15-418 Project proposal

Andrew Kim (akim2)
Martin Lee (hyungwol)

Title

Parallel Video Reconstruction via Motion-Aware Predictive Upsampling

URL

Project website: `https://github.com/punchthatface/15418-FinalProject`

Summary

We will implement a parallel video reconstruction pipeline that reduces effective memory or transmission bandwidth by subsampling each video frame, then reconstructing missing pixels through spatial and temporal prediction. We will progressively implement these three functionalities: naive spatial parallelism, motion-aware dynamic tile scheduling, and a heterogeneous CPU–GPU pipeline. We will evaluate throughput, scaling, reconstruction quality (PSNR/SSIM), and bandwidth savings.

Background

Many modern visual workloads—including mobile GPUs, cloud gaming systems, video conferencing, and camera ISPs—are increasingly *bandwidth-bound* rather than compute-bound. A common architectural strategy is to store or transmit only a subset of pixels and then reconstruct missing data using inexpensive spatial and temporal prediction. Our project will investigate this compute–bandwidth tradeoff by building a simplified but realistic parallel video reconstruction pipeline.

We will draw on several classical image and video processing techniques:

- **Sum of Absolute Differences (SAD)** for per-tile motion detection.
- **Jacobi/Laplacian stencil smoothing** for local propagation of known samples.
- **Edge-aware spatial filters** to avoid blurring across boundaries.
- **Temporal IIR filtering**, extending the 1-D exponential moving average:

$$y_t = \alpha x_t + (1 - \alpha) y_{t-1},$$

  into a per-pixel 2-D temporal refinement:

$$R_t^{(1)}(x, y) = \alpha R_t^{(0)}(x, y) + (1 - \alpha) R_{t-1}(x, y).$$

Given a video consisting of frames F_t, our reconstruction pipeline will operate in four algorithmic stages:

1. **Frame Subsampling.** We will keep only sparse known pixels per frame (e.g., 1 of 4) and treat all others as missing.
2. **Spatial Prediction.** We will generate an initial reconstruction using interpolation or edge-aware smoothing.
3. **Temporal Refinement.** We will blend this spatial estimate with the previous frame to reduce flicker and exploit coherence.
4. **Iterative Local Refinement.** We will apply several stencil iterations to propagate known information into missing regions.

To introduce realistic irregular parallelism, we will incorporate **motion-aware tile classification**. Each frame will be partitioned into tiles (e.g., 16×16), and we will compute SAD against R_{t-1}:

- *Static tiles* will reuse the previous frame's values.
- *Active tiles* will undergo full reconstruction.

This motion-driven variability in work will motivate the need for more advanced parallelization strategies beyond naive pixel parallelism.

The Challenge

A naive tile- or pixel-parallel implementation will provide straightforward speedup, but would not meaningfully challenge our parallel systems design. The true difficulty of this project lies in parallelizing a *multi-stage, data-dependent, and temporally coupled* pipeline rather than a single pass or filter.

We will structure our work as a *progressive build-up* of increasingly sophisticated parallelization approaches, each exposing new systems challenges:

1. Naive Spatial Parallelism (Baseline)

We will first parallelize reconstruction by distributing tiles across CPU threads (OpenMP) or GPU thread blocks (CUDA). This will provide a reference point for performance, but it will not address:

- unnecessary computation on static tiles,
- synchronization across multi-pass stages,
- temporal dependence across frames,
- memory locality constraints during stencil operations.

This baseline will highlight the limitations of simple parallelism and motivate more advanced strategies.

2. Motion-Aware Dynamic Scheduling

We will introduce **parallel motion detection** and **active-tile worklist construction** using prefix-sum/compaction on the GPU or CPU. Only tiles classified as "active" will be reconstructed.

This stage will create *non-trivial parallelism* because:

- the number of active tiles will vary significantly between frames,
- active tiles may be spatially clustered or scattered,
- compaction overhead must be weighed against saved computation,
- load balancing becomes dynamic rather than uniform,
- memory locality changes when stencils operate on sparse tile sets.

Parallelizing this irregular, input-dependent workload will be substantially more complex than naive tiling.

3. Heterogeneous CPU–GPU Pipeline (Stretch Goal)

If time permits, we will explore splitting work across CPU and GPU:

- The CPU will perform frame I/O, motion detection, and static-tile reuse.
- The GPU will reconstruct only active tiles using spatial–temporal filtering and iterative stencils.
- We will attempt to overlap CPU processing of frame $t{+}1$ with GPU reconstruction of frame t to create cross-frame pipeline parallelism.

