Embedded Real-time HD Video Deblurring

Timothy J. Dysart, Jay B. Brockman: Emu Solutions

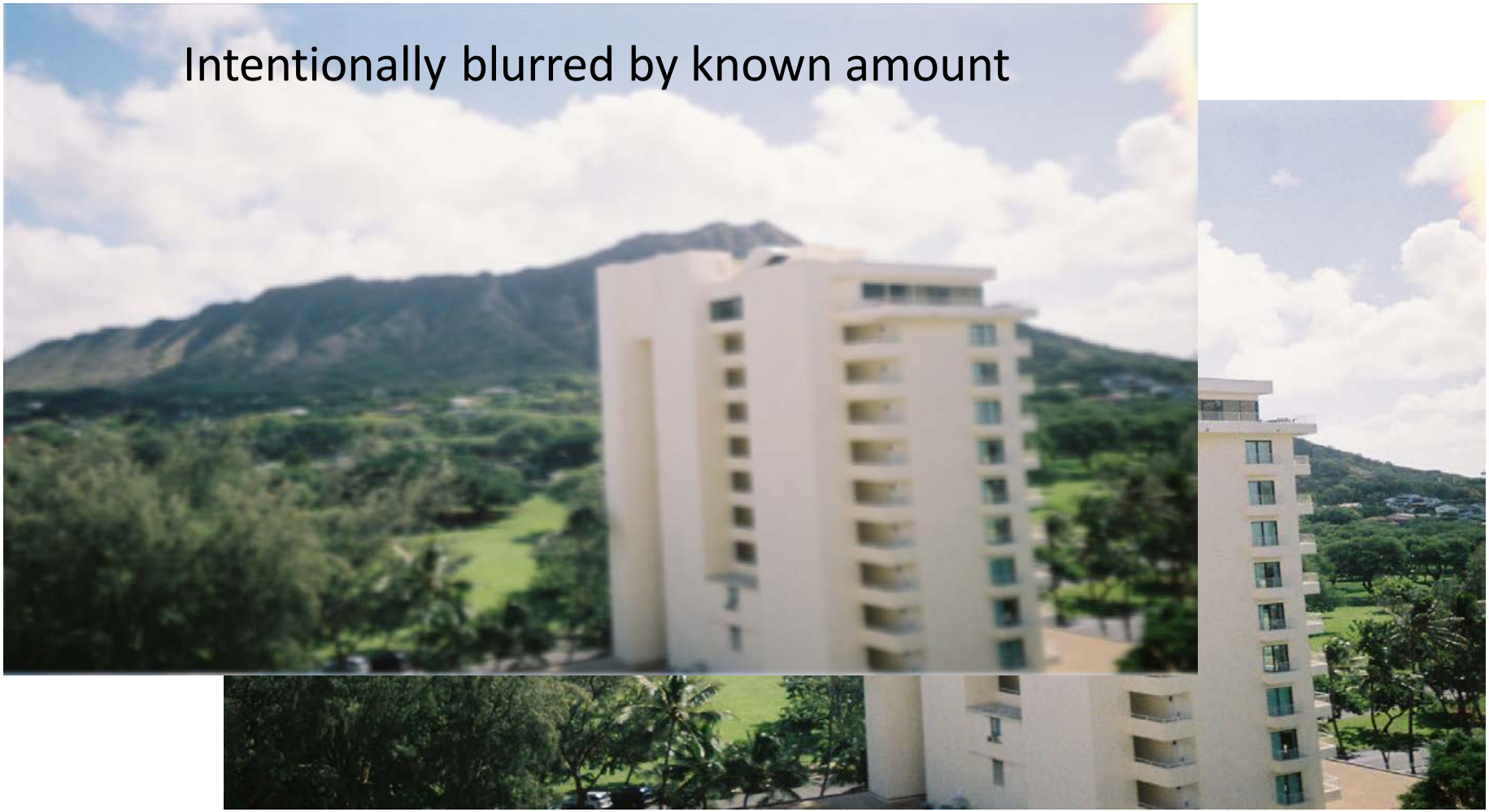
Stephen Jones, Fred Bacon: Aerodyne Research

Sponsored by the U.S. Army Research Laboratory

Overview

- Problem Overview
- Deblurring Algorithm
- Wavelet Transform
- Performance
- Future Work/Summary

