsole.cpp

See Also

LambdaVueConsole.cpp

2.1.2 Member Enumeration Documentation

2.1.2.1 enum EventValue::ValueType

The callback event value type, Always check value type before any accessing

2.1.3 Member Data Documentation

2.1.3.1 enum MagEngineEvent EventValue::event

Event index, indicate the type of event

2.1.3.2 std::string EventValue::sValue

String value type

2.1.3.3 double EventValue::dValue

Double value type

2.1.3.4 int64_t EventValue::iValue

int64_t value type

2.1.3.5 void* EventValue::dataBuffer

pointer to data buffer, can be a image or a queue of signal data

2.1.3.6 uint16_t EventValue::imageWidth

image width

2.1.3.7 uint16_t EventValue::imageHeight

image height

2.2 FilterParameter Struct Reference

`#include <MagEngineAPI.h>`

Public Attributes

- size_t **alpha**

- float **highcut**
- float **lowcut**
- float **lambda**
- size_t **level**
- size_t **radialOctave**
- size_t **angularOrder**
- size_t **gaussianSigma**
- float **framerate**
- float **lambdaBound**
- float **cutOffBound**

2.2.1 Detailed Description

Algorithm parameters structure, store all the parameter value

2.3 GekoWriterState Struct Reference

`#include <MagEngineAPI.h>`

Public Attributes

- enum RecordType **recordType**
- int processedWidth
- int processedHeight
- std::string processedDestFileName
- int originalWidth
- int originalHeight
- std::string originalDestFileName

2.3.1 Detailed Description

Current video writer status

2.3.2 Member Data Documentation

2.3.2.1 int GekoWriterState::processedWidth

processed video width

2.3.2.2 int GekoWriterState::processedHeight

processed video height

2.3.2.3 std::string GekoWriterState::processedDestFileName

name of the processed video file

2.3.2.4 int GekoWriterState::originalWidth

original video width

2.3.2.5 int GekoWriterState::originalHeight

original video height

2.3.2.6 std::string GekoWriterState::originalDestFileName

name of the original video file

2.4 MediaSourceBase Class Reference

Public Member Functions

- virtual void destroy ()=0

  Destroy this media source.
- virtual std::string getSourceName ()=0

  Get the video source name.
- virtual bool switchSource (const char *sourceURL)=0

  Switch to the specified source name.
- virtual void closeSource ()=0

  Close this media source.
- virtual void getImage (char *buf, uint32_t &len, uint16_t &width, uint16_t &height, int64_t &timestamp)=0

  Assign your frame data so Geko can process it.
- virtual size_t getFrameCount ()=0

  Return the amount of frame in this media source.
- virtual size_t getWidth ()=0

  The image width of this media source.
- virtual size_t getHeight ()=0

  The image height of this media source.
- virtual float getFrameRate ()

  Return the frame rate of this media source.
- virtual int getCameraCount ()

  Return the number of connected cameras.
- virtual bool isValid ()=0

  Return whether the source is valid.
- virtual bool isFile ()

  Return whether this media source is a file.
- virtual int64_t query (enum geko::SourceAccess)

  Return the specify info of this media source.
- virtual bool control (enum geko::SourceAccess, int)

  Set the specify control type of media source.

2.4.1 Member Function Documentation

2.4.1.1 virtual void MediaSourceBase::destroy ( ) [pure virtual]

Destroy this media source.

implement how your media source should be destroyed

2.4.1.2 virtual std::string MediaSourceBase::getSourceName ( ) [pure virtual]

Get the video source name.

Returns

the name of this media source

2.4.1.3 virtual bool MediaSourceBase::switchSource (const char * *sourceURL*) [pure virtual]

Switch to the specified source name.

implement the way to switch your media source from current video source to the specified video source

if the device is a webcam, its name should be videoX, where X is the camera index

Parameters

sourceURL	the name of the source (file name or device name)

Returns

whether the switch is successful

2.4.1.4 virtual void MediaSourceBase::closeSource ( ) [pure virtual]

Close this media source.

implement how your media source should be closed

2.4.1.5 virtual void MediaSourceBase::getImage (char * *buf,* uint32_t & *len,* uint16_t & *width,* uint16_t & *height,* int64_t & *timestamp*) [pure virtual]

Assign your frame data so Geko can process it.

put your input image data in framebuf, and specify its width, height and timestamp

Parameters

framebuf	put your image data in here, the image color space should be YUV420
width	the width of the image
height	the height of the image
timestamp	image timestamp, used for file recording

2.4.1.6 virtual size_t MediaSourceBase::getFrameCount ( ) [pure virtual]

Return the amount of frame in this media source.

Returns

the total number of frame in this media source

2.4.1.7 virtual size_t MediaSourceBase::getWidth ( ) [pure virtual]

The image width of this media source.

Returns

the image width

2.4.1.8 virtual size_t MediaSourceBase::getHeight ( ) [pure virtual]

The image height of this media source.

Returns

the image height

2.4.1.9 virtual float MediaSourceBase::getFrameRate ( ) [virtual]

Return the frame rate of this media source.

Returns

the frame rate

2.4.1.10 virtual int MediaSourceBase::getCameraCount ( ) [virtual]

Return the number of connected cameras.

Returns

the number of connected cameras

2.4.1.11 virtual bool MediaSourceBase::isValid ( ) [pure virtual]

Return whether the source is valid.

implement your method to check whether this media source is valid or not

Returns

whether this media source is valid

2.4.1.12 virtual bool MediaSourceBase::isFile ( ) [virtual]

Return whether this media source is a file.

Returns

true if this source is a file, false otherwise

2.4.1.13 virtual int64_t MediaSourceBase::query (enum geko::SourceAccess) [virtual]

Return the specify info of this media source.

please define how your source will return the source information define in SourceAccess

Returns

should return the value of the specified control type, please return UNAVAILABLE when the access is unavailable

2.4.1.14 virtual bool MediaSourceBase::control (enum geko::SourceAccess , int) [virtual]

Set the specify control type of media source.

please define how to set your media source

Returns

should return whether the control is successful

Index