Module 1.5

OpenCV Basics

Read, Write and Display Videos

Satya Mallick, Ph.D.
LearnOpenCV.com

Table of Contents

Read, write and display a video using OpenCV

In this section, we talk about videos in OpenCV.

A video is a sequence of fast moving images. So, the obvious question that follows is how fast are the pictures moving? The measure of how fast the images are transitioning is given by a metric called **frames per second (fps)**. Thus, fps renders a measure of the number of unique consecutive images that are displayed each second.

So, when someone says that the video has an fps value of 40, it means that there are 40 images being displayed every second. Alternatively, after every 25 milliseconds, a new image is loaded.

We often encounter videos while working in the domain of computer vision. In this section, we talk about the basic operations involving videos in OpenCV, namely reading and writing. Thankfully, OpenCV provides a very simple interface to perform these operations.

Reading a Video

In OpenCV, a video can be read either by using the feed from a camera connected to a computer or by reading a video file. The first step towards reading a video file is to create a **VideoCapture** object. Its argument can be either the device index or the name of the video file to be read.

In most cases, only one camera is connected to the system. So, all we do is pass '0' and OpenCV uses the only camera attached to the computer. When more than one camera is connected to the computer, we can select the second camera by passing '1', the third camera by passing '2' and so on.

C++ [Creating VideoCapture Object]

```
// Create a VideoCapture object and open the input file
// If the input is taken from the camera, pass 0 instead of the video file name

VideoCapture cap("../../data/videos/chaplin.mp4");
```

Python [Creating VideoCapture Object]

```
# Create a VideoCapture object and read from input file
# If the input is taken from the camera, pass 0 instead of the video file name.

cap = cv2.VideoCapture('../../data/videos/chaplin.mp4')
```

After the VideoCapture object is created, we can capture the video frame by frame.

Displaying a video

After reading a video file, we can display the video frame by frame. A frame of a video is simply an image and we display each frame the same way we display images, i.e., we use the function **imshow()**.

As in the case of an image, we use the **waitKey()** after imshow() function to pause each frame in the video. In the case of an image, we pass '0' to the waitKey() function, but for playing a video, we need to pass a number greater than '0' to the waitKey() function. This is because '0' would pause the frame in the video for an infinite amount of time and in a video we need each frame to be shown only for some finite interval of time, so we need to pass a number greater than '0' to the waitKey() function. This number is equal to the time in milliseconds we want each frame to be displayed.

While reading the frames from a webcam, using waitKey(1) is appropriate because the display frame rate will be limited by the frame rate of the webcam even if we specify a delay of 1 ms in waitKey.

While reading frames from a video that you are processing, it may still be appropriate to set the time delay to 1 ms so that the thread is freed up to do the processing we want to do.

In rare cases, when the playback needs to be at a certain frame rate, we may want the delay to be higher than 1 ms.

The Python and C++ implementation of reading and displaying a video file follows.

C++ [Reading a video file] [videoRead.cpp]

In the following code, we will use the VideoCapture Object to read a video file and display it.