Problem Overview



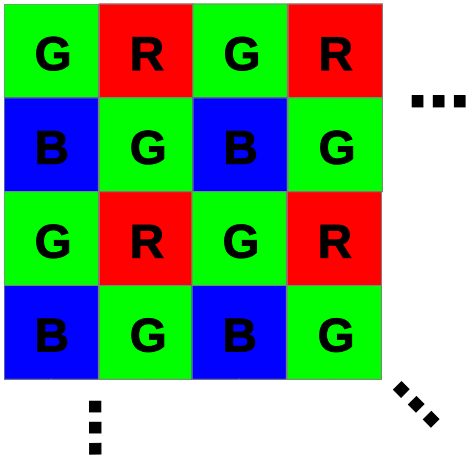
HD Video Feed at 30 frames per second (fps)

Problem Overview

Problem Overview: 1 step back

Camera output: Bayer formatted image

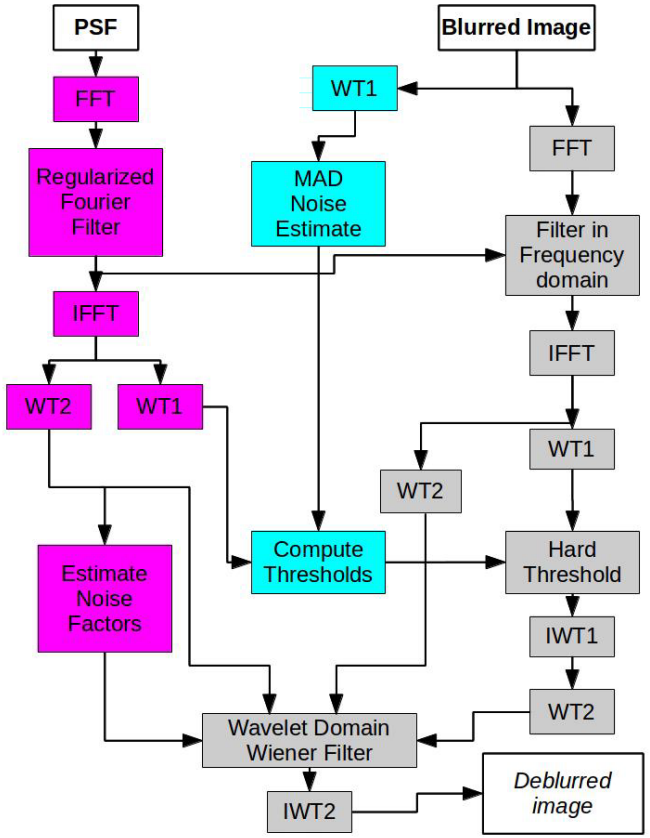
Bayer format



Demosaicking: Fill in the missing pixel data

Full Problem: Demosaic and Deblur

- $y(x) = g(x) * f(x) + n(x)$
  - $y(x)$: blurred image
  - $g(x)$: point-spread function (function of camera and lens)
  - $f(x)$: original image
    - Need to recover!
  - $n(x)$: additive noise
- Deblur via the ForWaRD algorithm
  - Frequency domain operations do main image recovery
  - Wavelet processing to remove noise
    - Use stationary wavelet transform – no subsampling



Neelamani *et al*, "ForWaRD: Fourier-Wavelet Regularized Deconvolution for ill-conditioned systems," *IEEE Trans. on Signal Processing*, vol 52, no 2, Feb. 2004

