Modular Implementation Plan for Video Compression Project

Intro to Digital Signal ProcessingMay 2025

Overview

This document outlines a comprehensive and modular plan for implementing the Digital Signal Processing project titled *"Simplified Video Compression with DCT and Predictive Coding"*. The project involves building a basic video encoder/decoder in MATLAB using techniques like DCT, quantization, RLE, and predictive coding. The implementation is divided into phases for clarity and efficiency.

Phase 0 – Setup and Familiarization

- **Duration:** 1 Day
- **Objective:** Understand the theoretical foundation of DCT-based compression.
- **Tasks:**
  - Study project instructions and concepts including DCT, quantization, RLE, and GOP structures.
  - Review the helper functions: `frame_to_mb.m` and `mb_to_frame.m`.
  - Create the project folder structure for organizing scripts and data.

Phase 1 – Core Components (Modules)

- **Duration:** 2–3 Days
- **Objective:** Develop independent modules that will be used in both encoding and decoding pipelines.
- **Modules to Implement:**
  1. **DCT and Quantization:** `dct_quantize_block.m`
     - Applies 2D DCT using `dct2`.
     - Quantizes each coefficient using a standard quantization matrix.

2. **Zigzag + RLE Encoding:** `rle_encode_block.m`
   - Performs zigzag scanning of an 8x8 block.
   - Applies Run-Length Encoding for compression.
3. **Inverse RLE and Zigzag:** `rle_decode_block.m`
   - Reconstructs the 2D quantized block from RLE + zigzag data.
4. **Inverse Quantization + IDCT:** `dequantize_idct_block.m`
   - Reconstructs the macroblock using inverse quantization and `idct2`.
5. **Serialization:** `serialize_data.m`, `deserialize_data.m`
   - Save and load binary streams using `fwrite`, `fread`.

Phase 2 – Implement `compress.m`

- **Duration:** 2 Days
- **Steps:**
  1. Read input frames and convert them to `double`.
  2. Divide each frame into 8x8x3 macroblocks.
  3. Loop over frames:
     - For I-frames: process with DCT → Quant → Zigzag → RLE.
     - For P-frames: compute residual with previous frame and compress.
  4. Serialize and save to `result.bin`.

Phase 3 – Implement `decompress.m`

- **Duration:** 2 Days
- **Steps:**
  1. Load `result.bin`.
  2. For each frame:
     - I-frame: decompress with inverse RLE → zigzag → dequant → IDCT.
     - P-frame: decompress residual and add to previous frame.
  3. Convert macroblocks back to image and save as .jpg.

Phase 4 – Metric Plots and Analysis

- **Duration:** 1 Day
- **Deliverables:**
    1. **Compression Ratio Plot:** GOP size vs compressed ratio.
    2. **PSNR Plot:** For GOP sizes 1, 15, 30 compare PSNR frame-by-frame.

Phase 5 – Improved Algorithm (Part 2)

- **Duration:** 2–3 Days
- **Choose one improvement path:**
    - **Option A:** Implement block matching for motion estimation.
    - **Option B:** Add B-frames and experiment with quantization matrices.
- **Re-implement encoder/decoder as:**
    - `improved_compress.m`, `improved_decompress.m`
- **Generate updated compression ratio and PSNR plots.**

Phase 6 – Finalization and Reporting

- **Duration:** 1 Day
- **Tasks:**
    - Write project report with methodology, plots, and parameter explanations.
    - Prepare `README.md` for script usage.
    - Verify all scripts run without errors.

Conclusion

By following this modular and time-managed plan, the project can be completed efficiently with clear milestones. Each phase is structured to support incremental testing and verification, ensuring robustness in the final implementation.