cv::VideoCapture Class Reference

Media I/O

Class for video capturing from video files, image sequences or cameras. The class provides C++ API for capturing video from cameras or for reading video files and image sequences. Here is how the class can be used: : More...

```
#include "videoio.hpp"
```

Public Member Functions

	VideoCapture ()
	VideoCapture (const **String** &filename)
	VideoCapture (const **String** &filename, int apiPreference)
	VideoCapture (int index)
virtual	**~VideoCapture** ()
virtual double	**get** (int propId) const Returns the specified **VideoCapture** property. More...
virtual bool	**grab** () Grabs the next frame from video file or capturing device. More...
virtual bool	**isOpened** () const Returns true if video capturing has been initialized already. More...
virtual bool	**open** (const **String** &filename) Open video file or a capturing device for video capturing. More...
virtual bool	**open** (int index)
virtual bool	**open** (const **String** &filename, int apiPreference)
virtual **VideoCapture** &	**operator>>** (**Mat** &image)
virtual **VideoCapture** &	**operator>>** (**UMat** &image)
virtual bool	**read** (**OutputArray** image) Grabs, decodes and returns the next video frame. More...
virtual void	**release** ()

	Closes video file or capturing device. More...
virtual bool	**retrieve** (**OutputArray** image, int flag=0) Decodes and returns the grabbed video frame. More...
virtual bool	**set** (int propId, double value) Sets a property in the **VideoCapture**. More...

Protected Attributes

Ptr< **CvCapture** >	**cap**
Ptr< IVideoCapture >	**icap**

Detailed Description

Class for video capturing from video files, image sequences or cameras. The class provides C++ API for capturing video from cameras or for reading video files and image sequences. Here is how the class can be used: :

```
#include "opencv2/opencv.hpp"

using namespace cv;

int main(int, char**)
{
    VideoCapture cap(0); // open the default camera
    if(!cap.isOpened())  // check if we succeeded
        return -1;

    Mat edges;
    namedWindow("edges",1);
    for(;;)
    {
        Mat frame;
        cap >> frame; // get a new frame from camera
        cvtColor(frame, edges, COLOR_BGR2GRAY);
        GaussianBlur(edges, edges, Size(7,7), 1.5, 1.5);
        Canny(edges, edges, 0, 30, 3);
        imshow("edges", edges);
        if(waitKey(30) >= 0) break;
    }
    // the camera will be deinitialized automatically in VideoCapture destructor
    return 0;
}
```

Note

In C API the black-box structure CvCapture is used instead of **VideoCapture**.

- A basic sample on using the **VideoCapture** interface can be found at opencv_source_code/samples/cpp/starter_video.cpp
- Another basic video processing sample can be found at opencv_source_code/samples/cpp/video_dmtx.cpp
- (Python) A basic sample on using the **VideoCapture** interface can be found at opencv_source_code/samples/python/video.py
- (Python) Another basic video processing sample can be found at opencv_source_code/samples/python/video_dmtx.py
- (Python) A multi threaded video processing sample can be found at opencv_source_code/samples/python/video_threaded.py

Examples:

laplace.cpp, and **segment_objects.cpp**.

Constructor & Destructor Documentation

cv::VideoCapture::VideoCapture ( )

Note

In C API, when you finished working with video, release CvCapture structure with **cvReleaseCapture()**, or use **Ptr**<CvCapture> that calls **cvReleaseCapture()** automatically in the destructor.

cv::VideoCapture::VideoCapture (const String & filename)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

filename name of the opened video file (eg. video.avi) or image sequence (eg. img_%02d.jpg, which will read samples like img_00.jpg, img_01.jpg, img_02.jpg, ...)

cv::VideoCapture::VideoCapture (const String & filename,
int apiPreference
)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

filename name of the opened video file (eg. video.avi) or image sequence (eg. img_%02d.jpg, which will read samples like img_00.jpg, img_01.jpg, img_02.jpg, ...)

apiPreference preferred Capture API to use. Can be used to enforce a specific reader implementation if multiple are available: e.g. CAP_FFMPEG or CAP_IMAGES

cv::VideoCapture::VideoCapture (int index)

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

index = camera_id + domain_offset (CAP_*). id of the video capturing device to open. If there is a single camera connected, just pass 0. Advanced Usage: to open Camera 1 using the MS Media Foundation API: index = 1 + CAP_MSMF

virtual cv::VideoCapture::~VideoCapture ( ) virtual

Member Function Documentation

virtual double cv::VideoCapture::get (int propId) const

virtual

Returns the specified **VideoCapture** property.

