A Super-Resolution Video Player Based on GPU Accelerated Upscaling from Local Self-Examples

Yen-Hao Chen (陳韻豪), Yi-Da Wu (吳易達), Chi-Wei Tseng (曾智為)
Instructed by Wei-Chao Chen (陳維趙)

ABSTRACT
Many super-resolution algorithms have been proposed for upsampling static images, yet upsampling video footages in real-time with descent quality remains a challenging problem. In this project, we started out developing a super-resolution video player based on the work by Gilad Freedman and Raanan Fattal [4], for their proposed approach is essentially suitable for application in video upscaling. The algorithm take advantage of the fact that a nature image patch is redundant in its locality, hence, utilizes neighboring patches as hints on the high-frequency detail that is lost in interpolative upscaling. High memory locality makes this algorithm an excellent candidate to be implemented on commodity GPUs. With CUDA, we achieved a 30x acceleration and a frame rate of 10 fps for 1.5x upscaling on VCD quality video. Through analyzing our resource usage, higher frame rates are anticipated through further development.

- from Image to Real-Time Video Upscaling

Classes of Approaches Proposed for Upscaling Images

- Multiple Frame Image Reconstruction
- Example-Based High Frequency Component Guessing
- Example-Based High Frequency Component Guessing

Challenges in Extending Algorithms to Video in Real-Time
- High computational effort is required for descent quality in upscaled video.
- Quality of output video depends largely on video content.
- Incoherence between frames arises due to patch-by-patch nature in algorithms.

- Strategies for Accelerating with GPU

Taking Advantage of Locality in Nearest Patch Searching
- Storing input and intermediate images in texture memory.
- Efficient for memory access patterns with high 2D locality.
- A thread is responsible for matching one pair of patches

Implementing Efficient Non-Dyadic Filters
- Load non-dyadic filters into constant memory for efficient computation.
- A thread upsamples or downsamples one row of the image.

Utilizing CUDA-Specific Optimization Functionalities

CUDA Stream Processing
- Stream processing enables us to pipeline video frame through the interpolation, downsampling and patch matching steps.

Interoperating with OpenGL
- Directly shows the upscaled result in GPU. This reduces the time spent on moving frames between host and device memory.

Improvements in Efficiency

<table>
<thead>
<tr>
<th>Acceleration</th>
<th>CPU Version</th>
<th>Nvidia GPU Version</th>
<th>Texture/Constant Memory Utilized</th>
<th>Reorganize-Register Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing Time Per Frame</td>
<td>4500 ms</td>
<td>800 ms</td>
<td>220 ms</td>
<td>120 ms</td>
</tr>
<tr>
<td>Improvement</td>
<td>1.5x</td>
<td>5.6x</td>
<td>20.5x</td>
<td>37.5x</td>
</tr>
</tbody>
</table>

- Image Upscaling from Local Self-Examples

Patch Redundancy in A Restricted Window
- Locally, a slightly downsampled image looks similar to a patch from the original image.
- On upsampling, utilize the original image in a restricted window as a hint for high frequency detail.

Algorithm Steps
- U: Interpolation Operator
- D: Downsampling Operator
- U and D are implemented with non-dyadic filter banks.
- Repeat several times for larger upscales.

Results from Single Frame

Left: Original image. Right: Image of super resolution.

Potential in Applying to Real-time Video
- It is a highly parallelizable algorithm, implementing on GPU is beneficial.
- Consistency is achieved between upsampled and original image: Guarantees output stability.

- Challenges, Conclusion and Future Work

We have built a super-resolution video player which upscales frames from local examples. Up to poster submission, the achieved frame rate is 10 fps on upsampling video of VCD quality 1.5x, which is not satisfactory.

Bottleneck Analysis
- The smoothing and upsampling kernel, which intensively write to global memory, dominates the timeline.

Prospective
- The relatively simple image smoothing and upsampling procedure takes up most of the computation time.
- The power of shared memory is not utilized yet. Incorporating shared memory into the algorithm will be our next step.

- References


2012 GPGPU HD Koala Bears Team