Name: Seshasai.PB Reg\_N0: 21MIA1005

Image and Video Analytics Assignment 2

**Lab Task 1: Setup and Basic Extraction**

**Objective:**

Install the necessary tools and libraries, and extract frame information from a video.

**Steps:**

1. **Install ffmpeg and ffmpeg-python**:
   * Install the ffmpeg tool and the ffmpeg-python library.
2. **Extract Frame Information**:
   * Extract frame information from a sample video.

Code:

import sys

import ffmpeg

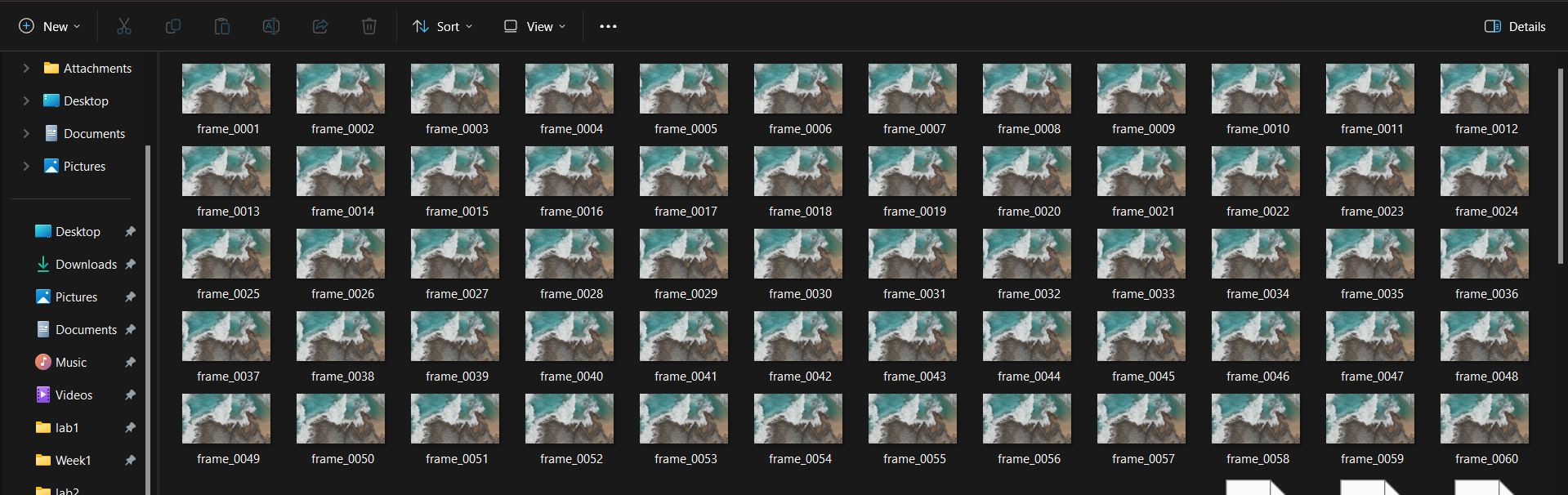
sys.path.append(r'C:\ffmpeg')

input\_file = 'in.mp4'

