GPU ACCELERATING SUPER-RESOLUTION FOR CONVERTING HD TO 4K

ZHANGZONG ZHAO¹, LI SONG¹, CHUANFEI LUO², RONG XIE¹, XIAOKANG YANG¹, WENJUN ZHANG¹

¹Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
²Shanghai Research Institute of China Telecom

Shanghai, China
# Introduction

## Background

Audience expects high resolution videos/images to enjoy high quality visual experience. Currently the video content providers have no many “true” 4K video contents. In contrast, there are a large amount of videos/films on the resolution of 1080p in market. Thus, it is desired to convert the 1080p videos to 4K videos offline with high quality and a reasonable running time.

## Existing SR Methods

Quality and speed are two basic requirements for the video super-resolution technique. Early upscaling methods (e.g., bicubic, Lanczos) typically have low complexity and computational cost, but the quality of their upscaled images is relatively poor. Recent state-of-the-art SR methods produce high quality images with extremely large computational cost. Existing GPU accelerating methods achieve significant speed-up at the cost of degraded quality. Generally they are either too slow or of poor quality.

# HD to 4K framework

We propose a super-resolution based HD to 4K video converting. It first decodes the HD video from high quality Blu-ray disc, then transfers the data from host memory to GPU memory and performs the GPU accelerated A+(Adjusted Anchored Neighborhood Regression)[1,2] super-resolution. After that, additional enhancement operations like denoising or color grading can be conducted. Finally, the 4K image clips are re-encoded into the appropriate 4K video format for distribution purpose. In the following parts of this poster, we will introduce our GPU implementation and optimizations of A+ super-resolution, to accelerate it from 47s/frame to 0.16s/frame, which is no longer the bottleneck of this HD to 4K converting procedure.
The prototype of our super-resolution algorithm is the A+ (Adjusted Anchored Neighborhood Regression). The A+ not only produces the best quality among the state-of-the-art SR method to our knowledge, but also requires a moderate computational cost. It searches for the best match patch from the precomputed example dictionary and apply corresponding projection matrix to the input patch feature. The example dictionary and projection matrix dictionary is precomputed offline in training phase, so that they can be directly accessed in the executing phase.

GPU Accelerating A+
In our work, we propose GPU implementation and optimization technologies to accelerate the A+. In order to get maximum ratio of acceleration and minimize CPU/GPU data transfer, the A+ process is fully parallelized. We separate A+ into many steps, parallelize each step and GPU implement them. We also optimize it by taking the advantage of GPU's coalesced memory access mechanism, instruction level parallelism and CPU/GPU hybrid implementation. Our experiments show that the overall execution time for 1080p to 4K upscaling is reduced from 47s/frame to 0.16s/frame, while the image quality is exactly the same as the original A+.

Steps of A+
(1) A bicubic interpolation as the preprocessing to get the LR Image.
(2) The Difference Image is generated by first order and second order difference of LR Image along height and width.
(3) Difference Features and low frequency patch are collected
(4) The LR Features is obtained by PCA (principle component analyzing) the Difference Feature.
(5) The ANR (anchored neighborhood search and regression) step is the most critical, magical, and time consuming one, consisting a matrix-matrix multiply, a column-wise max search, and a column-wise matrix-vector multiply.
(6) Low Frequency Patch and High Frequency Patch are added to generate the HR Patch
(7) HR Patches are overlapped back to SR Image. All seven steps must be parallelized and GPU implemented to parallel A+ entirely.
**Parallelization of A+ Steps**

### Basic Methods
The 2nd (difference), 3rd (collecting feature) and 6th (low-high frequency patches addition) steps are parallelized in a similar way. The problem is split into thousands of millions of mini-tasks along width and height or along features. Each task of generating corresponding output pixel or corresponding output feature is assigned to a unique thread. Because there are no interdependencies between the mini-tasks, they can be processed independently, without intercommunication and executed in arbitrary sequence.

### Parallelism of ANR
It is composed of a matrix-matrix multiply, a feature-wise max search and a feature-wise matrix-vector multiply. The key problem is split along features into mini-tasks and assigned to threads, as shown in figure. In each task, a max absolute value and index search is performed in corresponding column inside the Match Matrix, and then the index's projection matrix is multiplied to the LR Feature to generate the High Frequency Patch.