This design will introduce challenges including:

- managing host–device transfers,
- hiding latency behind multi-stage GPU computation,
- coordinating temporal dependencies across frames,
- adapting the pipeline to different motion profiles.

This heterogeneous setup will mirror real-world reconstruction pipelines and will allow us to study compute/bandwidth tradeoffs in a realistic setting.

Resources

We will use C++ with OpenMP for CPU parallelism and CUDA for GPU kernels. Video I/O will be implemented via OpenCV or pre-extracted frame sequences. Experiments will run on GHC machines and optionally PSC Bridges-2 for higher-resolution tests.

Goals and Deliverables

Plan to Achieve

- A complete sequential pipeline implementing subsampling, spatial prediction, temporal refinement, and quality evaluation.
- Parallel CPU and GPU versions of the naive spatial approach.
- A motion-aware dynamic scheduling implementation with tile compaction.
- Performance evaluation: throughput, scaling, and tile activation statistics.
- Quality evaluation: PSNR (Peak Signal to Noise Ratio), SSIM (Structural Similarity Index Measure), and artifact analysis.
- Bandwidth analysis: reconstruction error vs subsampling ratio.

Hope to Achieve

- A heterogeneous CPU–GPU pipeline with frame-level overlap.
- Optimized GPU implementations using shared memory for stencil passes.
- Real-time or near real-time reconstruction on selected videos.

Demo Plan

We will show the reconstructed videos compared with the original video, along with tile activation visualizations, and speedup graphs comparing parallelization approaches. Our choice of video is not yet finalized, but we're planning on to use an open-sourced movie

Platform Choice

GPUs are well suited for tile-parallel spatial processing, while CPUs offer flexible control over dynamic scheduling and motion analysis. Using both systems allows us to explore heterogeneous pipelines and understand the tradeoffs between compute-bound and bandwidth-bound stages.

Schedule

- **Week 1 (Baseline Algorithm):**
  - Implement sequential end-to-end reconstruction pipeline
  - Subsampling + spatial prediction
  - Temporal blending
  - PSNR/SSIM computation for baseline metrics
- **Week 2 (Parallelization)** (50% goal)**:**

  - Implement OpenMP parallel reconstruction on CPU
  - Implement CUDA naive full-frame reconstruction kernel
  - Evaluate CPU-only and GPU-only parallel performance (no heterogeneity yet)
- **Week 3 (Motion-Aware Scheduling)** (75–100% goals)**:**
  - Implement motion/change detection for each tile
  - Active-tile worklist construction (prefix-sum / compaction)
  - GPU reconstruction for active tiles only
  - Measure load balance and motion-driven variability
- **Week 4 (Heterogeneous System + Evaluation)** (125–150% goals)**:**
  - Integrate CPU motion detection + GPU active-tile reconstruction into a heterogeneous pipeline
  - Explore CPU/GPU overlap across frames (if time permits)
  - Full benchmarking and scalability analysis
  - Quality evaluation (PSNR/SSIM vs. subsampling/motion)
  - Final report and website

Division of work

We will divide responsibilities so that each member leads different parts of the system while both contributors share design and evaluation.

Andrew Kim (akim2)

- Lead implementation of the sequential reconstruction pipeline (subsampling, spatial prediction, temporal refinement, and iterative smoothing).
- Implement and tune the OpenMP-based CPU parallelization.
- Implement CPU-side motion/change detection and active-tile classification.
- Develop PSNR/SSIM and other evaluation metrics, and build scripts for benchmarking and plotting.
- Co-design and help integrate the heterogeneous CPU–GPU pipeline and participate in final debugging.

Martin Lee (hyungwol)

- Lead CUDA implementation of spatial prediction, temporal refinement, and stencil kernels.
- Implement GPU-side active-tile worklist construction (prefix-sum / compaction) and active-tile reconstruction.

- Optimize GPU performance (shared memory usage, memory coalescing, tile/block configuration).
- Implement and tune mechanisms for overlapping CPU motion detection with GPU reconstruction in the heterogeneous pipeline.
- Assist with performance experiments on GHC/PSC and integrate results into the final report.

Collaboration Plan

- We will jointly design data layouts (frame storage, tile grids, tile metadata) and pipeline interfaces so that CPU and GPU components integrate cleanly.
- We will pair-program the initial sequential pipeline to ensure both members understand the full reconstruction algorithm.
- Both team members will participate in debugging, parameter tuning (e.g., motion thresholds, subsampling ratios), and analyzing performance/quality tradeoffs.
- We will co-author the final report and project website, with Andrew focusing slightly more on methodology/analysis and Martin on implementation details and optimization choices.