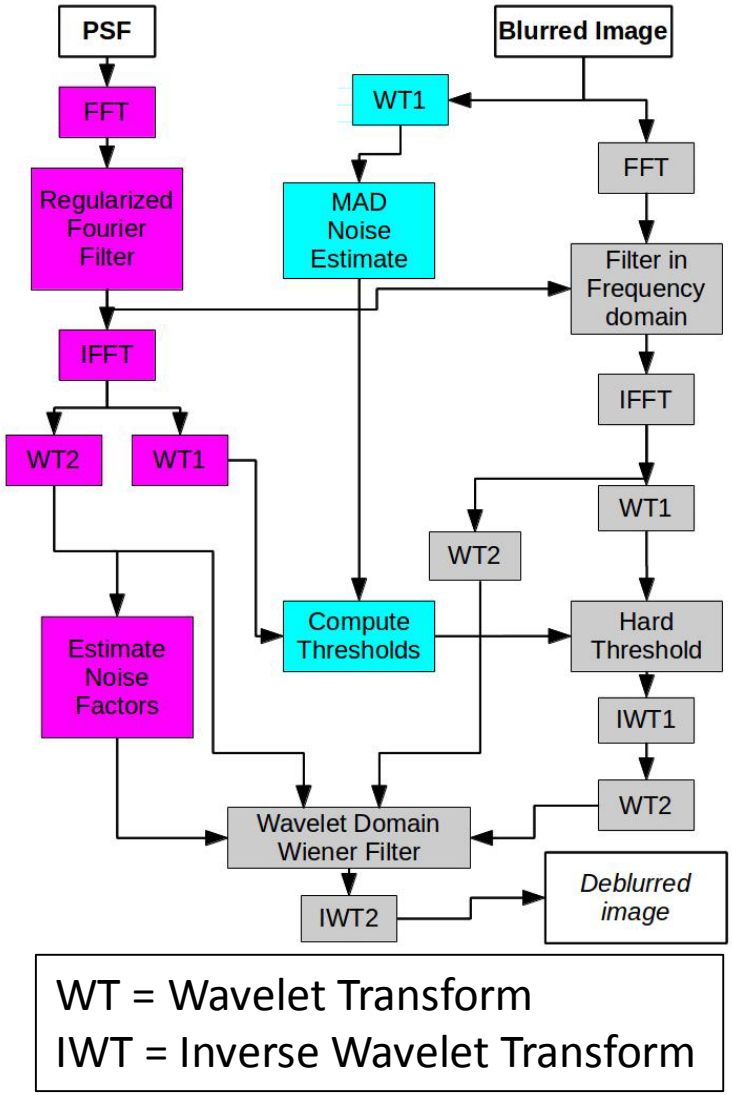
Implementation

- Create DDGPU Library
  - Demosaic and Deblur GPU
- Developed for several Nvidia GPU generations using CUDA
- Use GPGPU capabilities rather than graphics capabilities
- Bandwidth, bandwidth, bandwidth

Data type	Bytes per pixel	Bandwidth to copy 30 frames/sec
uint8	3	356 MB/s
SP Floating Point	12	1.39 GB/s
Complex SP Floating Point	24	2.78 GB/s

