Learn OpenCV by Examples

OpenCV simplified for beginners by the use of examples. Learn OpenCV with basic implementation of different algorithms.

Capture Video from Camera

class **VideoCapture** - Class for video capturing from video files or cameras.

bool VideoCapture::open(const string& filename) // filename – name of the opened video file
bool VideoCapture::open(int device) // device – id of the opened video capturing device (i.e. a camera index)
bool VideoCapture::isOpened() // Returns true if video capturing has been initialized already
void VideoCapture::release() // Closes video file or capturing device
bool VideoCapture::grab() // Grabs the next frame from video file or capturing device
bool VideoCapture::retrieve(Mat& image, int channel=0) // Decodes and returns the grabbed video frame

> 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.

VideoCapture& VideoCapture::(operator)>>(Mat& image) // Grabs, decodes and returns the next video frame
double VideoCapture::get(int propId) // Returns the specified VideoCapture property

SEARCH CONTENTS OF THIS BLOG

Search

POPULAR POSTS

Find Contour

Basic drawing examples

Line Detection by Hough Line Transform

Face Detection using Haar-Cascade Classifier

Perspective Transform

Sobel Edge Detection

Parameters:

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

- **CV_CAP_PROP_POS_MSEC** Current position of the video file in milliseconds or video capture timestamp.
- **CV_CAP_PROP_POS_FRAMES** 0-based index of the frame to be decoded/captured next.
- **CV_CAP_PROP_POS_AVI_RATIO** Relative position of the video file: 0 - start of the film, 1 - end of the film.
- **CV_CAP_PROP_FRAME_WIDTH** Width of the frames in the video stream.
- **CV_CAP_PROP_FRAME_HEIGHT** Height of the frames in the video stream.
- **CV_CAP_PROP_FPS** Frame rate.
- **CV_CAP_PROP_FOURCC** 4-character code of codec.
- **CV_CAP_PROP_FRAME_COUNT** Number of frames in the video file.
- **CV_CAP_PROP_FORMAT** Format of the Mat objects returned by `retrieve()`.
- **CV_CAP_PROP_MODE** Backend-specific value indicating the current capture mode.
- **CV_CAP_PROP_BRIGHTNESS** Brightness of the image (only for cameras).
- **CV_CAP_PROP_CONTRAST** Contrast of the image (only for cameras).
- **CV_CAP_PROP_SATURATION** Saturation of the image (only for cameras).
- **CV_CAP_PROP_HUE** Hue of the image (only for cameras).
- **CV_CAP_PROP_GAIN** Gain of the image (only for cameras).
- **CV_CAP_PROP_EXPOSURE** Exposure (only for cameras).
- **CV_CAP_PROP_CONVERT_RGB** Boolean flags indicating whether images should be converted to RGB.
- **CV_CAP_PROP_WHITE_BALANCE** Currently not supported
- **CV_CAP_PROP_RECTIFICATION** Rectification flag for stereo cameras (note: only supported by DC1394 v 2.x backend currently)

bool VideoCapture::set(int propId, double value) // Sets a property in the VideoCapture

Parameters:

- **propId** –
  - **CV_CAP_PROP_POS_MSEC** Current position of the video file in milliseconds.
  - **CV_CAP_PROP_POS_FRAMES** 0-based index of the frame to be decoded/captured next.
  - **CV_CAP_PROP_POS_AVI_RATIO** Relative position of the video file: 0 - start of the film, 1 - end of the film.
  - **CV_CAP_PROP_FRAME_WIDTH** Width of the frames in the video stream.
  - **CV_CAP_PROP_FRAME_HEIGHT** Height of the frames in the video stream.
  - **CV_CAP_PROP_FPS** Frame rate.
  - **CV_CAP_PROP_FOURCC** 4-character code of codec.

CATEGORIES

- **CV_CAP_PROP_FRAME_COUNT** Number of frames in the video file.
- **CV_CAP_PROP_FORMAT** Format of the Mat objects returned by `retrieve()`.
- **CV_CAP_PROP_MODE** Backend-specific value indicating the current capture mode.
- **CV_CAP_PROP_BRIGHTNESS** Brightness of the image (only for cameras).
- **CV_CAP_PROP_CONTRAST** Contrast of the image (only for cameras).
- **CV_CAP_PROP_SATURATION** Saturation of the image (only for cameras).
- **CV_CAP_PROP_HUE** Hue of the image (only for cameras).
- **CV_CAP_PROP_GAIN** Gain of the image (only for cameras).
- **CV_CAP_PROP_EXPOSURE** Exposure (only for cameras).
- **CV_CAP_PROP_CONVERT_RGB** Boolean flags indicating whether images should be converted to RGB.
- **CV_CAP_PROP_WHITE_BALANCE** Currently unsupported
- **CV_CAP_PROP_RECTIFICATION** Rectification flag for stereo cameras (note: only supported by DC1394 v 2.x backend currently)

- **value** – Value of the property.

Example 1:

```
#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;

    namedWindow("Video",1);
    while(1)
    {
        Mat frame;
        cap >> frame;         // get a new frame from camera
        imshow("Video", frame);

        // Press 'c' to escape
        if(waitKey(30) == 'c') break;
    }
    return 0;
}
```

Example 2:

```
#include "opencv2/objdetect/objdetect.hpp"
#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
using namespace cv;

int main( int argc, const char** argv )
{
    CvCapture* capture;
    Mat frame;

    // Read the video stream
    capture = cvCaptureFromCAM( -1 );
    namedWindow("Video",CV_WINDOW_AUTOSIZE);
    if( capture )
    {
        while( true )
        {
            frame = cvQueryFrame( capture );
            imshow("Video",frame);

            // Press 'c' to escape
            int c = waitKey(10);
            if( (char)c == 'c' ) { break; }
        }
    }
    return 0;
}
```

Labels: Basics

No comments:

Post a Comment