GPU-accelerated Texture Decompression of Biomedical Image Stacks

Chirantan Ekbote[1,2], Won-Ki Jeong[1], Jens Schneider[3], Hanspeter Pfister[1]

[1]School of Engineering and Applied Sciences, Harvard University
[2]Department of Computer Science and Engineering, The Ohio State University
[3]Geometric Modeling and Scientific Visualization Research Center, King Abdullah University of Science and Technology (KAUST)

Introduction

Background: Histopathology is the microscopic examination of tissue in order to study the manifestations of disease. High resolutions images are vital for accurate diagnoses and a major obstacle to the use of digital imaging in histopathology has been the inability to display these large images at interactive rates.

Figure 1: Conventional optical microscope on which a pathologist views glass slides moved by hand.

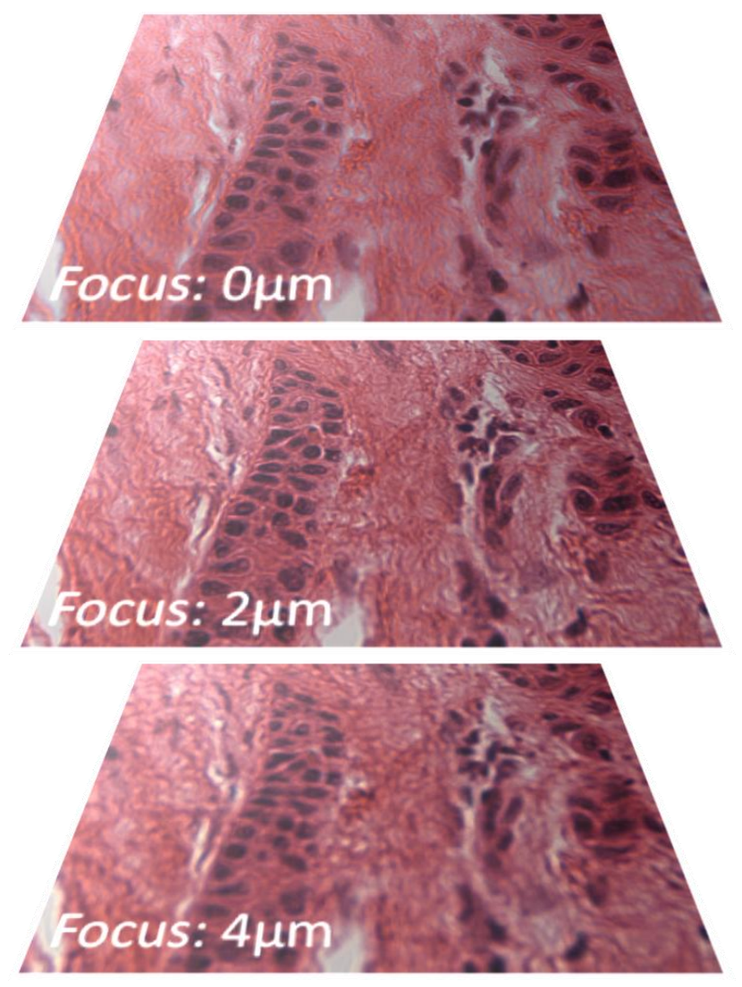


Figure 2: Stack of optical microscope images. Each image is taken with a different focal length

Purpose: Create a tool for interactive visualization of biomedical image stacks using GPU-accelerated on-the-fly texture decompression [1]. The image stacks are compressed using a novel approach custom tailored for the data we are dealing with, i.e. data exhibiting exceptionally high coherence between the slices of each image stack.

Compression

The compression algorithm we employed is the Lloyd's method, which reduces the data size by encoding a cluster of similar pixels to a single element in the codebook.

The stack compression relies on the fact that every image is very similar to the images above and below it and is accomplished by:

1. Jointly encoding only the top and bottom images of the stack.
2. Using linear interpolation to predict the intermediate slices and only encoding the differences between the actual and predicted values.
3. Ensuring the number of bits used to store the differences is less than the number of bits that would be necessary to store the image itself.