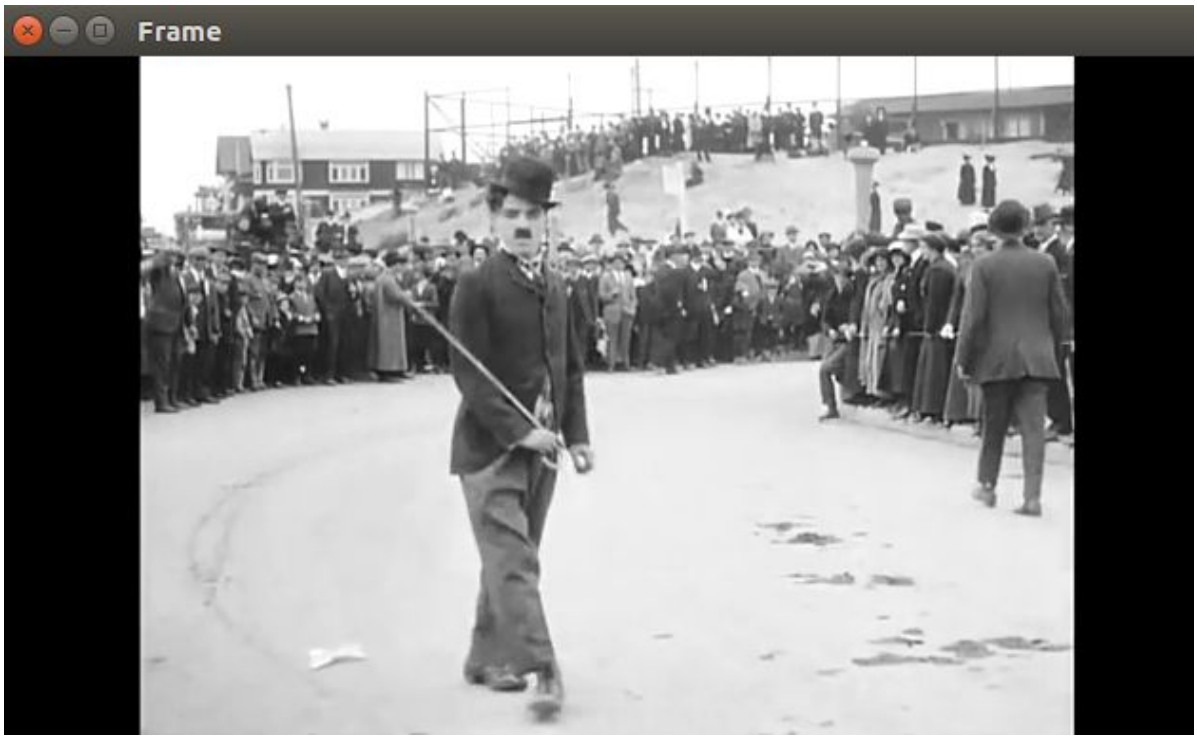
```
1 // Include opencv header files
2 #include "opencv2/opencv.hpp"
3 #include <iostream>
4
5 using namespace std;
6 using namespace cv;
7
8 int main(){
```

In this block,we create a video capture object and read a input video file.

```
// Create a VideoCapture object and open the input file
// If the input is the web camera, pass 0 instead of the video file name
VideoCapture cap("../../data/videos/chaplin.mp4");
// Check if camera opened successfully and read a frame from the object cap
if(!cap.isOpened()){
  cout << "Error opening video stream or file" << endl;
  return -1;
}
```

Read and display video frames until video is completed or user quits by pressing ESC

```
while(1){
  Mat frame;
 // Capture frame-by-frame
 cap >> frame;
 // If the frame is empty, break immediately
 if (frame.empty())
   break;
 // Display the resulting frame
 imshow( "Frame", frame );
```



A snapshot from the video

```
 // Press  ESC on keyboard to exit
 char c=(char)waitKey(25);
 if(c==27)
   break;
}
// When everything done, release the video capture object
```

```
32   cap.release();
33   // Closes all the frames
34   destroyAllWindows();
35   return 0;
36 }
```

Python [Reading a video file] [videoRead.py]

In the following code, we will use the VideoCapture Object to read a video file and display it.

