Multi-Frame Analysis:
The Future of Video Processing

Nikola Bozinovic, VP Engineering

October 2, 2009

GPU Technology Conference
Overview

• Introduction: Video enhancement & reconstruction
  – Background & postulates of multi-frame enhancement

• Applications: next-generation video processing in practice
  – Forensics (Ikena) & Consumer (vReveal)

• Need for speed: the performance
  – GPGPU: making it real (time)

• Multi-frame CUDA development
  – Lessons learned

• Future:
  – Where do we go from here?
MotionDSP: company making video enhancement software

- What is video enhancement?
  - It used to be purely subjective: “whatever looks better to you”.

- New way of doing video enhancement:
  - making video **objectively** better

But also:
Visual (re)evolution – the background

- Communication is becoming increasingly visual
- Video plays a central role

Short history of digital video:

- **Stage I: coding and communication**
  - D1 in 1986, QuickTime in 1990
  - all about coding

- **Stage II: enhancement, characterization**
  - focus shifting to post-processing

- **Stage III: understanding video**
  - Video search, HCI, AI
Digital Video Processing: Stage 1 - Coding

- Transformed the world over the last 20 years
- Main focus: Getting video from point A to point B
  - Original material is a ground truth
- Encoding is a well defined problem
  - Original material is a ground truth
- This problem is **solved**. Solution: AVC/H.264 (no plans for H.265)
H.264 – How it works?

- Hybrid coder = Temporal prediction + spatial transform coding

- Any motion will work (although it will affect coding efficiency), because texture (prediction error) can cover the difference

- Conclusion: motion doesn’t have to be perfect …
H.264: What makes it great?

- ...which doesn’t mean that motion in H.264 isn’t very good (for coding).
- Rate-distortion optimized motion compensation
  - Variable-size block matching (VSBM)
  - Quarter-pixel motion accuracy
  - Motion vectors outside picture boundaries
  - Hierarchical bi-directional prediction
  - Multi-hypothesis prediction
  - Efficient entropy coding of motion
  - Quantized, block-based motion serves the encoding purpose well
Digital video: Stage II - a paradigm shift

10 years ago

- Handful of creators
- Powerful encoders
- Lousy decoders
  - H/W decoding only
- HQ content

Now

- Millions of creators
- Many low-power encoders
- Powerful decoders
  - 100s of GFLOPS
- LQ content
Digital video beyond encoding in this new paradigm

- Better encoding can help, but it’s often limited:
  - Small aperture
  - Cheap (noisy) sensors
  - Cheap DSPs
  - Limited power (battery life)
  - Limited bandwidth/bitrate
  - Poor shooting conditions (low light, camera shake)

- Q: What to do once video is recorded?
  A: Despair: wait for better hardware
  B: Do over: Relive (or reenact) the moment
  C: Improve: apply smart post-processing

- Fortunately, video has a unique property:
  - abundant information about the same scene (unlike audio/stills)
Things that can be fixed

• Poor Resolution

• Noise

• Camera shake

MotionDSP’s software can correct these problems
Is objective video enhancement possible?

- Q: Does it really work? Or is it garbage in/garbage out?
- Can you make something out of nothing?

- No new information can be added to the video (as a whole)
  but …

- Multi-frame processing can increase information in individual frames
Multi-frame video processing

Combines multiple (5-50) frames together to re-construct and enhance video

+ spatial processing
Multi-frame video processing – cont.

• Q: Does it really work?
• A: Yes! Entropy of each individual frame can be increased
• This is perceived as better/clearer video
Digital video enhancement: open-loop structure

Q: Can we simply reuse motion estimated from the encoding part?

• **A: No. Motion needs to be reinvented and re-estimated**
  – similarities with distributed video coding

• There is no ground truth (unlike in coding):
  1. Can not work with quantized motion
     1/4–pixel motion accuracy is not enough, have to use float accuracy
  2. Can’t use block-based model
     Higher-order parametrical models and flow based motion
Core technology: conclusions

Motion for video coding

- Two frames
- Block-matching (simple model)
- Quantized motion vectors (1/4 pel)
- Simple temporal modeling

Motion for enhancement

- Many frames
- True motion (complex model)
- Float motion vectors
- Advanced temporal modeling

- We built first start-to-end multi-frame video enhancement framework
- First to port it all to GPU for faster implementation
MotionDSP - Core processing elements

• Multi-frame
  – Resolution
    • Registration (multi-frame motion estimation)
    • Fusion (ML-type optimization)
  – Stabilization (cheap, once accurate motion is available)
  – Deinterlacing

• Single-frame
  – Deblurring (out-of-focus and motion)
  – Contrast (locally adaptive)
  – Light and color
Motion – Spatial Modeling