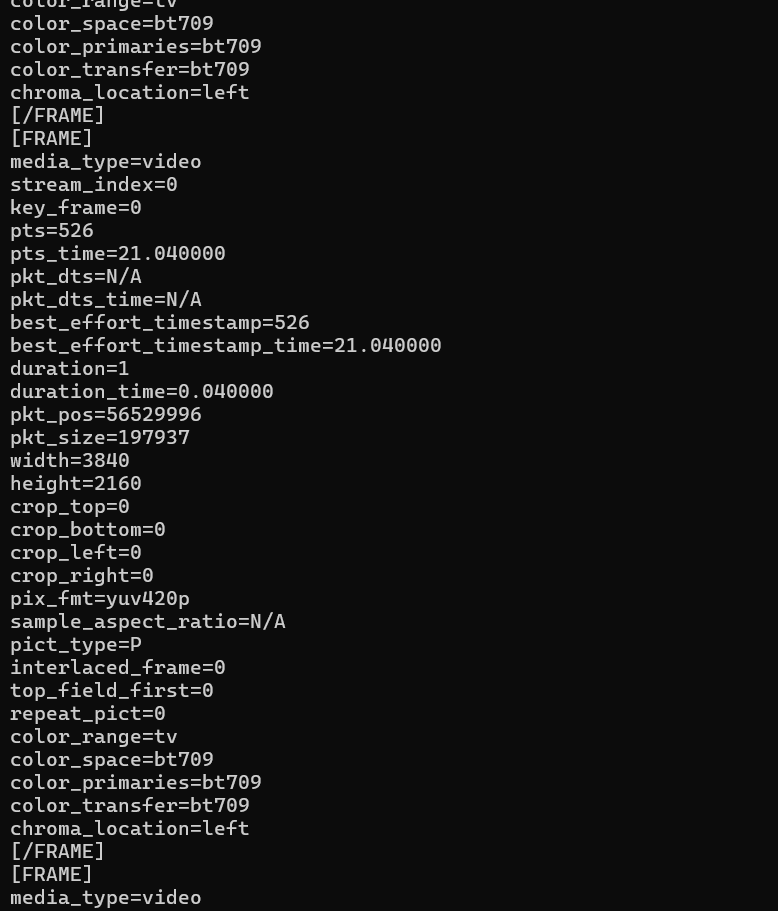
output\_pattern = 'frames/frame\_%04d.jpeg'

ffmpeg.input(input\_file).output(output\_pattern).run()

****my spyder console is not recognizing **‘ffmpeg’** I’m running it in **‘command prompt’**

So after running this we get our frames

To get information about each frame we can run this prompt in cmd

This will display information about each frame

**Lab Task 2: Frame Type Analysis**

**Objective:**

Analyze the extracted frame information to understand the distribution of I, P, and B frames in a video.

**Steps:**

1. **Modify the Script**:
   * Count the number of I, P, and B frames.
   * Calculate the percentage of each frame type in the video.
2. **Analyze Frame Distribution**:
   * Plot the distribution of frame types using a library like matplotlib.
   * Plot a pie chart or bar graph showing the distribution of frame types using matplotlib.

To get type of frame information, Enter

This will display type of each frame

We can extract this and information to analyse and visualize the frames to get information

**Code:**

frame\_types =["I"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"I"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"I"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"P"

,"B"

,"B"

,"B"

,"P"

,"I"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"I"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"I"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"I"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"B"

,"P"

,"B"

,"B"

,"P"

,"P"]

# Count the number of each frame type

frame\_counts = {'I': 0, 'P': 0, 'B': 0}

for frame\_type in frame\_types:

    if frame\_type in frame\_counts:

        frame\_counts[frame\_type] += 1

# Calculate total frames

total\_frames = len(frame\_types)

# Calculate percentage of each frame type

frame\_percentages = {ftype: (count / total\_frames) \* 100 for ftype, count in frame\_counts.items()}

# Print the results

print(f"Frame Counts: {frame\_counts}")

print(f"Frame Percentages: {frame\_percentages}")

import matplotlib.pyplot as plt

def plot\_distribution(frame\_counts, frame\_percentages):

    # Plotting the Pie Chart

    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)

    plt.pie(frame\_counts.values(), labels=frame\_counts.keys(), autopct='%1.1f%%')

    plt.title('Frame Type Distribution (Pie Chart)')

    # Plotting the Bar Graph

    plt.subplot(1, 2, 2)

    plt.bar(frame\_counts.keys(), frame\_counts.values(), color=['red', 'green', 'blue'])

    plt.xlabel('Frame Type')

    plt.ylabel('Count')

    plt.title('Frame Type Distribution (Bar Graph)')

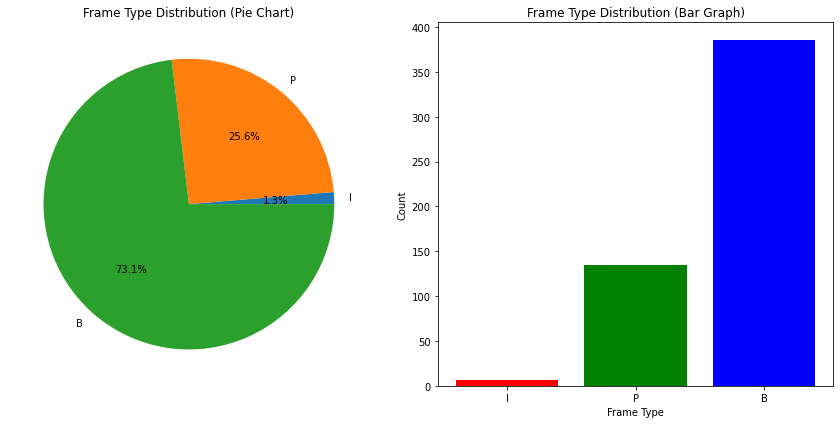
    plt.tight\_layout()

