18BCS6201-CV Practical-1 (Gauri Prabhakar) (AI-ML-2)(B)

Aim: Find out the frame rate of a video and print it in video using python and OpenCV.

In [2]:

```python
# Importing necessary modules.
import cv2
import time

# Creating a variable to store the video using the '.VideoCapture()' function.
video_cap = cv2.VideoCapture(r'C:\Users\gauri\Desktop\OpenCV Media\Sand - 73847.mp4')

# Initializing time that is setting up the start time.
time_1 = time.time()
time_2 = time.time()

# Setting up the infinite while loop.
while 1:

    # Capturing the video frame by frame using the '.read()' method.
    res, frame = video_cap.read()
    time_3 = time.time()

    # Calculating frames per second and storing it in the variable 'FPS'.
    # FPS = 1/ (Total Time - Starting Time) that is 1/ (Total time to process the loop)
    frames_per_sec = 1 / (time_3 - time_1)
    time_1 = time_3

    # Using '.putText' to write a string over the video, which will be the frames per second.
    # We first pass the frame then the string (frames/sec) then the co-ordinates of the text string
    # then the font style, scale, color and finally the thickness of the font.
    cv2.putText(frame, f'FPS: {int(frames_per_sec)}', (20, 70), cv2.FONT_HERSHEY_COMPLEX_SMALL, 4, (0, 0, 0), 5)

    # Setting up the loop.
    if res:

        # Rendering the video with effective FPS to the console by using the function '.imshow()'.
        cv2.imshow("Output Video with FPS", frame)

        # Setting up '.waitkey()' to wait for a specific time until any key is pressed and break the loop.
        # '.waitkey(1)' displays a frame for 1ms after which it moves to the next frame in the video.
        # Setting 'x' as the quitting button.

        # Note: '.waitkey(1)' returns a 32-bit integer and -1 when no input is made.
        # Since ASCII values can go up to a maximum of 255,
        # we perform a bitwise AND operation to achieve only the last 8 bits.
        # '0xFF' is nothing but a hexadecimal constant having the value of 11111111 (8-bit binary).
        # Then we compare the bit value obtained after the AND operation with the ASCII value of the quit key which is
        # 'x' in this case. If the both are same, we break out of the loop.
        if cv2.waitKey(1) & 0xFF == ord('x'):
            break
    else:
        break

# Releasing the variable/object 'video_cap'.
video_cap.release()

# Destroying all windows.
cv2.destroyAllWindows()
```

In [ ]:

```

```