15-418 Project Milestone Report

Parallel Video Reconstruction via Motion-Aware Predictive Upsampling

Andrew Kim (akim2)
Martin Lee (hyungwol)

https://martinlhw.github.io/PVR-via-MAPU/

1. Project Overview and Current Progress

Our project aims to accelerate a video reconstruction pipeline that restores missing pixels from aggressively subsampled video. Our system reconstructs full-resolution frames from a sparse pixel grid by combining spatial interpolation, iterative smoothing, and motion-aware selective refinement. Ultimately, our goal is to utilize both CPU parallelism (OpenMP) and GPU execution (CUDA) to push reconstruction performance toward real-time or near real-time throughput while preserving perceptual quality (PSNR/SSIM).

At this point in the semester, we estimate that we have completed approximately **45% of the project**. The serial CPU pipeline is fully implemented and verified, and we have integrated OpenMP into all major computational loops. We have also begun the CUDA portion of our work: GPU detection, host-side integration, and the refinement entry points are complete, and we are now constructing the first major CUDA kernel (stencil-based iterative refinement). Our tile-based motion classifier is also complete, allowing us to dynamically identify active regions for selective refinement.

We originally planned to incorporate a temporal refinement path. However, after experimentally evaluating its impact, we found that temporal blending across frames offered no significant benefit in PSNR or SSIM and sometimes degraded quality due to motion smearing. Following the same iterative reasoning documented in the example milestone reports, we removed this step from our system. This change allows us to concentrate our remaining time on optimizing the spatial refinement path on the GPU, which we now understand is the component with the highest performance leverage.

Overall, we have maintained steady progress and remain on track to achieve our core deliverables. The remaining major work includes CUDA kernel correctness, shared-memory optimizations, and full performance evaluation across multiple configurations.

2. Summary of Completed Work

Serial Reconstruction Pipeline

We implemented a full end-to-end reconstruction pipeline capable of handling entire video sequences. The system includes:

- **Subsampling and mask generation:** Downsamples input frames by a given factor.
- **Spatial interpolation:** Uses bilinear interpolation to fill initial pixel estimates.
- **Iterative smoothing (Jacobi refinement):** Repeated 3×3 stencil averaging significantly improves frame quality.
- **Temporal reuse (removed):** Initially implemented, later removed following experiments.

All components have been validated with MSE, PSNR, and SSIM metrics.

OpenMP Parallelism

We parallelized all major CPU-bound loops:

- subsampling and mask creation,
- spatial interpolation,
- iterative refinement (inner stencil loop).

Even prior to CUDA acceleration, these optimizations substantially reduced execution time and allowed us to explore the impact of reconstruction iteration count on quality.

CUDA Integration (In Progress)

We completed:

- Device detection and fallback execution,
- CUDA API hooks integrated with the main pipeline,
- Host-side memory management structure,
- Refinement kernel interfaces and data flow.

The first working CUDA stencil kernel is under development; the next milestone will be correctness validation and shared-memory tiling.

Motion-Aware Tile Classification

We implemented a SAD-based tile classifier that identifies "active" tiles by differences between subsampled frames. This will allow us to minimize unnecessary reconstruction work on static regions of video.

3. Updated Goals and Deliverables

Progress Against Original Deliverables

Deliverable	Status
Serial baseline pipeline	Completed
CPU OpenMP parallelism	Completed
Motion-aware scheduling	Partially completed (CPU path done)
CUDA refinement	In progress
Performance instrumentation	Completed
Quality evaluation (MSE/PSNR/SSIM)	Completed

We still expect to deliver all core functionality listed in the proposal. CUDA refinement and GPU shared-memory optimization remain the largest tasks but are well underway.

Updated "Nice-to-Haves"

- Overlapping CPU tile classification with GPU refinement.
- Near real-time playback for 480p or smaller inputs.
- Parameter auto-tuning and heuristic selection.

4. Planned Final Output

- **Side-by-side visual comparison:**
  - original video,
  - subsampled input,
  - reconstructed output (serial, OpenMP, CUDA).
- **Graphical performance results:**
  - CPU OpenMP speedup vs. thread count,
  - GPU throughput vs. subsampling factor,
  - per-stage timing breakdown (subsample, classify, reconstruct, refine).
- **Tile activation heatmaps** illustrating motion-aware refinement.
- **Quality plots** showing PSNR and SSIM over time.

5. Preliminary Results

Spatial Interpolation

Bilinear interpolation significantly improves visual smoothness. For a factor-16 subsampled frame:

- MSE improved from $\sim$10,000 to $\sim$700.

Temporal Refinement (Removed)

Temporal reuse did not yield measurable quality improvements:

- no improvement in MSE or SSIM,
- sometimes worsened quality due to motion smearing.

Iterative Refinement

Our iterative 3×3 smoothing provided the largest quality gains:

- MSE improved from approximately 27 to 10.89.

Quantitative Comparisons

Method	MSE	SSIM	Time (s)
No spatial recon	10.9255	0.977856	44.1016
Spatial interpolation only	∼700 (factor-16)	–	–
Iterative refinement	10.89	–	34.36
OpenMP (early pipeline)	49	–	14.79

After tweaking the knobs (tile size, SAD threshold, iteration count), we realized that having having very low MSE (≈ 10) results in a very high latency, so we decided to stick with slightly higher MSE with significantly lower latency for now.

6. Updated Schedule (Half-Week Increments)

Week 3 (Mon–Wed)

- Implement full-frame CUDA stencil refinement kernel (Martin)
- Validate CUDA correctness with controlled test images (Andrew)
- Integrate debugging/timing instrumentation for GPU execution (Martin)

Week 3 (Thu–Sun)

- Implement shared-memory tiling for CUDA refinement (Martin)
- Build CPU–GPU comparison test harness (Andrew)
- Integrate active-tile path for GPU refinement (Both)

Week 4 (Mon–Wed)

- Conduct full performance evaluation on GHC/PSC machines (Andrew)
- Execute parameter sweeps (iteration count, tile size, subsample factor) (Andrew)
- Tune motion-aware thresholds for GPU path (Both)

Week 4 (Thu–Sun)

- Integrate CPU motion detection with GPU refinement into a unified pipeline (Both)
- Remove unused temporal-refinement components (Martin)
- Prepare final quality/performance graphs and complete final report (Andrew)

7. Issues and Remaining Unknowns

- **CUDA stencil correctness:** Boundary conditions, memory alignment, and numerical consistency require careful debugging.
- **Shared-memory optimization:** We must design a memory layout that balances bank-conflict avoidance, coalescing, and per-block tile size.
- **GPU worklist construction:** Prefix-sum + compaction for the active-tile list is complex and may require multiple kernel launches.
- **CPU–GPU pipeline coordination:** Overlapping tile classification with GPU refinement will likely require restructuring the main loop.
- **Metric consistency:** MSE/SSIM vary significantly depending on which pipeline components are active; we must ensure standardized evaluation.