

```
1 # 18BCS6201-CV Practical-1 (Gauri Prabhakar) (AI-ML-2
  )(B)
2 # Aim: Find out the frame rate of a video and print
  it in video using python and OpenCV.
3
4 # Importing necessary modules.
5 import cv2
6 import time
7
8 # Creating a variable to store the video using the '.
  VideoCapture()' function.
9 video_cap = cv2.VideoCapture(r'C:\Users\gauri\Desktop
  \OpenCV Media\Sand - 73847.mp4')
10
11 # Initializing time that is setting up the start time
  .
12 time_1 = time.time()
13 time_2 = time.time()
14
15 # Setting up the infinite while loop.
16 while 1:
17
18     # Capturing the video frame by frame using the '.
  read()' method.
19     res, frame = video_cap.read()
20     time_3 = time.time()
21
22     # Calculating frames per second and storing it in
  the variable 'FPS'.
23     # FPS = 1/ (Total Time - Starting Time) that is 1
  / (Total time to process the loop)
24     frames_per_sec = 1 / (time_3 - time_1)
25     time_1 = time_3
26
27     # Using '.putText' to write a string over the
  video, which will be the frames per second.
28     # We first pass the frame then the string (frames
  /sec) then the co-ordinates of the text string
29     # then the font style, scale, color and finally
  the thickness of the font.
30     cv2.putText(frame, f'FPS: {int(frames_per_sec)}'
```

```
30 , (20, 70), cv2.FONT_HERSHEY_COMPLEX_SMALL, 4, (0, 0
    , 0), 5)
31
32     # Setting up the loop.
33     if res:
34
35         # Rendering the video with effective FPS to
        the console by using the function '.imshow()'.
36         cv2.imshow("Output Video with FPS", frame)
37
38         # Setting up '.waitkey()' to wait for a
        specific time until any key is pressed and break the
        loop.
39         # '.waitkey(1)' displays a frame for 1ms
        after which it moves to the next frame in the video.
40         # Setting 'x' as the quitting button.
41
42         # Note: '.waitkey(1)' returns a 32-bit
        integer and -1 when no input is made.
43         # Since ASCII values can go up to a maximum
        of 255,
44         # we perform a bitwise AND operation to
        achieve only the last 8 bits.
45         # '0xFF' is nothing but a hexadecimal
        constant having the value of 11111111 (8-bit binary).
46         # Then we compare the bit value obtained
        after the AND operation with the ASCII value of the
        quit key which is
47         # 'x' in this case. If the both are same, we
        break out of the loop.
48         if cv2.waitKey(1) & 0xFF == ord('x'):
49             break
50     else:
51         break
52
53 # Releasing the variable/object 'video_cap'.
54 video_cap.release()
55
56 # Destroying all windows.
57 cv2.destroyAllWindows()
58
```