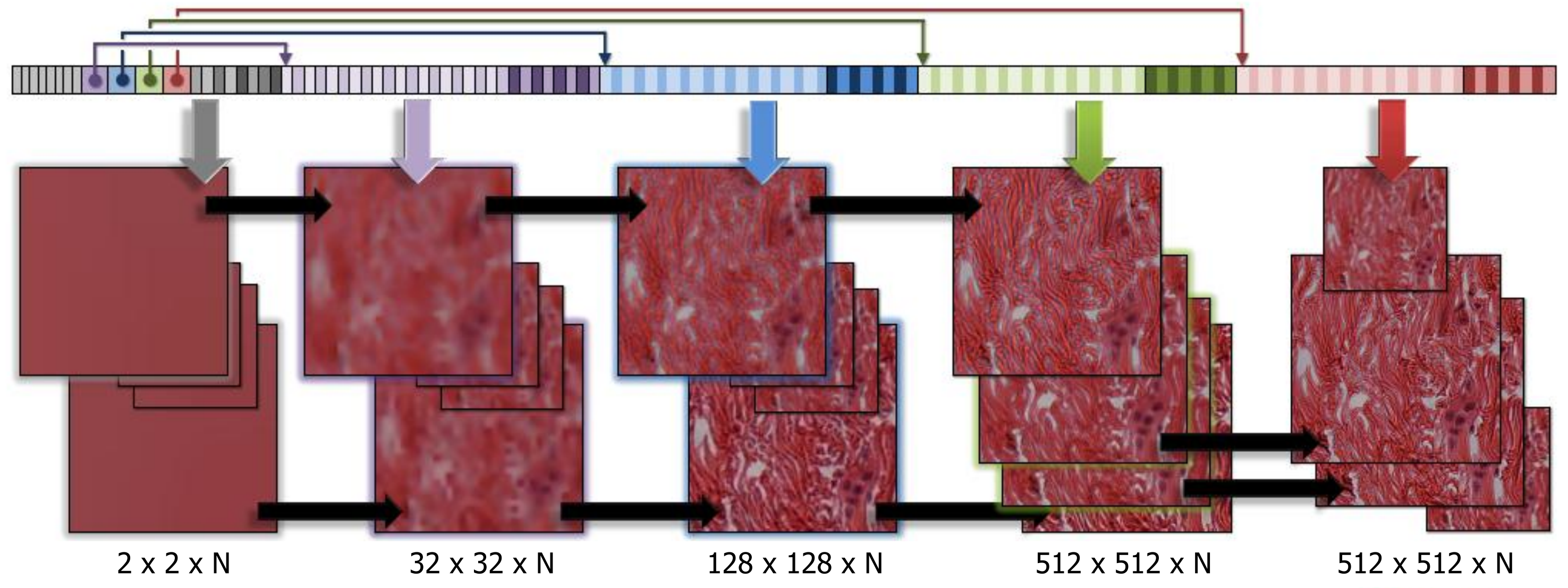