• Using mixture of motion models
• Hierarchical Gaussian pyramid over every input video frame
• On each level of a pyramid, motion is a mixture of:
  – Global parametric motion (modeling camera)
  – Local parametric motion
  – Local optical-flow motion
• Occlusion detection
  – Occluded and newly exposed pixels are explicitly modeled
• There are N-1 motion fields per output frame
• [Speed]
  – It is critical to reuse/combine motion mappings when possible
• [Accuracy]
  – It is necessary to refine all motion mappings before using them for SR
There are N-1 motion fields per output frame

[Speed]
  - It is critical to reuse/combine motion mappings when possible

[Accuracy]
  - It is necessary to refine all motion mappings before using them for SR
Motion – temporal modeling

- There are N-1 motion fields per output frame
- [Speed]
  - It is critical to reuse/combine motion mappings when possible
- [Accuracy]
  - It is necessary to refine all motion mappings before using them for SR
• There are N-1 motion fields per output frame
• [Speed]
  – It is critical to reuse/combine motion mappings when possible
• [Accuracy]
  – It is necessary to refine all motion mappings before using them for SR
Parallelization challenges

• Registration
  – Global nature of the problem (i.e., not fixed block like in encoding)
  – Not so easy to parallelize
  – Process is iterative, but # of iterations on the order of 10
  – New data coming in all the time

• Fusion
  – Parallelizes extremely well (once motion is known)
  – Almost for free on CUDA using textures for interpolation

• Stabilization
  – Basically free (all computations on parametric MFs)

• Deblurring (and other single frame filters)
  – Great for parallelization
Overview

• Introduction: Video enhancement & reconstruction
  – Background, postulates of multi-frame enhancement

• Applications: next-generation video processing in practice
  – Forensics (Ikena) & Consumer (vReveal)

• Need for speed: the performance
  – GPGPU: making it real (time)

• Multi-frame CUDA development
  – Lessons learned

• Future:
  – Where do we go from here?
MotionDSP’s Core Technology

Intelligence

Consumer

Core Software
Ikena: Forensics

- Windows application (XP/Vista)
- Laptop and Workstation versions
- “CSI-style” tool for video enhancement
- Imagery Analysis and Video Forensics
- High-profile customers
- GPU accelerated: NVIDIA CUDA
vReveal: Consumer

- **What:** a Windows (Vista/XP) video enhancement app for consumers
- **Why it’s cool:** unrivalled “CSI-Style” video enhancement for consumers
- **Tech requirements:** Runs on any Windows PC (XP or Vista)
  - With CUDA-compatible GPU it runs up to 5x faster
- **When:** Launched March 24th, 2009
- **Available now from MotionDSP (www.vreveal.com) and NVIDIA**
- **Price:** $50
Overview

• Introduction: Video enhancement & reconstruction
  – Background, postulates of multi-frame enhancement

• Applications: next-generation video processing in practice
  – Forensics (Ikena) & Consumer (vReveal)

• Need for speed: the performance
  – GPGPU: making it real (time)

• Multi-frame CUDA development
  – Lessons learned

• Future:
  – Where do we go from here?
Save enhancements to video in vReveal up to 5x faster with the parallel processing power of CUDA-enabled NVIDIA GPUs

The processing speed test measures how many enhanced VGA (640x480) frames vReveal can reconstruct per second in Vista. Best prices avail. from Newegg.com or comparable online store.
Benchmarks

- Rendering Performance (decode/enhance/encode/save to disk)
- QCIF and QVGA output at 2x original resolution
- VGA output at 1x original resolution

- Vista overhead caused by WDDM
- Vista driver is partially implemented in user mode, API to access the kernel
Overview

• Introduction: Video enhancement & reconstruction
  – Background, postulates of multi-frame enhancement

• Applications: next-generation video processing in practice
  – Forensics (Ikena) & Consumer (vReveal)

• Need for speed: the performance
  – GPGPU: making it real (time)

• Multi-frame CUDA development
  – Lessons learned

• Future:
  – Where do we go from here?
First example - Problem definition

Our setup:

- Complex, real world video application
- Multi threaded environment
- Filters added and removed dynamically
- Multiple executions of a same filter with different parameters

Practical problem: Memory allocation & deallocation
CUDA memory allocation

• Memory allocation in CUDA is expensive

• Our first solution: allocate in advance
  – Large memory consumption
  – Complex, error prone, code. Why?

• We are allocating same memory sizes all over again!
  – Plus execution is periodical

• Our next solution: Simple memory manager
  – Singleton for managing CUDA memory
  – Reusing same pointers
CUDA memory manager

- Hash table of memory records

- Each record contains:
  - GPU pointer, size, thread id, age

- Two main operations:
  - malloc(), free()

- Secondary operation:
  - tick()
CUDA memory manager – cont.