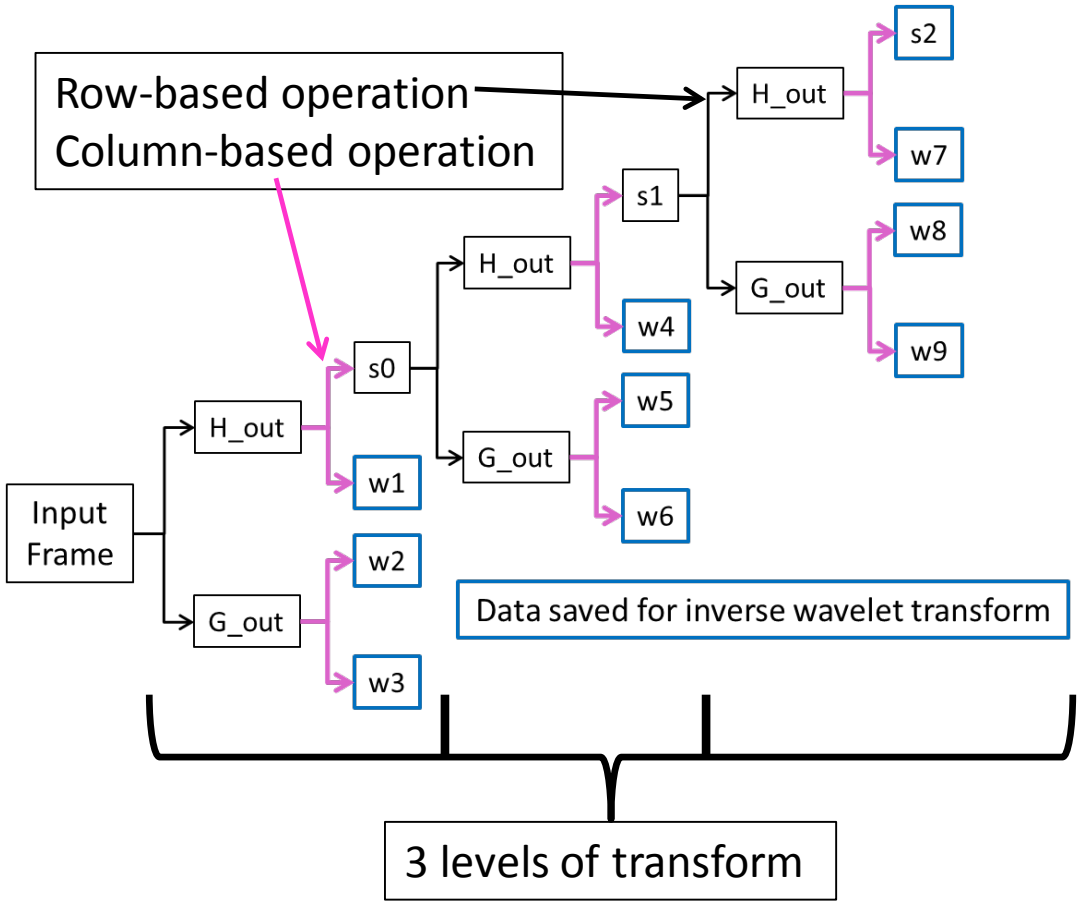
Deblurring Walk Through

- **Pre-Compute**
  - Do at startup, save on device
- **Threshold Update**
  - Prefer per frame, but can be slower
- **Deblurring**
  - Does the work
- All transforms are 2D
- Each color operated on individually



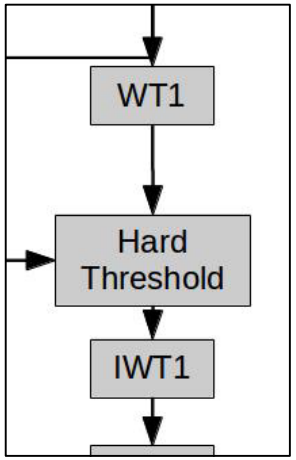
Stationary Wavelet Transforms

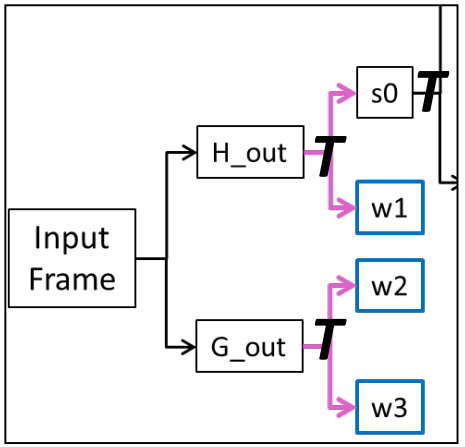
- Forward transform:
  - 10 output frames per input frame
  - Bandwidth:
    - 9 frame reads
    - 18 frame writes
- Inverse transform:
  - Bandwidth:
    - 18 frame reads
    - 9 frame writes
- Single precision FP



WT-Threshold-Inv WT Bandwidth

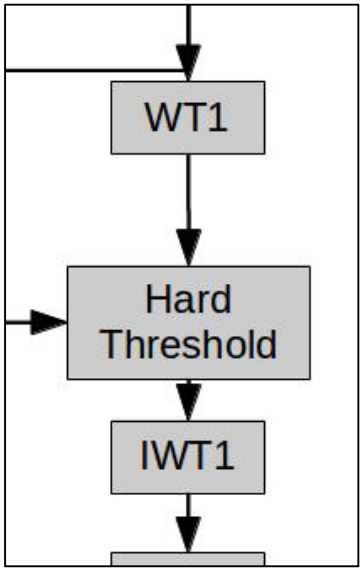
- Forward wavelet transform bandwidth (computation): **19 GB/s**
  - Transposes (8) to ensure all accesses are row-based for memory coalescing: **22 GB/s**
- Hard threshold bandwidth: **14 GB/s**
- Inverse wavelet transform bandwidth (computation): **19 GB/s**
  - Transposes (8): **22 GB/s**
- **Total: 96 GB/sec!**
  - 44 GB/s just for data alignment
  - Transpose count already heavily reduced