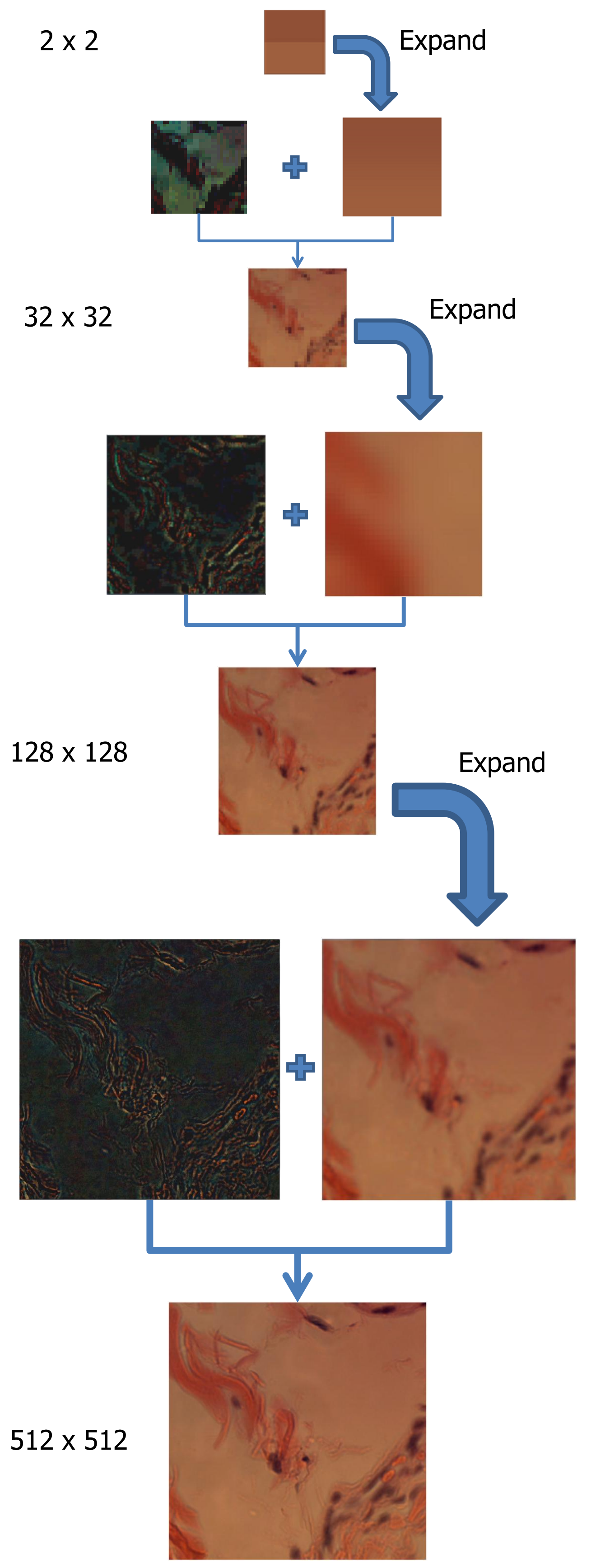