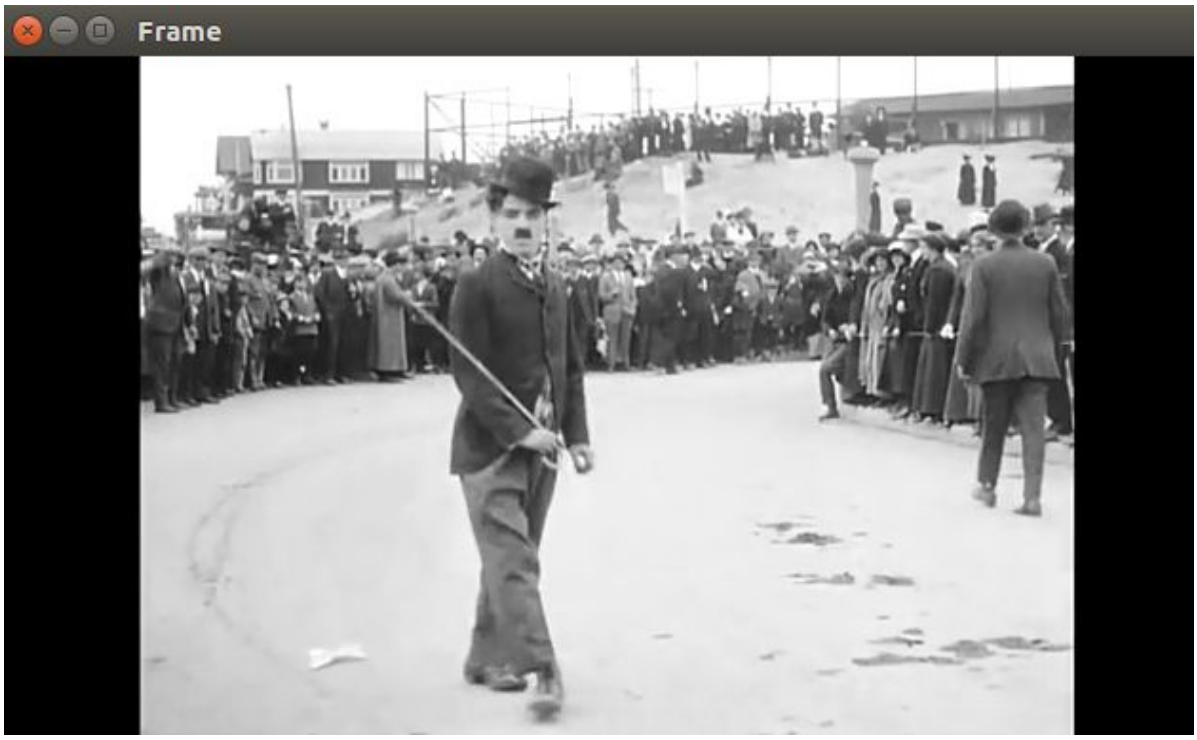
```
1 # Import OpenCV and numpy
2 import cv2
3 import numpy as np
```

In this block,we create a video capture object and read a input video file.

```
4 # Create a VideoCapture object and open the input file
5 # If the input is the web camera, pass 0 instead of the video file name
6 cap = cv2.VideoCapture('../../data/videos/chaplin.mp4')
7 # Check if camera opened successfully
8 if (cap.isOpened()== False):
9   print("Error opening video stream or file")
```

Read and display video frames until video is completed or user quits by pressing ESC

```
10 while(cap.isOpened()):
11   # Capture frame-by-frame
12   ret, frame = cap.read()
13   if ret == True:
14     # Display the resulting frame
15     cv2.imshow('Frame',frame)
```



A snapshot from the video

```
    # Press ESC on keyboard to  exit
    if cv2.waitKey(25) & 0xFF == 27:
      break
  # Break the loop
  else:
    break
# When everything done, release the video capture object
cap.release()
# Closes all the frames
cv2.destroyAllWindows()
```

Writing a video

After we are done with capturing and processing the video frame by frame, the next step we would want to do is to save the video.

For images, it is straightforward. We just need to use cv2.imwrite(). But for videos, we need to toil a bit harder. We need to create a **VideoWriter** object. First, we should specify the output file name with its format (eg: output.avi). Then, we should specify the **FourCC** code and the number of frames per second (FPS). Lastly, the frame size should be passed.

C++ [Creating VideoWriter Object]

```
// Define the codec and create VideoWriter object.The output is stored in 'outcpp.avi' file.
// Define the fps to be equal to 10. Also frame size is passed.

VideoWriter video("outcpp.avi",CV_FOURCC('M','J','P','G'),10, Size(frame_width,frame_height));
```

Python [Creating VideoWriter Object]

```
# Define the codec and create VideoWriter object.The output is stored in 'outpy.avi' file.
# Define the fps to be equal to 10. Also frame size is passed.

out = cv2.VideoWriter('outpy.avi',cv2.VideoWriter_fourcc('M','J','P','G'), 10,
(frame_width,frame_height))
```

FourCC is a 4-byte code used to specify the video codec. The list of available codes can be found at fourcc.org. There are many FOURCC codes available, but in this post we would work only with MJPG.

Note: Only a few of the FourCC codes listed above will work on your system based on the availability of the codecs on your system. Sometimes, even when the specific codec is available, OpenCV may not be able to use it. MJPG is a safe choice.