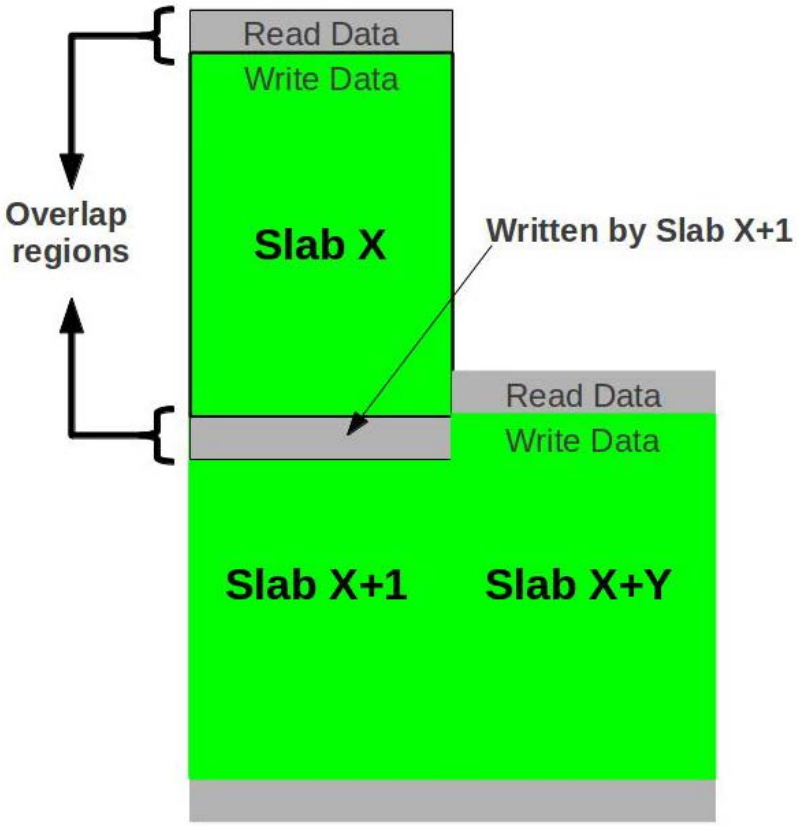
Reducing Bandwidth

- Transpose count reduction:
  - Do not transpose frames generated by WT used in Inv WT (labeled w*X*)
  - Saves 20 transposes (~28 GB/s)
    - Not counted on previous slide
- Embed "Hard Threshold" in Inv WT processing step
  - Saves 14 GB/sec
  - Down to 82 GB/sec
- Use "slabs" to remove remaining transposes



Reduce Bandwidth via Slabs

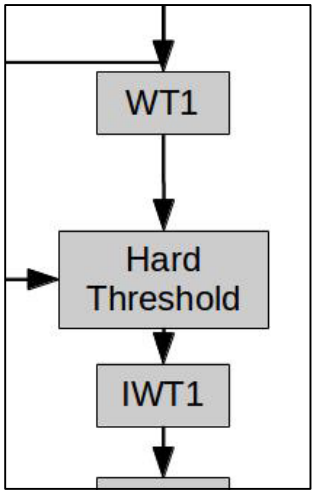
- Column-based operations via “Slabs”
- Slab characteristics
  - Independent
  - 2^N columns wide: Used N=4
    - Preserve memory coalescing
  - Read M+N rows: Used 120
  - Write N rows: Used 80
- Overhead of 50% of frame for extra reads

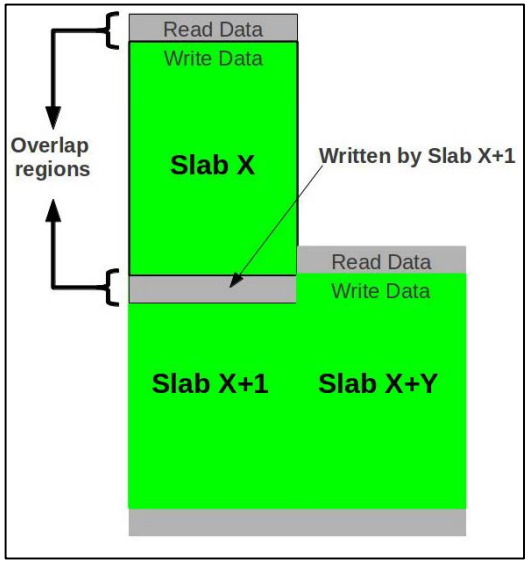


Adopted from: van der Laan *et al*, “Accelerating wavelet lifting on graphics hardware using CUDA,” *IEEE Trans. on Parallel and Distributed Systems*, vol 22, no 1, Jan. 2011