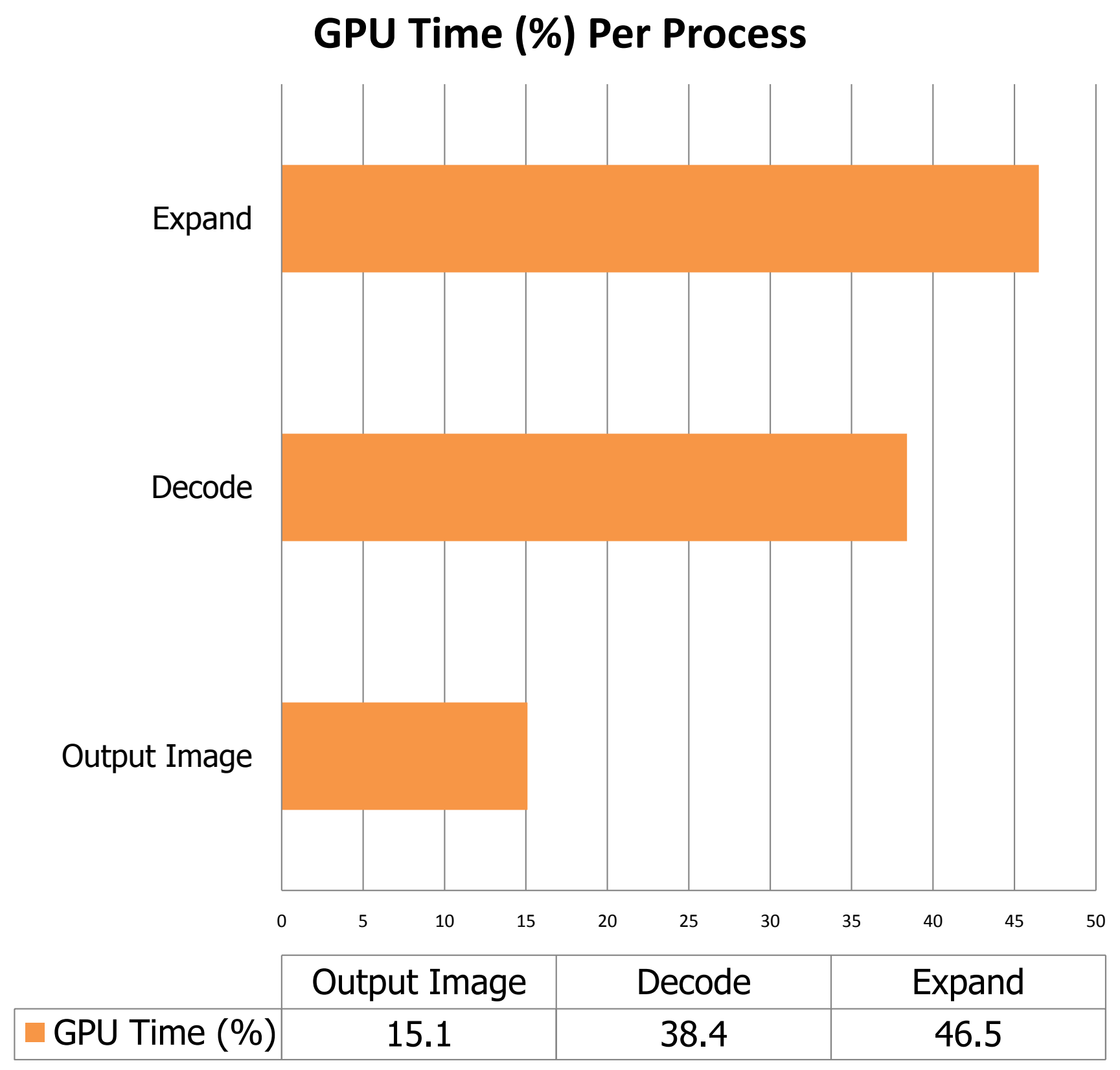
Figure 3: Storage layout of the compressed data. Only the top and bottom slices are encoded while linear interpolation is used to predict and delta-code the intermediate slices.