Parameters

propId Property identifier. It can be one of the following:

- **CAP_PROP_POS_MSEC** Current position of the video file in milliseconds or video capture timestamp.
- **CAP_PROP_POS_FRAMES** 0-based index of the frame to be decoded/captured next.
- **CAP_PROP_POS_AVI_RATIO** Relative position of the video file: 0 - start of the film, 1 - end of the film.
- **CAP_PROP_FRAME_WIDTH** Width of the frames in the video stream.
- **CAP_PROP_FRAME_HEIGHT** Height of the frames in the video stream.
- **CAP_PROP_FPS** Frame rate.
- **CAP_PROP_FOURCC** 4-character code of codec.
- **CAP_PROP_FRAME_COUNT** Number of frames in the video file.
- **CAP_PROP_FORMAT** Format of the **Mat** objects returned by **retrieve()** .
- **CAP_PROP_MODE** Backend-specific value indicating the current capture mode.
- **CAP_PROP_BRIGHTNESS** Brightness of the image (only for cameras).
- **CAP_PROP_CONTRAST** Contrast of the image (only for cameras).
- **CAP_PROP_SATURATION** Saturation of the image (only for cameras).
- **CAP_PROP_HUE** Hue of the image (only for cameras).
- **CAP_PROP_GAIN** Gain of the image (only for cameras).
- **CAP_PROP_EXPOSURE** Exposure (only for cameras).
- **CAP_PROP_CONVERT_RGB** Boolean flags indicating whether images should be converted to RGB.
- **CAP_PROP_WHITE_BALANCE** Currently not supported
- **CAP_PROP_RECTIFICATION** Rectification flag for stereo cameras (note: only supported by DC1394 v 2.x backend currently)

Note

When querying a property that is not supported by the backend used by the **VideoCapture** class, value 0 is returned.

Examples:

laplace.cpp.

virtual bool cv::VideoCapture::grab ( )

virtual

Grabs the next frame from video file or capturing device.

The methods/functions grab the next frame from video file or camera and return true (non-zero) in the case of success.

The primary use of the function is in multi-camera environments, especially when the cameras do not have hardware synchronization. That is, you call **VideoCapture::grab()** for each camera and after that call the slower method **VideoCapture::retrieve()** to decode and get frame from each camera. This way the overhead on demosaicing or motion jpeg decompression etc. is eliminated and the retrieved frames from different cameras will be closer in time.

Also, when a connected camera is multi-head (for example, a stereo camera or a Kinect device), the correct way of retrieving data from it is to call **VideoCapture::grab** first and then call **VideoCapture::retrieve** one or more times with different values of the channel parameter. See https://github.com/Itseez/opencv/tree/master/samples/cpp/openni_capture.cpp

virtual bool cv::VideoCapture::isOpened ( ) const

virtual

Returns true if video capturing has been initialized already.

If the previous call to **VideoCapture** constructor or **VideoCapture::open** succeeded, the method returns true.

Examples:

laplace.cpp, and **segment_objects.cpp**.

virtual bool cv::VideoCapture::open (const String & filename) virtual

Open video file or a capturing device for video capturing.

Parameters

filename name of the opened video file (eg. video.avi) or image sequence (eg. img_%02d.jpg, which will read samples like img_00.jpg, img_01.jpg, img_02.jpg, ...)

The methods first call **VideoCapture::release** to close the already opened file or camera.

Examples:

laplace.cpp, and **segment_objects.cpp**.

virtual bool cv::VideoCapture::open (int index) virtual

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

index = camera_id + domain_offset (CAP_*). id of the video capturing device to open. If there is a single camera connected, just pass 0. Advanced Usage: to open Camera 1 using the MS Media Foundation API: index = 1 + CAP_MSMF

virtual bool cv::VideoCapture::open (const String & filename,
int apiPreference
) virtual

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.