    plt.show()

plot\_distribution(frame\_counts, frame\_percentages)

**Output:**

Frame Counts: {'I': 7, 'P': 135, 'B': 386}

Frame Percentages: {'I': 1.3257575757575757, 'P': 25.568181818181817, 'B': 73.10606060606061}

**Lab Task 3: Visualizing Frames**

**Objective:**

Extract actual frames from the video and display them using Python.

**Steps:**

1. **Extract Frames**:
   * Use ffmpeg to extract individual I, P, and B frames from the video.
   * Save these frames as image files.
2. **Display Frames**:
   * Use a library like PIL (Pillow) or opencv-python to display the extracted frames.

**Tasks:**

1. Save I, P, and B frames as separate image files using ffmpeg.
2. Use PIL or opencv-python to load and display these frames in a Python script.
3. Compare the visual quality of I, P, and B frames.

**Code:**

import os

# Path to frames directory

frames\_dir = './frames/'  # Adjust this path as needed

# Rename frames

for i, frame\_type in enumerate(frame\_types):

    original\_filename = f"frame\_{i+1:04d}.jpeg"

    new\_filename = f"{frame\_type}\_frame\_{i+1:04d}.jpeg"

    original\_path = os.path.join(frames\_dir, original\_filename)

    new\_path = os.path.join(frames\_dir, new\_filename)

    if os.path.exists(original\_path):

        os.rename(original\_path, new\_path)

    else:

        print(f"File {original\_filename} does not exist in the directory.")

import cv2

import matplotlib.pyplot as plt

# Define a function to display images using OpenCV

def display\_frames(frame\_paths):

    for path in frame\_paths:

        # Load the image

        image = cv2.imread(path)

        # Convert BGR to RGB for displaying using matplotlib

        image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

        plt.imshow(image\_rgb)

        plt.title(f"Frame: {os.path.basename(path)}")

        plt.axis('off')

        plt.show()

# Example paths (Replace with actual paths after renaming)

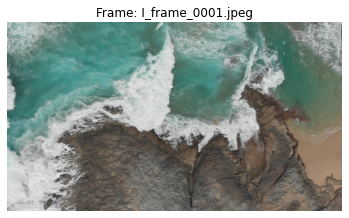
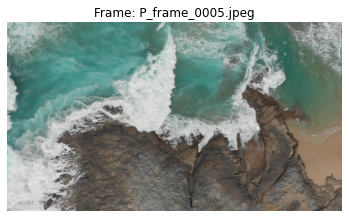
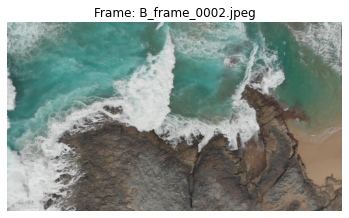
example\_I\_frame = os.path.join(frames\_dir, 'I\_frame\_0001.jpeg')

example\_P\_frame = os.path.join(frames\_dir, 'P\_frame\_0005.jpeg')

example\_B\_frame = os.path.join(frames\_dir, 'B\_frame\_0002.jpeg')

# Displaying example frames

display\_frames([example\_I\_frame, example\_P\_frame, example\_B\_frame])

**Quality comparison of different image types:**

**Lab Task 4: Frame Compression Analysis**

**Objective:**

Analyze the compression efficiency of I, P, and B frames.

**Steps:**

1. **Calculate Frame Sizes**:
   * Calculate the file sizes of extracted I, P, and B frames.
   * Compare the average file sizes of each frame type.
2. **Compression Efficiency**:
   * Discuss the role of each frame type in video compression.
   * Analyze why P and B frames are generally smaller than I frames.