### Parallelism of overlapping
This algorithm cannot be directly parallelized because there might be concurrent read/write operation to the same position and results in wrong answers. To deal with this problem, we handle the overlapping patches from the view of SR Image. To be concurrently solved, the problem is split along pixels of SR Image. A unique thread is responsible for generating one pixel of SR Image. It first finds the patches that includes the pixel, then reads and averages those pixel values inside corresponding SR Patches. The race issue is fixed by avoiding writing to same location.
Further Optimization

Coalesced memory access
When a thread inside GPU access a global memory, the hardware will automatically combine or coalesce the request with other thread’s request, if these requests access adjacent memory locations. We elaborately rearrange the data formatting to meet the coalesced memory requirement. In the steps of bicubic, difference, collecting features, matrix multiply, feature addition and overlapping, the memory is fully coalesced accessed to have a speed up of more than 10x. In the ANR (5th step), the memory access to the Match Matrix is also fully coalesced, but we fail to speed up the matrix-vector multiply because the location of desired Projection Matrix is undetermined before execution.

Instruction level parallelism
We exploit instruction level parallelism (ILP) to further accelerate the feature-wise matrix-vector multiply. Because the access to Projection Matrix is uncoalesced, the hardware will launch a 32 bytes read request while only 4 bytes (one floating-point number) of them is useful and the rest are discarded. To improve this situation, we adjust the instructions and data type inside the kernel. The read operations use data type float4 to launch read request of 16 bytes instead of 4 bytes.

CPU/GPU Hybrid Implementation for Video Processing
We propose the CPU/GPU hybrid implementation, to concurrently process the Y/U/V channels. The A+ procedure of Y channel is assigned to GPU and bicubic procedures of U/V channel are assigned to two CPU threads. The concurrent execution can be achieved as the CPU procedure of bicubic is faster than the GPU procedure of A+. Thus the CPU/GPU hybrid implementation reduce the video super-resolution execution time from $3^* t_{I/O} + t_{GPU A+} + 2 t_{GPU bicubic}$ to $t_{I/O} + t_{GPU A+}$.
Experiments
We conducted experiments about the execution time for each step processed by CPU and GPU. A significant speed up of 295x is achieved. The overall execution time is reduced to 160ms.

<table>
<thead>
<tr>
<th></th>
<th>CPU(ms)</th>
<th>GPU(ms)</th>
<th>Speed up</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bicubic</td>
<td>27</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>Difference&amp; Collect Feature</td>
<td>8,157</td>
<td>8</td>
<td>1020</td>
</tr>
<tr>
<td>PCA</td>
<td>384</td>
<td>15</td>
<td>25</td>
</tr>
<tr>
<td>ANR</td>
<td>17,924</td>
<td>125</td>
<td>143</td>
</tr>
<tr>
<td>Patches Addition</td>
<td>69</td>
<td>3</td>
<td>23</td>
</tr>
<tr>
<td>Overlapping</td>
<td>20,654</td>
<td>2</td>
<td>10327</td>
</tr>
<tr>
<td>Overall</td>
<td>47,300</td>
<td>160</td>
<td>295</td>
</tr>
</tbody>
</table>

All implementations, optimizations and measurements were done on a PC with dual Intel E5-2697v2 @2.7GHz 12 cores processors, 64GB host memory, NVIDIA GTX980Ti. The experiment is done on a 1920x1080 to 3840x2160 single channel common image super-resolution. The result is listed in the table.

Correctness Verification
We conduct additional image quality experiment to verify our implementation and measure the difference of computational accuracy between CPU and GPU. The correctness of our parallelism and codes can be verified. The GPU accelerated A+ achieves identical quality as the original CPU A+, which is far better than simple bicubic and better than other state-of-the-art SR methods.

<table>
<thead>
<tr>
<th></th>
<th>PSNR(dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU A+ vs GPU A+</td>
<td>88.97</td>
</tr>
<tr>
<td>Ground Truth vs CPU A+</td>
<td>37.84</td>
</tr>
<tr>
<td>Ground Truth vs GPU A+</td>
<td>37.84</td>
</tr>
<tr>
<td>Ground Truth vs Bicubic</td>
<td>35.37</td>
</tr>
</tbody>
</table>

Conclusion
We propose a GPU implementation on one of the best image super-resolution algorithm: A+. In our work, the A+ is divided into 7 steps, all of which are parallelized by GPU acceleration. Specifically, we optimize the code by three techniques: coalesced memory access, instruction level parallelism and CPU/GPU hybrid implementation. Our experiments show the execution time for converting a video frame of 1920x1080 to 3840x2160 can be reduced from 47.3s to 160ms, a 295x speed up, while the quality of GPU result is exactly the same as CPU result.

Reference

Acknowledge
The authors would like to thank the following projects: The Shanghai Zhangjiang national independent innovation demonstration zone development fund (No.201501-PD-SB-B201-001) and the NSFC (61521062, 61527804, and 61420106008), and the Shanghai Key Laboratory of Digital Media Processing and Transmissions.

Contact Detail
First Author
Name: Zhangzong Zhao
Email: dragon_nsc2@Hotmail.com
Affiliation: Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University
Corresponding Author
Name: Li Song
Email: song_li@sjtu.edu.cn
Affiliation: Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University