Video Frame Extraction Prep

I have a folder containing several videos (serving as the video data) and another folder containing csv files (serving as the metadata for each corresponding video).
For a computer vision regression model. I want to pre-process the sample of videos such that, 2 frames per second are extracted and stored locally as images in a newly created video corresponding to each video, with the name of the folder being derived from the index/ID of the video.

Below is an example of how you might approach this pre-processing task using Python. The general idea is to loop through all your videos, open each video with a library like OpenCV, extract frames at a rate of 2 frames per second, and then save the frames into a new folder whose name corresponds to the video's identifier (which you can derive from its filename). You can later use the CSV metadata (which—for example—could be matched by filename) as needed for your regression model. Here's a step-by-step explanation along with a sample code snippet.

---

1. Overview

- **Video Folder & CSV Folder:**
  You have two folders, one with video files (e.g., MP4, AVI) and another containing CSV files with metadata for each video. The assumption is that the video filenames share an "index" or ID with the CSV files (e.g., `video01.mp4` matches `video01.csv`).
- **Task:**
  For each video, extract 2 frames per second and save them as images inside a newly created folder. The folder name should be derived from the video's index or ID (for example, using the base filename without its extension).
- **Why 2 FPS?**
  You want a standardized sampling rate to reduce the volume of data while retaining enough temporal information for a regression task.

---

2. Steps to Pre-Process the Videos

1. **Import Required Libraries:**
   - `os` and/or `pathlib` for file system operations.
   - `cv2` from the OpenCV library to handle video operations.
   - Optionally, `pandas` if you intend to load and possibly merge the CSV metadata later.
2. **Loop Through All Video Files:**
   - Identify video files (by extension) in your video folder.
   - For each video, derive its index/ID from its file name (for example, removing the file extension).
3. **Create an Output Directory for Each Video:**
   - Create a new folder (if it doesn't exist already) to store the extracted frames.
   - The folder name should be the video's index or ID.
4. **Extract Frames at 2 FPS:**
   There are two common methods:
   - **Method A – Based on Frame Count:**
     Retrieve the original video FPS (frames per second) and compute a frame interval. For example, if a video has 30 FPS, then every 15th frame (30 / 2) is roughly one frame per 0.5 seconds.
   - **Method B – Based on Timestamp:**
     You can set the video position (in milliseconds) and extract frames at fixed 0.5-second intervals.

   In the sample below, we use Method A assuming the video FPS is known and stable.
5. **Save the Frames:**
   - Save each extracted frame as an image file (e.g., JPG) inside the corresponding folder.
   - Use an incremental filename (e.g., `0001.jpg`, `0002.jpg`, ...) to keep the sequence clear.
6. **Integrate Metadata (Optional):**
   - If needed, you can load the matching CSV file for each video using pandas.
   - This metadata can later be used to label the images or as regression targets.

3. Python Code Example

Below is an annotated example script that implements the above steps:

python

```python
import os
import cv2
import pandas as pd  # if you plan to work with CSV metadata

def extract_frames(video_path, output_dir, fps_extraction=2):
    """
    Extract frames from a video at a specified rate (frames per second)
    and save them to the output directory.

    Args:
        video_path (str): Path to the input video file.
        output_dir (str): Path to the directory where frames will be stored.
        fps_extraction (int): Number of frames per second to extract.
    """
    # Ensure the output directory exists
    os.makedirs(output_dir, exist_ok=True)

    # Open the video file
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print(f"Error opening video file: {video_path}")
        return

    # Retrieve the video's frames per second (FPS)
    video_fps = cap.get(cv2.CAP_PROP_FPS)
    if video_fps == 0:
        print(f"Cannot retrieve FPS for: {video_path}")
        return

    # Calculate the interval between frames to extract
    frame_interval = int(round(video_fps / fps_extraction))

    frame_count = 0
    saved_count = 0
```

```
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        # Save the frame if it's time according to the frame interval
        if frame_count % frame_interval == 0:
            # Create a filename with zero-padding
            frame_filename = os.path.join(output_dir, f"{saved_count:04d}.jpg")
            cv2.imwrite(frame_filename, frame)
            saved_count += 1

        frame_count += 1

    cap.release()
    print(f"Extracted {saved_count} frames from {os.path.basename(video_path)}")

# Define paths for your folders
video_folder = '/path/to/your/video_folder'      # Folder with videos
metadata_folder = '/path/to/your/csv_metadata'     # Folder with CSV metadata (if
needed)
output_base_folder = '/path/to/your/output_images'  # Folder to store the extracted
frames

# Loop through each video file in the video folder
for video_filename in os.listdir(video_folder):
    # Check if the file is a video by its extension (add more extensions if needed)
    if video_filename.lower().endswith(('.mp4', '.avi', '.mov')):
        # Derive the video ID (for example, using the filename without extension)
        video_id = os.path.splitext(video_filename)[0]
        video_path = os.path.join(video_folder, video_filename)

        # Create an output directory for this particular video
        video_output_folder = os.path.join(output_base_folder, video_id)

        # Optionally, load the corresponding CSV metadata if required
        metadata_filename = f"{video_id}.csv"
        metadata_path = os.path.join(metadata_folder, metadata_filename)
        if os.path.exists(metadata_path):
            try:
                metadata_df = pd.read_csv(metadata_path)
                # You can process metadata_df as needed for your regression task
```

```
        except Exception as e:
            print(f"Error reading {metadata_path}: {e}")

    # Extract frames at 2 FPS and save them into the designated folder
    extract_frames(video_path, video_output_folder, fps_extraction=2)
```

---

4. Additional Considerations

- **Environment Setup:**
  Make sure you have the required libraries. You can install OpenCV and pandas (if needed) using pip:

  ```
  bash

  pip install opencv-python pandas
  ```

- **Frame Extraction Accuracy:**
  - When using the frame count method, slight inaccuracies might occur if the video FPS isn't a precise integer.
  - For more accurate temporal sampling (particularly for videos with variable or nonstandard frame rates), consider seeking frames using the video timestamp with `cap.set(cv2.CAP_PROP_POS_MSEC, time_in_ms)`.
- **Error Handling:**
  The above code includes basic error handling (e.g., checking if the video opens correctly, handling non-existent CSV files). For production, you might add more robust logging or exception handling.
- **Integration with Metadata:**
  Once your frames are extracted, you can use the CSV data (for example, regression labels) to further process or label the images for your computer vision model.

---

5. Summary

This solution outlines how to pre-process videos by:

- Extracting frames at a uniform rate (2 FPS).
- Organizing the extracted images into folders named after each video's index or ID.
- Optionally integrating CSV metadata for further processing in a regression task.

By using OpenCV to handle the video I/O and Python's file operations, you can create a flexible and efficient pre-processing pipeline tailored to your computer vision regression application.

Feel free to modify the code to fit your exact folder structure or to add additional data handling logic as needed.