Decompression



Reconstruction of the images is then accomplished by simply reversing the encoding process:

1. Fetch the 2x2 pixels for the top and bottom images and linearly interpolate the values to get the desired slice.
2. Expand the image to 32x32 pixels.
3. Fetch and incorporate the encoded differences into the predicted image.
4. Repeat step 3 once by expanding the image to 128x128 pixels and once more by expanding to 512x512 pixels.

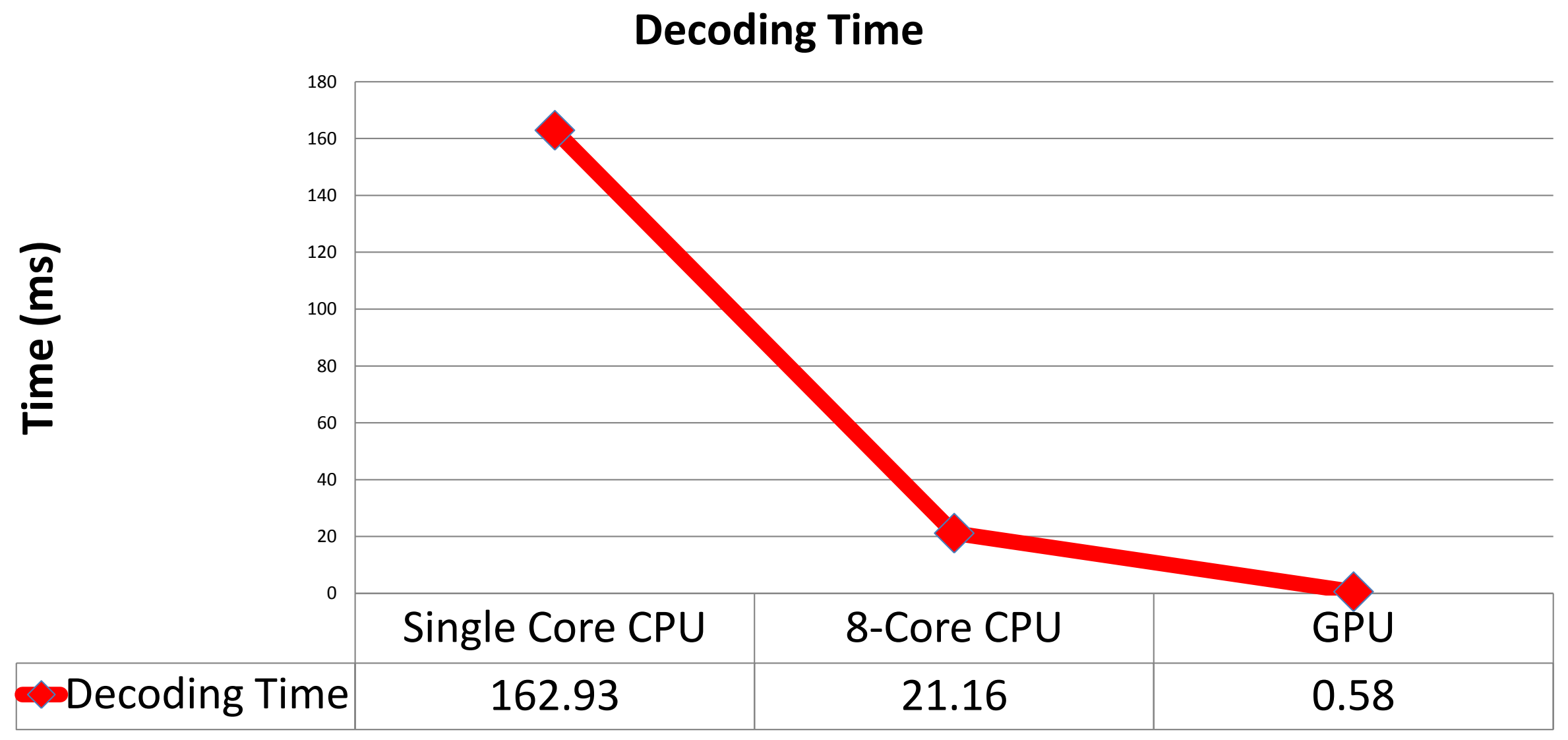


	Output Image	Decode	Expand
🟧 GPU Time (%)	15.1	38.4	46.5

Figure 4: Overview of the stages of decompression (left) and GPU time for each decompression stage (right).

