video.h File Reference

ARToolkit video subroutines. More...

```
#include <AR/config.h>
#include <AR/ar.h>
```

Functions

AR_DLL_API int	**arVideoDispOption** (void) *display the video option.*
AR_DLL_API int	**arVideoOpen** (char *config) *open a video source.*
AR_DLL_API int	**arVideoClose** (void) *close the video source. After your application has finished using a video stream, this function must be called to close the link to the input source, and free resources associated with the capture operation.*
AR_DLL_API int	**arVideoCapStart** (void) *start the capture of video.*
AR_DLL_API int	**arVideoCapStop** (void) *stop the capture of video.*
AR_DLL_API int	**arVideoCapNext** (void) *call for the next grabbed video frame.*
AR_DLL_API ARUint8 *	**arVideoGetImage** (void) *get the video image.*
AR_DLL_API int	**arVideoInqSize** (int *x, int *y) *get the video image size, in pixels.*
AR_DLL_API int	**ar2VideoDispOption** (void) *display the video option (multiple video inputs)*
AR_DLL_API AR2VideoParamT *	**ar2VideoOpen** (char *config) *open a video source (multiple video inputs)*
AR_DLL_API int	**ar2VideoClose** (AR2VideoParamT *vid) *close a video source (multiple video inputs)*
AR_DLL_API int	**ar2VideoCapStart** (AR2VideoParamT *vid) *start the capture of a video source (multiple video inputs)*
AR_DLL_API int	**ar2VideoCapNext** (AR2VideoParamT *vid) *call for the next grabbed video frame of a video source (multiple video inputs)*
AR_DLL_API int	**ar2VideoCapStop** (AR2VideoParamT *vid) *stop the capture of a video source (multiple video inputs)*
AR_DLL_API ARUint8 *	**ar2VideoGetImage** (AR2VideoParamT *vid) *get a video image from a video source (multiple video inputs)*
AR_DLL_API int	**ar2VideoInqSize** (AR2VideoParamT *vid, int *x, int *y) *get the video image size of a video source (multiple video inputs)*

Detailed Description

ARToolkit video subroutines.

This library provides multi-platform video input support for ARToolKit. It abstracts access to hardware video input available on different machines and operating systems.

The actual supported platforms (and the driver/library used) are:

- Windows: with Microsoft DirectShow (VFW obsolete).
- Linux: with Video4Linux library, GStreamer, IEEE1394 camera library and DV camera library.
- Macintosh: with QuickTime.
- SGI: with VL.

This library provides two sets of functions, depending on whether your program needs to use only one video stream, or more than one video stream. These two sets are functionally identical.

- one camera: use the **arVideo*** functions.
- multiple cameras: use the **ar2Video*** functions.

More information on establishing video streams is available in the ARToolKit manual.

Remarks:
The arVideo* functions maintain the current video stream in a global variable and internally call the ar2Video* functions.

History : modified by Thomas Pintaric (pintaric@ims.tuwien.ac.at) to add a fully transparent DirectShow Driver. modified by Hartmut Seichter (hartmut@technotecture.com) to add GStreamer video support

Author:
Hirokazu Kato kato@sys.im.hiroshima-cu.ac.jp

Atsishi Nakazawa nakazawa@inolab.sys.es.osaka-u.ac.jp

Thomas Pintaric pintaric@ims.tuwien.ac.at (Windows DirectShow video support).

Philip Lamb phil@eden.net.nz (Macintosh Quicktime video support).

Hartmut Seichter hartmut@technotecture.com (GStreamer Video support)

Version:
4.3b

Date:
03/02/02

Function Documentation

AR_DLL_API int ar2VideoCapNext(AR2VideoParamT * *vid*)

call for the next grabbed video frame of a video source (multiple video inputs)

Companion function to arVideoCapNext for multiple video sources.

Parameters:
vid a video handle structure for multi-camera grabbing

AR_DLL_API int ar2VideoCapStart(AR2VideoParamT * *vid*)

start the capture of a video source (multiple video inputs)

Companion function to arVideoCapStart for multiple video sources.

Parameters:
vid a video handle structure for multi-camera grabbing

AR_DLL_API int ar2VideoCapStop(AR2VideoParamT * *vid*)

stop the capture of a video source (multiple video inputs)

Companion function to arVideoCapStop for multiple video sources.

Parameters:
vid a video handle structure for multi-camera grabbing

AR_DLL_API int ar2VideoClose(AR2VideoParamT * *vid*)

close a video source (multiple video inputs)

Companion function to arVideoClose for multiple video sources.

Parameters:
vid a video handle structure for multi-camera grabbing.

AR_DLL_API int ar2VideoDispOption(void)

display the video option (multiple video inputs)

Companion function to arVideoDispOption, for multiple video sources.