Reduce Bandwidth via Slabs

- For full processing step:
  - 13.5 frame reads replaces 16 transposes
    - No useful work during transpose
    - Useful work done via slabs
  - 19 GB/s instead of 44 GB/s
- With all reductions
  - 57 GB/s instead of 96 GB/s

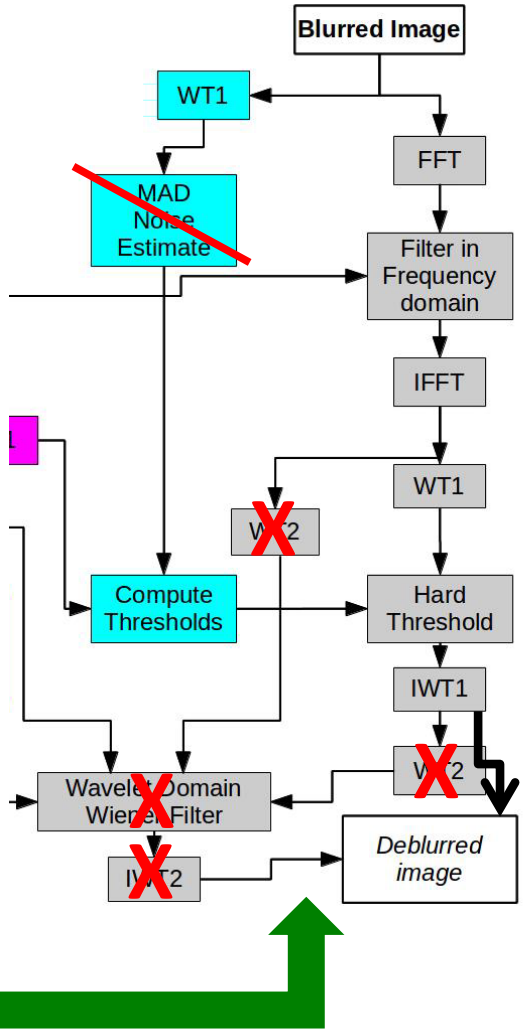




Performance

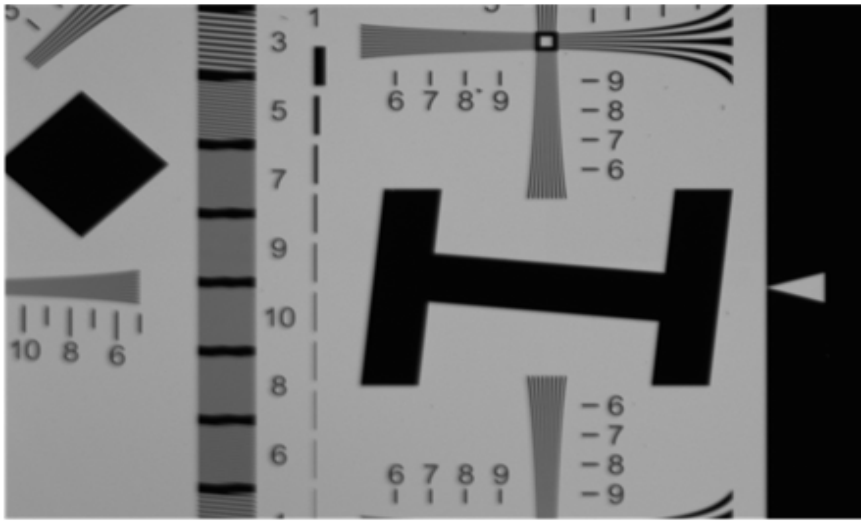
	GTX 580	GTX 670	GTX 780
Compute Cores	512	1344	2304
Cores/SM	32	192	192
Max GFLOPs (SP)	1030	2460	3977
Mem. Bandwidth (GB/s)	192.4	192.2	288.4
Cuda Arch & Capability	Fermi, 2.0	Kepler, 3.0	Kepler, 3.5

