# Getting Started with Videos

## Goal

- Learn to read video, display video, and save video.
- Learn to capture video from a camera and display it.
- You will learn these functions : **cv.VideoCapture()**, **cv.VideoWriter()**

## Capture Video from Camera

Often, we have to capture live stream with a camera. OpenCV provides a very simple interface to do this. Let's capture a video from the camera (I am using the built-in webcam on my laptop), convert it into grayscale video and display it. Just a simple task to get started.

To capture a video, you need to create a **VideoCapture** object. Its argument can be either the device index or the name of a video file. A device index is just the number to specify which camera. Normally one camera will be connected (as in my case). So I simply pass 0 (or -1). You can select the second camera by passing 1 and so on. After that, you can capture frame-by-frame. But at the end, don't forget to release the capture.

```
import numpy as np
import cv2 as cv

cap = cv.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open camera")
    exit()
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    # Our operations on the frame come here
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
    # Display the resulting frame
    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv.destroyAllWindows()
```

`cap.read()` returns a bool ( `True` / `False` ). If the frame is read correctly, it will be `True` . So you can check for the end of the video by checking this returned value.

Sometimes, cap may not have initialized the capture. In that case, this code shows an error. You can check whether it is initialized or not by the method **cap.isOpened()**. If it is `True` , OK. Otherwise open it using **cap.open()**.

You can also access some of the features of this video using **cap.get(propId)** method where propId is a number from 0 to 18. Each number denotes a property of the video (if it is applicable to that video). Full details can be seen here: **cv::VideoCapture::get()**. Some of these values can be modified using **cap.set(propId, value)**. Value is the new value you want.

For example, I can check the frame width and height by `cap.get(cv.CAP_PROP_FRAME_WIDTH)` and `cap.get(cv.CAP_PROP_FRAME_HEIGHT)` . It gives me 640x480 by default. But I want to modify it to 320x240. Just use `ret = cap.set(cv.CAP_PROP_FRAME_WIDTH,320)` and `ret = cap.set(cv.CAP_PROP_FRAME_HEIGHT,240)` .

**Note**

If you are getting an error, make sure your camera is working fine using any other camera application (like Cheese in Linux).

## Playing Video from file

Playing video from file is the same as capturing it from camera, just change the camera index to a video file name. Also while displaying the frame, use appropriate time for `cv.waitKey()` . If it is too less, video will be very fast and if it is too high, video will be slow (Well, that is how you can display videos in slow motion). 25 milliseconds will be OK in normal cases.

```
import numpy as np
import cv2 as cv

cap = cv.VideoCapture('vtest.avi')
```

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

    # if frame is read correctly ret is True
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    gray = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)

    cv.imshow('frame', gray)
    if cv.waitKey(1) == ord('q'):
        break

cap.release()
cv.destroyAllWindows()
```

**Note**

Make sure a proper version of ffmpeg or gstreamer is installed. Sometimes it is a headache to work with video capture, mostly due to wrong installation of ffmpeg/gstreamer.

## Saving a Video

So we capture a video and process it frame-by-frame, and we want to save that video. For images, it is very simple: just use `cv.imwrite()`. Here, a little more work is required.

This time we create a **VideoWriter** object. We should specify the output file name (eg: output.avi). Then we should specify the **FourCC** code (details in next paragraph). Then number of frames per second (fps) and frame size should be passed. And the last one is the **isColor** flag. If it is `True`, the encoder expect color frame, otherwise it works with grayscale frame.

FourCC is a 4-byte code used to specify the video codec. The list of available codes can be found in fourcc.org. It is platform dependent. The following codecs work fine for me.

- In Fedora: DIVX, XVID, MJPG, X264, WMV1, WMV2. (XVID is more preferable. MJPG results in high size video. X264 gives very small size video)
- In Windows: DIVX (More to be tested and added)
- In OSX: MJPG (.mp4), DIVX (.avi), X264 (.mkv).

FourCC code is passed as `cv.VideoWriter_fourcc('M','J','P','G')` or `cv.VideoWriter_fourcc(*'MJPG')` for MJPG.

The below code captures from a camera, flips every frame in the vertical direction, and saves the video.

```
import numpy as np
import cv2 as cv

cap = cv.VideoCapture(0)

# Define the codec and create VideoWriter object
fourcc = cv.VideoWriter_fourcc(*'XVID')
out = cv.VideoWriter('output.avi', fourcc, 20.0, (640,  480))

while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame (stream end?). Exiting ...")
        break
    frame = cv.flip(frame, 0)

    # write the flipped frame
    out.write(frame)

    cv.imshow('frame', frame)
    if cv.waitKey(1) == ord('q'):
        break

# Release everything if job is finished
cap.release()
out.release()
cv.destroyAllWindows()
```

## Additional Resources

## Exercises