Results

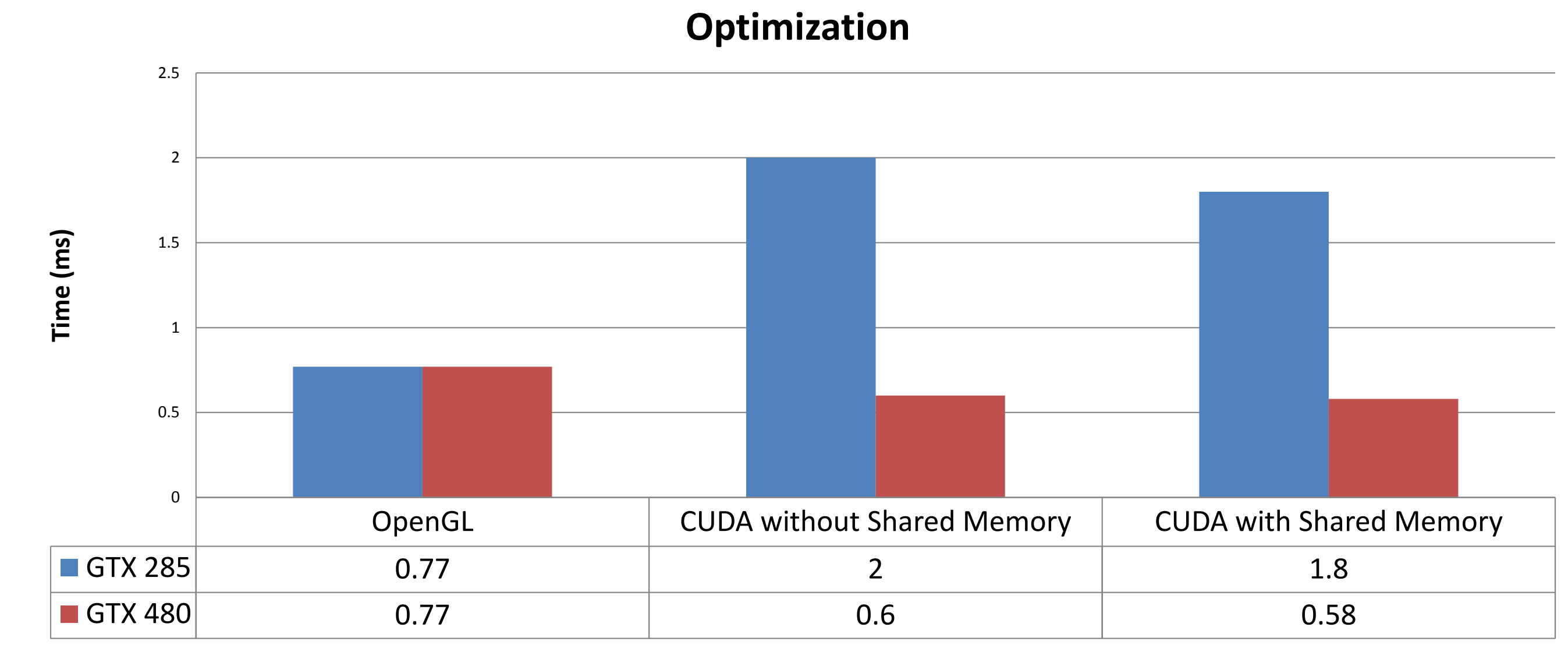
CPU vs. GPU: Decoding a single slice on the GPU was more than **36x** faster than on an 8-core CPU. This is because the highly parallel architecture of the GPU enables decoding of blocks of pixels together, significantly reducing computation time.



	Single Core CPU	8-Core CPU	GPU
◆ Decoding Time	162.93	21.16	0.58

OpenGL vs. CUDA / GTX 285 vs. GTX 480:

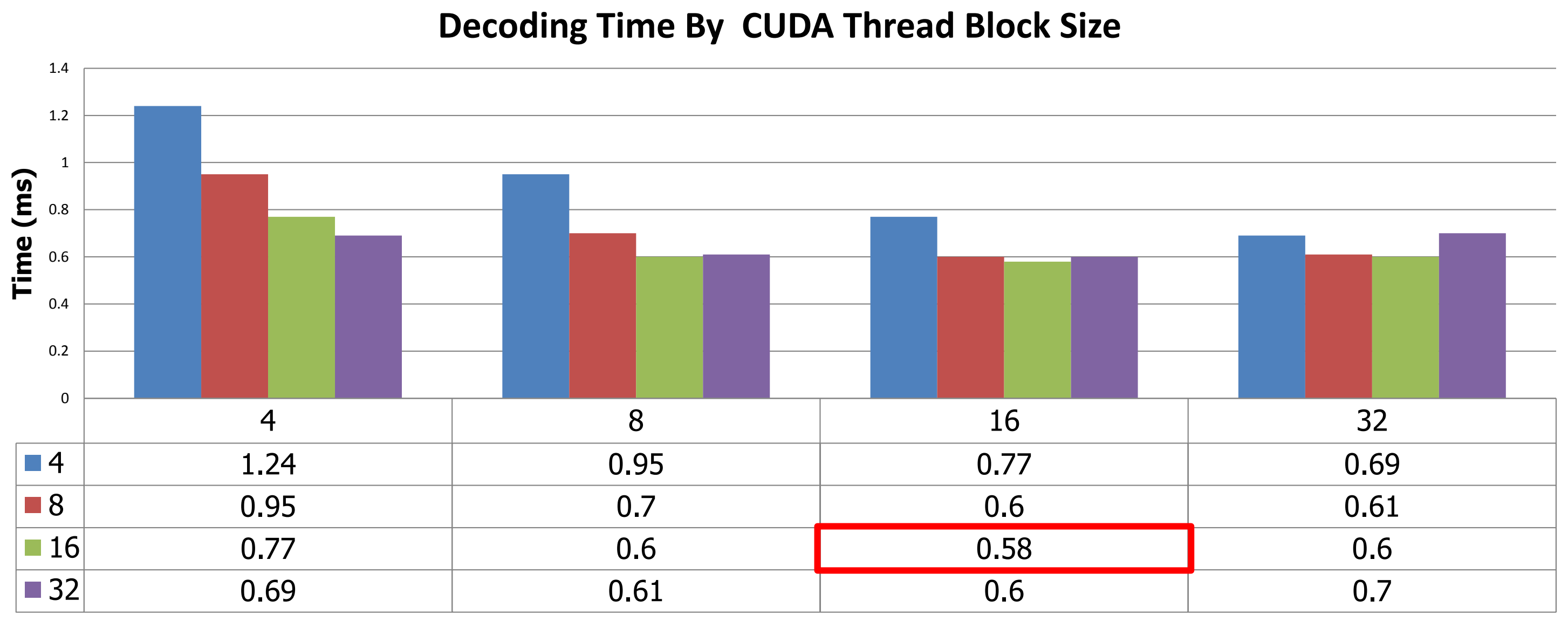
- Decoding time for a single slice using OpenGL was same on the GTX 285 and 480, although CUDA performance was much worse on the 285.
- Using shared memory provided a **10%** increase in speed over using global memory on the GTX 285.
- Decoding a single slice on the GTX 480 was **24%** faster using CUDA than using OpenGL, mainly due to the efficient memory management and the L1/L2 hardware cache. A side effect of having the cache was that using shared memory provided no significant advantage over using global memory.



	OpenGL	CUDA without Shared Memory	CUDA with Shared Memory
🟦 GTX 285	0.77	2	1.8
🟥 GTX 480	0.77	0.6	0.58

Kernel Block Dimensions:

- Decoding times were recorded for varying thread block dimensions of each kernel call from 4x4 threads per block to 32x32 threads per block.
- Empirically, we found that using a block size of 16x16 threads per block was ideal for our implementation.



	4	8	16	32
🟦 4	1.24	0.95	0.77	0.69
🟥 8	0.95	0.7	0.6	0.61
🟩 16	0.77	0.6	0.58	0.6
🟪 32	0.69	0.61	0.6	0.7

Conclusion

Summary: We have created a tool for pathologists to visualize large high-resolution optical microscopy image stacks at interactive rates. This is accomplished by a novel variation on predictive hierarchical vector quantization that can be fully decoded on the GPU.

Future Work: Test and implement the compression algorithm and decoder for electron microscopy data to incorporate it into a framework for semi-automatic segmentation and visualization of neural processes.

References

[1] Won-Ki Jeong, Jens Schneider, Stephen G. Turney, Beverly E. Faulkner-Jones, Dominik Meyer, Rüdiger Westermann, Clay Reid, Jeff Lichtman, Hanspeter Pfister, Interactive Histology of Large-Scale Biomedical Image Stacks, IEEE Transactions on Visualization and Computer Graphics, 2010 (to appear).