Parameters

filename	name of the opened video file (eg. video.avi) or image sequence (eg. img_%02d.jpg, which will read samples like img_00.jpg, img_01.jpg, img_02.jpg, ...)
apiPreference	preferred Capture API to use. Can be used to enforce a specific reader implementation if multiple are available: e.g. CAP_FFMPEG or CAP_IMAGES

The methods first call **VideoCapture::release** to close the already opened file or camera.

virtual VideoCapture& cv::VideoCapture::operator>> (Mat & image) virtual

virtual VideoCapture& cv::VideoCapture::operator>> (UMat & image) virtual

virtual bool cv::VideoCapture::read (OutputArray image) virtual

Grabs, decodes and returns the next video frame.

The methods/functions combine **VideoCapture::grab** and **VideoCapture::retrieve** in one call. This is the most convenient method for reading video files or capturing data from decode and return the just grabbed frame. If no frames has been grabbed (camera has been disconnected, or there are no more frames in video file), the methods return false and the functions return NULL pointer.

Note

OpenCV 1.x functions cvRetrieveFrame and cv.RetrieveFrame return image stored inside the video capturing structure. It is not allowed to modify or release the image! You can copy the frame using :ocvcvCloneImage and then do whatever you want with the copy.

virtual void cv::VideoCapture::release ( ) virtual

Closes video file or capturing device.

The methods are automatically called by subsequent **VideoCapture::open** and by **VideoCapture** destructor.

The C function also deallocates memory and clears *capture pointer.

virtual bool cv::VideoCapture::retrieve (OutputArray image,
int flag = 0
) virtual

Decodes and returns the grabbed video frame.

The methods/functions decode and return the just grabbed frame. If no frames has been grabbed (camera has been disconnected, or there are no more frames in video file), the methods return false and the functions return NULL pointer.

Note

OpenCV 1.x functions cvRetrieveFrame and cv.RetrieveFrame return image stored inside the video capturing structure. It is not allowed to modify or release the image! You can copy the frame using :ocvcvCloneImage and then do whatever you want with the copy.

virtual bool cv::VideoCapture::set (int propId,
double value
) virtual

Sets a property in the **VideoCapture**.

Parameters

propId Property identifier. It can be one of the following:

- **CAP_PROP_POS_MSEC** Current position of the video file in milliseconds.
- **CAP_PROP_POS_FRAMES** 0-based index of the frame to be decoded/captured next.
- **CAP_PROP_POS_AVI_RATIO** Relative position of the video file: 0 - start of the film, 1 - end of the film.
- **CAP_PROP_FRAME_WIDTH** Width of the frames in the video stream.
- **CAP_PROP_FRAME_HEIGHT** Height of the frames in the video stream.
- **CAP_PROP_FPS** Frame rate.
- **CAP_PROP_FOURCC** 4-character code of codec.
- **CAP_PROP_FRAME_COUNT** Number of frames in the video file.
- **CAP_PROP_FORMAT** Format of the **Mat** objects returned by **retrieve()** .
- **CAP_PROP_MODE** Backend-specific value indicating the current capture mode.
- **CAP_PROP_BRIGHTNESS** Brightness of the image (only for cameras).
- **CAP_PROP_CONTRAST** Contrast of the image (only for cameras).
- **CAP_PROP_SATURATION** Saturation of the image (only for cameras).
- **CAP_PROP_HUE** Hue of the image (only for cameras).
- **CAP_PROP_GAIN** Gain of the image (only for cameras).
- **CAP_PROP_EXPOSURE** Exposure (only for cameras).
- **CAP_PROP_CONVERT_RGB** Boolean flags indicating whether images should be converted to RGB.
- **CAP_PROP_WHITE_BALANCE** Currently unsupported
- **CAP_PROP_RECTIFICATION** Rectification flag for stereo cameras (note: only supported by DC1394 v 2.x backend currently)

value Value of the property.

Examples:

laplace.cpp.

Member Data Documentation

Ptr<CvCapture> cv::VideoCapture::cap protected

Ptr<IVideoCapture> cv::VideoCapture::icap protected

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

- videoio/include/opencv2/**videoio.hpp**