**Code:**

import os

# Directory containing the renamed frames

frames\_dir = './frames/'

# Initialize dictionaries to store sizes and counts

frame\_sizes = {'I': [], 'P': [], 'B': []}

# Calculate the file sizes

for filename in os.listdir(frames\_dir):

    if filename.startswith('I\_') or filename.startswith('P\_') or filename.startswith('B\_'):

        frame\_type = filename.split('\_')[0]

        file\_path = os.path.join(frames\_dir, filename)

        file\_size = os.path.getsize(file\_path)

        frame\_sizes[frame\_type].append(file\_size)

# Calculate average sizes

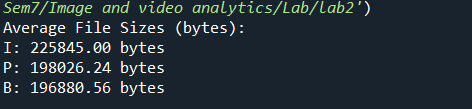
average\_sizes = {frame\_type: sum(sizes) / len(sizes) if sizes else 0 for frame\_type, sizes in frame\_sizes.items()}

# Print out the results

print("Average File Sizes (bytes):")

for frame\_type, avg\_size in average\_sizes.items():

    print(f"{frame\_type}: {avg\_size:.2f} bytes")

**Output:**

**Analysis**

These results align more closely with typical expectations, where I-frames are larger than P-frames and B-frames:

1. **I-Frames** are expected to be larger because they store a complete image without reference to other frames.
2. **P-Frames** are smaller than I-frames as they only store differences from previous frames, using predictive coding.
3. **B-Frames** are usually the smallest, leveraging both past and future frames to encode differences with high efficiency.

**Lab Task 5: Advanced Frame Extraction**

**Objective:**

Extract frames from a video and reconstruct a part of the video using only I frames.

**Steps:**

1. **Extract and Save I Frames**:
   * Extract I frames from the video and save them as separate image files.
2. **Reconstruct Video**:
   * Use the extracted I frames to reconstruct a portion of the video.
   * Create a new video using these I frames with a reduced frame rate.

**Code:**

import cv2

import os

# Path to the directory containing I frames

i\_frame\_dir = './I\_frames/'  # Update this path as needed

output\_video\_path = 'reconstructed.mp4'  # Output path for the reconstructed video

# Define frame rate (we'll use 3.5 fps for at least 2 seconds duration)

frame\_rate = 3.5

num\_frames = 7

# Get the list of frame file names

frame\_files = [f for f in sorted(os.listdir(i\_frame\_dir)) if f.startswith('I\_frame\_')]

# Check if the number of frames matches the expected count

if len(frame\_files) != num\_frames:

    print(f"Error: Expected {num\_frames} frames, found {len(frame\_files)}.")

else:

    # Load the first frame to get the frame size

    first\_frame = cv2.imread(os.path.join(i\_frame\_dir, frame\_files[0]))

    height, width, layers = first\_frame.shape

    # Define the codec and create VideoWriter object

    fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  # Codec for .mp4 files

    video\_writer = cv2.VideoWriter(output\_video\_path, fourcc, frame\_rate, (width, height))

    # Write each frame to the video

    for frame\_file in frame\_files:

        frame\_path = os.path.join(i\_frame\_dir, frame\_file)

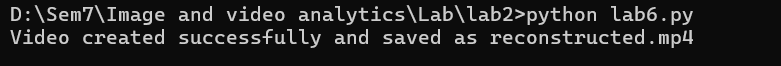
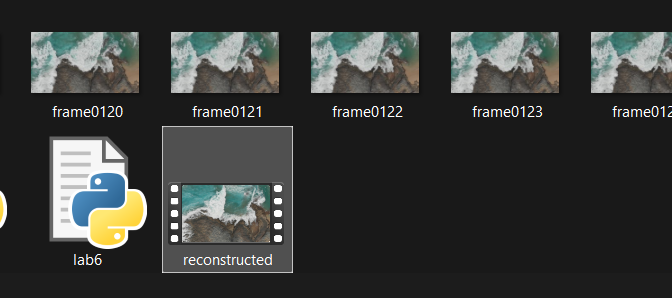
        frame = cv2.imread(frame\_path)

        video\_writer.write(frame)

    # Release the video writer

    video\_writer.release()

    print(f"Video created successfully and saved as {output\_video\_path}")

**Output:**