- **malloc**
  - `MemRecord malloc(int size)`
  - Searches hash table for size and thread id.

- **free**
  - `void free(MemRecord& rec)`
  - Returns memory record to hash table.

- **tick()**
  - Is periodically called.
  - Increments age.
  - If memory record get old it releases it.
CUDA smart pointer

- \texttt{template<class T> class CUDA\_pointer}
- Uses memory manager.
- Overrides \texttt{operator T*}

- Really simple usage:
  - \texttt{CUDA\_pointer<float> ptr(width*height);}
  - Use as \texttt{float*}
  - That’s all!
CUDA memory manager - Conclusion

- Faster execution
  - Removed fixed 10ms per frame

- Smaller memory footprint
  - Max filter consumption vs the sum

- Much, much simpler code
  - Faster prototyping and development
Second example - Problem definition

- Our case:
  - Gaussian convolution (convolution) heavily used
  - 50-70 convolutions per frame
  - Convolution used 60% of processing time

- We used `convolutionSeparable` from CUDA SDK
- Turned out we had to optimize it more
Optimized convolution

• First step:
  - Use very simple CUDA kernel for 3x3 convolution

    float central = src[ind_src];
    float left = (xi > 0) ? src[ind_src-1] : central;
    float right = (xi < width-1) ? src[ind_src+1] : central;

    dst[ind_dst] = a*left + b*central + c*right;

• Second step:
  - Sum of two Gaussian's is also a Gaussian
  - \( G(r_1, s_1^2) + G(r_2, s_2^2) \sim G(r_1+r_2, s_1^2+s_2^2) \)
  - Approximate general size convolution with 3x3
Optimize convolution

- Works faster than separableConvolution
  - But still not much faster

- Remark:
  - Row convolution works much slower than column
  - Misaligned float memory access in row convolution

- Solution:
  - Column convolution and transpose in same kernel
  - Again column convolution and transpose
Optimize convolution - Transpose

• Naive transpose:
  - $(i,j) \rightarrow (j,i)$.
  - Works slower then without transpose

• Efficient transpose:
  - Transpose thread block in shared memory
  - Write transposed block to global memory

• Now works really fast
  - About 60% faster then separableConvolution
__global__ void convolution_col_121_transpose(float* dst, int dpitch, float* src, int spitch, 
    int width, int height, float a, float b, float c)
{
    int xi = blockIdx.x*blockDim.x + threadIdx.x;
    int yi = blockIdx.y*blockDim.y + threadIdx.y;
    int ind_src = spitch*yi + xi;
    __shared__ float tmp[256];

    if ((xi<width) & (yi<height))
    {
        float central = src[ind_src];
        float up = (yi > 0) ? src[ind_src-spitch] : central;
        float down = (yi < height-1) ? src[ind_src+spitch] : central;

        // Store conv to shared mem.
        tmp[threadIdx.y*16+threadIdx.x] = a*up + b*central + c*down;
    }

    __syncthreads();

    int x_dst = blockIdx.y*blockDim.y + threadIdx.x;
    int y_dst = blockIdx.x*blockDim.x + threadIdx.y;

    if ((x_dst<height) & (y_dst<width))
    {
        // Transpose from shared memory to global.
        dst[dpitch*y_dst + x_dst] = tmp[threadIdx.x*16+threadIdx.y];
    }
}
Optimize convolution - Conclusion

- Convolution is heavily used
- 70 convolutions per frame
- 60% of execution time

**Optimize:**
- Use simple kernels for small convolutions
- Approximate large convolution with small ones
- Avoid misaligned memory access
- Use efficient transpose
Overview

• Introduction: Video enhancement & reconstruction
  – Background, postulates of multi-frame enhancement

• Applications: next-generation video processing in practice
  – Forensics (Ikena) & Consumer (vReveal)

• Need for speed: the performance
  – GPGPU: making it real (time)

• Multi-frame CUDA development
  – Lessons learned

• Future:
  – Where do we go from here?
Vision

- MotionDSP’s software in next-generation video applications

**Video Filters (Premiere-style)**

**Move to device**

**Display**

**Video sharing**

**Video Conferencing**

- Platforms: **CUDA**, but also OpenCL, Larrabee, DirectX11
- Open and powerful video framework on a client, enabling exciting new applications
Acknowledgments

- Entire MotionDSP team, esp. engineering team in Nis, Serbia
  - Ivan Vuckovic, Ivan Velickovic, Nemanja Grujic and everybody else
- Prof. Milanfar, UCSC, Prof. Konrad Boston University
- In-Q-Tel, NVIDIA
- Contact: nikola@motiondsp.com
Questions?