The Python and C++ implementation of capturing live stream from a camera and writing it to a file follows.

C++ [Writing to a video file] [videoWrite.cpp]

```
// Include opencv header files
#include "opencv2/opencv.hpp"
#include <iostream>

```

```
using namespace std;
using namespace cv;
```

In the following code, we will use the VideoCapture Object to read a video from the webcam, display the feed from webcam and save it by using a VideoWriter Object.

```
int main() {

  // Create a VideoCapture object and use camera to capture the video
  VideoCapture cap(0);
  // Check if camera opened successfully
  if(!cap.isOpened()){

    cout << "Error opening video stream" << endl;
    return -1;
  }
```

In this block of code, we obtain the system dependent frame width and height.

```
  // Default resolutions of the frame are obtained. The default resolutions are system dependent.
  int frame_width = cap.get(CV_CAP_PROP_FRAME_WIDTH);
  int frame_height = cap.get(CV_CAP_PROP_FRAME_HEIGHT);
```

We create the videoWriter object ‘video’, name the output file as ‘outcpp.avi’ and define the 4 FOURCC code. We also specify the ‘fps’ value and the dimensions of the system dependent frame.

```
  // Define the codec and create VideoWriter object.The output is stored in 'outcpp.avi' file.
  VideoWriter video("outcpp.avi",CV_FOURCC('M','J','P','G'),10, Size(frame_width,frame_height));
```

Read and save the feed from webcam until ESC is pressed .

```
  while(1){
    Mat frame;
    // Capture frame-by-frame
    cap >> frame;
    // If the frame is empty, break immediately
    if (frame.empty())
      break;
    // Write the frame into the file 'outcpp.avi'
    video.write(frame);
    // Display the resulting frame
    imshow( "Frame", frame );
```

A snapshot from Webcam Feed

```
33      // Press  ESC on keyboard to  exit
34      char c = (char)waitKey(1);
35      if( c == 27 )
36        break;
37    }
38    // When everything done, release the video capture and write object
39    cap.release();
40    video.release();
41   // Closes all the frames
42   destroyAllWindows();
43   return 0;
44  }
```

Python [Writing to a video file] [videoWrite.py]

```
1 # Import OpenCV and numpy
2 import cv2
3 import numpy as np
```

In the following code, we will use the VideoCapture Object to read a video from the webcam, display the feed from webcam and save it by using a VideoWriter Object.

```
4 # Create a VideoCapture object
5 cap = cv2.VideoCapture(0)
6 # Check if camera opened successfully
```

```
if (cap.isOpened() == False):
  print("Unable to read camera feed")
```

In this block of code, we obtain the system dependent frame width and height.

```
# Default resolutions of the frame are obtained. The default resolutions are system dependent.
# We convert the resolutions from float to integer.
frame_width = int(cap.get(3))
frame_height = int(cap.get(4))
```

We create the videoWriter object ‘video’, name the output file as ‘outpy.avi’ and define the 4 FOURCC code. We also specify the ‘fps’ value and the dimensions of the system dependent frame.

```
# Define the codec and create VideoWriter object.The output is stored in 'outpy.avi' file.
out = cv2.VideoWriter('outpy.avi',cv2.VideoWriter_fourcc('M','J','P','G'),
                                          10,  (frame_width,frame_height))
```

Read and save the feed from webcam until ESC is pressed .

```
while(True):
 ret, frame = cap.read()

 if ret == True:
   # Write the frame into the file 'output.avi'
   out.write(frame)
   # Display the resulting frame
   cv2.imshow('frame',frame)
```

A snapshot from Webcam feed

```
    # Press ESC on keyboard to stop recording
    if cv2.waitKey(25) & 0xFF == 27:
      break
  # Break the loop
  else:
    break
# When everything done, release the video capture and video write objects
cap.release()
out.release()
# Closes all the frames
cv2.destroyAllWindows()
```

References and Further reading

http://docs.opencv.org/3.0-beta/modules/videoio/doc/reading_and_writing_video.html

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/py_video_display.html

https://www.learnopencv.com/how-to-find-frame-rate-or-frames-per-second-fps-in-opencv-python-cpp/