GPU	ms/ frame	FPS	Major Updates
580	70	14.3	Initial implementation on GTX 580
580	55	18.2	Remove additional host/device transfers
580	165	6.1	Add demosaicking (essentially free)
580	100	10	Cut num. of transposes by 50% in WT/IWT steps
580	72	13.9	Implement slabs
670	37.5	26.7	Major algorithm modifications – no major impact on output image
780	23	43.5	New hardware; met RT requirement

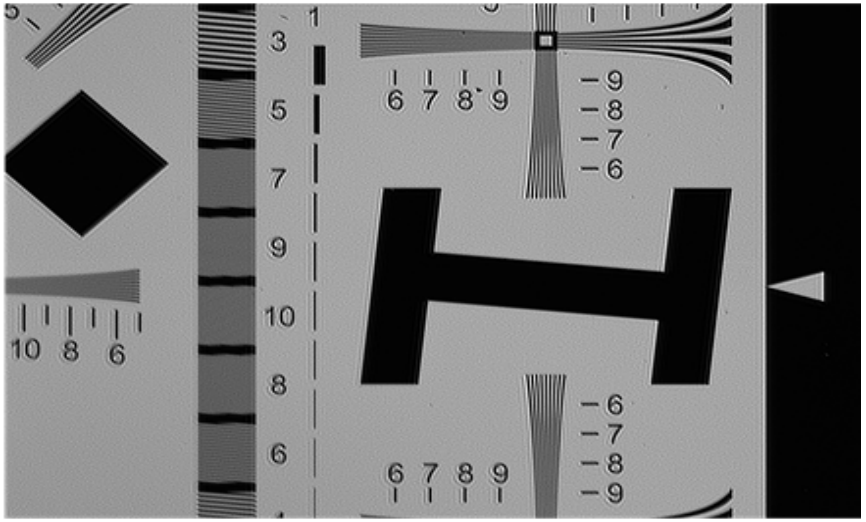


Visual Performance

Blurry



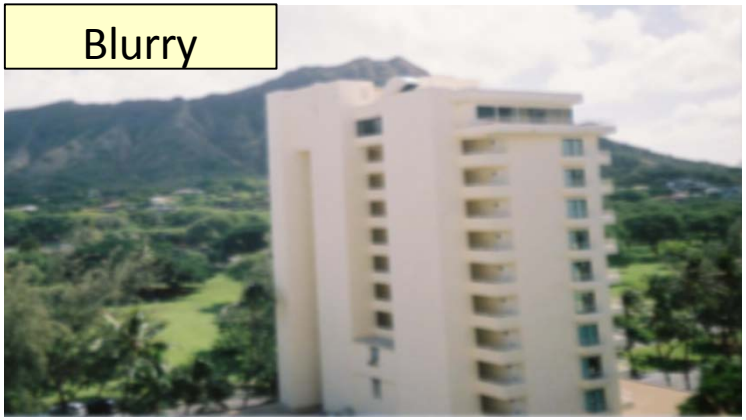
Deblurred



Visual Performance

Original

Blurry



Post-DDGPU

Future Work: Embedded System

- Volume Goal: 2”x2”x3”
  - Likely too small; will be either FPGA or Auto GPU platform (e.g., Nvidia Jetson)
  - Requires implementation improvements
    - Further reduce bandwidth!
- Targets for improvement: Wavelet processing
  - Based on results from attempting FPGA implementation
  - Do full WT, Threshold, IWT step on independent, small pixel blocks
  - Do not store w*X* temporary frames
    - Immediately expand/contract frames
  - Identify other computational approaches to combine row/column based operations (do more per slab)

Summary

- Goal: Implement a system capable of deblurring color HD video at 30 frames/sec
- Accomplished using Nvidia GTX 780 GPU with algorithm modifications
  - Modifications had no impact on visual output
  - Close to achieving with GTX 670 GPU
- Further advancements in implementation needed for embedded system target
  - Several potential improvements identified

Visual Performance: Full vs. Reduced Implementation

Reduced

Bilinear Demosaicking

Full

Original Demosaicking