AR_DLL_API ARUint8* ar2VideoGetImage(AR2VideoParamT * *vid*)

get a video image from a video source (multiple video inputs)

Companion function to arVideoGetImage for multiple video sources.

Parameters:
vid a video handle structure for multi-camera grabbing

```
AR_DLL_API int ar2VideoInqSize( AR2VideoParamT * vid,
                                int *            x,
                                int *            y
                              )
```

get the video image size of a video source (multiple video inputs)

Companion function to arVideoInqSize for multiple video sources.

Parameters:
vid a video handle structure for multi-camera grabbing

AR_DLL_API AR2VideoParamT* ar2VideoOpen(char * *config*)

open a video source (multiple video inputs)

Companion function to arVideoOpen for multiple video sources. This function can be called multiple times to open multiple video streams. The maximum number of streams is dependent on the operating system and the performance characteristics of the host CPU and video capture infrastructure.

Parameters:

config string of the selected video configuration.

Returns:

If the video path was successfully opened, this function returns a pointer to an AR2VideoParamT structure, an opaque structure which holds information and configuration for the video stream. This paramater should then be passed to other ar2Video* functions to specify which video stream is being operated upon. If the video path was not successfully opened, NULL will be returned. s

AR_DLL_API int arVideoCapNext(void)

call for the next grabbed video frame.

This function should be called at least once per frame. It has several purposes, depending on the operating system. It allows the video driver to perform housekeeping tasks and also signals to the video grabber that your code has finished using the most recent video frame returned by **arVideoGetImage()**, and that the video driver may re-use the memory occupied by the frame. The effect of this call is operating-system dependent. The best place to call this function is immediately after you have finished displaying the current video frame, i.e. after calling **arglDispImage()** or **argDispImage()**.

Remarks:

On some operating systems, this function is a no-op.

Returns:

0 if successful, -1 if the video driver encountered an error.

AR_DLL_API int arVideoCapStart(void)

start the capture of video.

This function starts the video capture routine.

Remarks:

On some operating systems, capture operations may run in a separate execution thread. This call starts that thread.

this function coupled with arVideoCapStop, can be call many times in your program (this may reduce the CPU load when video processing is stopped or for long and critical operations).

Returns:

0 if successful, -1 if the capture couldn't be started.

AR_DLL_API int arVideoCapStop(void)

stop the capture of video.

This function stops the video capture routine.

Remarks:

On some operating systems, capture operations may run in a separate execution thread. This call stops that thread.

this function coupled with arVideoCapStart, can be call many times in your program (this may reduce the CPU load when video processing is stopped or for long and critical operations).

Returns:

0 if successful, -1 if the capture couldn't be stopped.

AR_DLL_API int arVideoClose(void)

close the video source. After your application has finished using a video stream, this function must be called to close the link to the input source, and free resources associated with the capture operation.

Returns:

0 if shut down successfully, otherwise -1.

AR_DLL_API int arVideoDispOption(void)

display the video option.

The video configuration options vary by operating system and platform. This function outputs to the standard output the options available on the current OS and platform.

Returns:

0

AR_DLL_API ARUint8* arVideoGetImage(void)

get the video image.

This function returns a buffer with a captured video image. The returned data consists of a tightly-packed array of pixels, beginning with the first component of the leftmost pixel of the topmost row, and continuing with the remaining components of that pixel, followed by the remaining pixels in the topmost row, followed by the leftmost pixel of the second row, and so on. The arrangement of components of the pixels in the buffer is determined by the configuration string passed in to the driver at the time the video stream was opened. If no pixel format was specified in the configuration string, then an operating- system dependent default, defined in <AR/config.h> is used. The memory occupied by the pixel data is owned by the video driver and should not be freed by your program. The pixels in the buffer remain valid until the next call to arVideoCapNext, or the next call to arVideoGetImage which returns a non-NULL pointer, or any call to arVideoCapStop or arVideoClose.

Returns:
A pointer to the pixel data of the captured video frame, or NULL if no new pixel data was available at the time of calling.

```
AR_DLL_API int arVideoInqSize( int * x,
                               int * y
                             )
```

get the video image size, in pixels.

This function returns the size of the captured video frame, in pixels.

Parameters:

x	a pointer to the length of the captured image
y	a pointer to the width of the captured image

Returns:
0 if the dimensions are found successfully, otherwise -1

```
AR_DLL_API int arVideoOpen( char * config )
```

open a video source.

This function opens a video input path with the driver (and device) present on your platform. According to your operating system and the hardware the initialization will be different : a generic string structure is used for this issue. This function prepares the video stream for capture, but capture will not actually begin until arVideoCapStart is called.

Parameters:

config	string of the selected video configuration. See the video configuration documentation for more information on this parameter.

Returns:
0 if successful, -1 if a video path couldn't be opened