COMS W4115.1 – PLT

MatPix: Proposal

Group: David Burkat, Oliver Cossairt, Robert Hawkins, Ben London

We plan to implement a programming language for performing matrix arithmetic on the graphic processing unit (GPU). Modern GPUs are designed to perform extremely fast, highly parallelized matrix processing; but their power is often under-utilized by most common computing tasks. By offloading certain specialized processes to the GPU, one can expect massive performance gains.

Essentially, the compiler will translate our code into a combination of C and OpenGL calls. The compiler will have to decide, based on the nature of the each operation and the state of the memory, whether a task should be offloaded to the GPU. The biggest bottleneck in GPU processing is the bus. This means that repetitive processing of small matrices on the GPU might actually be slower than just using the CPU, because there will be more data transfer than processing; conversely, processing of larger matrices will be faster, by the same reasoning. The compiler should (ideally) take this into account and optimize the amount of data transferred to and from the GPU. However, these optimizations might be outside of the scope of this project.

Most CPU computations can be mapped to an equivalent GPU computation, but the mapping is not identical. GPU computations are typically treated as stream computations whereas arrays are processed by a kernel. Two important aspects of implementing our GPU backend will be array reduction and memory mapping. GPU stream computations do not naturally perform reductions from an array to a smaller array. There are established techniques to accomplish reduction, but it will take some work to integrate them into our compiler. Likewise, GPU memory is designed for efficient and simple allocation and read/write operations on 2D arrays of fixed maximum size. For simplicity, we will initially restrict our language to scalars, 1D and 2D arrays of bounded size. We will reassess the feasibility of implementing arbitrary size and dimension arrays after this initial implementation.

Naturally, the mixture of C and OpenGL will be invisible to the user. Syntactically, our language will be similar to Matlab, which is a well-known and easily intelligible language for matrix math. It will include all of the typical arithmetic operations for scalars, vectors and matrices, as summarized in the following:

- Add
- Subtract
- Multiply
- Divide (scalar only)
- Matrix Inverse
- Exponentiation (scalar only)
- Dot product

We will also implement the basic logical operators: AND, OR and NOT. Also important is variable allocation and array indexing, which will most likely follow a C syntax. Boolean operations are not implemented efficiently on the GPU, so they may be implemented in C code. Lastly, we would like to implement some basic control structures, such as: IF / ELSE, FOR and WHILE. Time permitting, we would like to implement an interesting library function, like the FFT.