CATEGORY: MEDICAL IMAGING & VISUALIZATION - MV06
POSTER P4210
CONTACT NAME Joakim da Silva: jd491@cam.ac.uk

GPU TECHNOLOGY CONFERENCE

Fast Kernel Superposition for Proton Therapy

Joakim da Silva[1,2], Richard Ansorge[1], Rajesh Jena[2]

[1]BSS, Cavendish Laboratory and [2]Department of Oncology, University of Cambridge

Adaptive proton therapy

Radiotherapy, where a tumour is irradiated with ionising radiation from an external source, is the preferred treatment for many types of cancer. Conventional radiotherapy uses x-rays, which deposit their energy in an exponentially decaying fashion when traversing matter. Conversely, beams of charged particles such as protons deposit the majority of their energy at a well-defined, energy-dependent depth producing a "Bragg peak" (Figure 1). By making the Bragg peak coincide with the target, radiotherapy using protons thus allows for sparing of healthy tissue both upstream and downstream of the tumour. With this precision comes the need for adaptive treatment with daily, or even real-time, patient imaging and treatment adaptation to compensate changes in patient position and anatomy. A crucial part of an adaptive proton therapy (APT) system is the ability to recalculate the dose distribution rapidly.

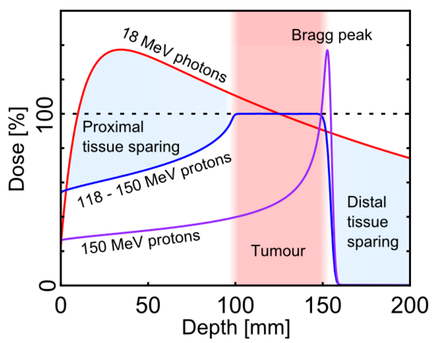

Figure 1. Comparison between depth-dose distributions for proton and photon beams. After a build-up region near the skin, photons (red) deposit their energy in an exponentially decaying fashion. Protons (purple) deposit the majority of their energy in the Bragg peak, making it possible to create a spread-out Bragg peak (blue) that coincides with the tumour and thus spares healthy tissue both upstream and downstream of the tumour.

The pencil beam algorithm

The gold standard for radiotherapy dose calculation is Monte Carlo (MC) simulation. However, MC simulations for charged particles are time-consuming and, due to the multitude of new particles created along the beam tracks, lend themselves poorly to direct implementation on SIMD architectures. The pencil beam algorithm (PBA; Figure 2), which generates approximate dose distributions much faster than MC simulations, has therefore become clinical standard. Conceptually, the PBA can be divided into two steps: i) the ray tracing, during which a laterally integrated dose is deposited along the rays according to their water-equivalent path-length (WEPL); and ii) the superposition, where the integrated dose is spread laterally to account for widening of the beam due to multiple Coulomb scattering [1]. If the width of the kernel is dependent on the ray path, the second step cannot be replaced by a 2D convolution. Instead, a Gaussian kernel superposition of complexity $O(N^2)$ in kernel width has to be carried out, making the complete calculation $O(N^5)$ in dose resolution. For accurate proton dose calculations, path-dependent beam widening is required, making the second step of the algorithm by far the most time-consuming.

$$D(x,y,z) = \sum_i I_i \times d_i(E_i, z_{\text{eff}}) \times \frac{1}{2\pi\sigma(z_{\text{eff}})^2} e^{-r_i^2/2\sigma(z_{\text{eff}})^2}$$

Figure 2. The pencil beam equation. The dose D at a given point is a sum over contributions from rays i. I is the ray intensity, d the laterally integrated dose, E the initial ray energy, z_{eff} the water-equivalent path-length, σ the beam width and r the lateral distance to the ray.

CUDA implementation

Several CUDA implementations of the 2D kernel superposition were developed. The most efficient was found to be a scatter approach which relies on implicit warp synchronisation (Figure 3). The performance was further improved by factors of 1.0 – 3.9 for different kernel widths if the shared memory used was declared using the `volatile` keyword, in which case a `__syncthreads()` statement of the inner-most loop could be omitted. A scatter approach also has the advantage that the entire calculation can be omitted if all the input values in a thread block equal zero, which often occurs in calculation of real treatment plans, reducing the amount of superfluous operations. In addition, the entire PBA, including the ray tracing, kernel width calculation and coordinate system transformation, was implemented as a sequence of CUDA kernels (Figure 4). As a result, all intermediates can be created and kept in GPU memory throughout the calculation, keeping the data transfers between the CPU and the GPU to a bare minimum, which improves the performance compared to other approaches.

```
template <int rad>
__global__ void gaussSuperposition(float *in, float *sigmas, float *out)
{
  extern volatile __shared__ float tile[]; // Holds thread block's result
  ... // Fill tile[] with zeros
  __syncthreads();

  int inIdx = ... // Calculate thread's input index
  if (__syncthreads_or(in[inIdx]>0.0f)) // Omit calculation if all zeroes
  {
    float kernel[rad+1];
    ... // Calculate kernel values from sigmas. Calls erf(): slow -> done once

    for (int i=0; i<2*rad+1; ++i) // Loops known at compile time -> unrolled
    {
      for (int j=0; j<2*rad+1; ++j) // Implicit sync. if tileX = blockDim.x
      {
        float *val = tile + (threadIdx.y+i)*(tileX+2*rad) + threadIdx.x + j;
        *val += in[inIdx] * kernel[abs(rad-i)] * kernel[abs(rad-j)];
      }
      __syncthreads(); // Care taken to leave this out of branching statement
    }
  }

  for (int row = threadIdx.y-rad+maxR; row < tileY+rad+maxR; row += tileY)
  {
    for (int col = threadIdx.x-rad+maxR; col < tileX+rad+maxR; col += tileX)
    {
      int outIdx = ... // Calculate thread's output index
      atomicAdd(out+outIdx, tile[(row+rad-maxR)*(tileX+2*rad)+col+rad-maxR]);
    }
  }
}
```

Figure 3. Code for the CUDA implementation of the kernel superposition.

Results

To evaluate the performance of the PBA, a test plan consisting of 20 energies covering a 10×10×10 cm^3 target in a water tank was adopted from the literature [2]. The only comparable GPU implementation reports a calculation time of 0.41 seconds for the kernel superposition of the highest energy alone on a GeForce GTX 480 (in turn 5 - 20 times faster than a CPU implementation) [2]. For a like-for-like comparison, the presented PBA was tuned to run on a Quadro 1000M of the same generation and clock frequency as the GeForce GTX 480 but with 5 times fewer cores. The total execution time including the ray tracing and kernel superposition for all 20 energies was found to be 1.38 seconds on this card. The significant increase in performance is attributed to the presented implementation of the kernel superposition, the reduced number of CPU/GPU memory transfers, and a pre-convolution step limiting the necessary kernel width in the superposition step. Preliminary results show that the complete dose calculation time for a typical head and neck treatment plan, including two fields of about 50 energies each, is around 1 second on a GeForce GTX 680 when using a dose resolution of 1×1×1 mm^3, which to our best knowledge is unprecedented in the literature.

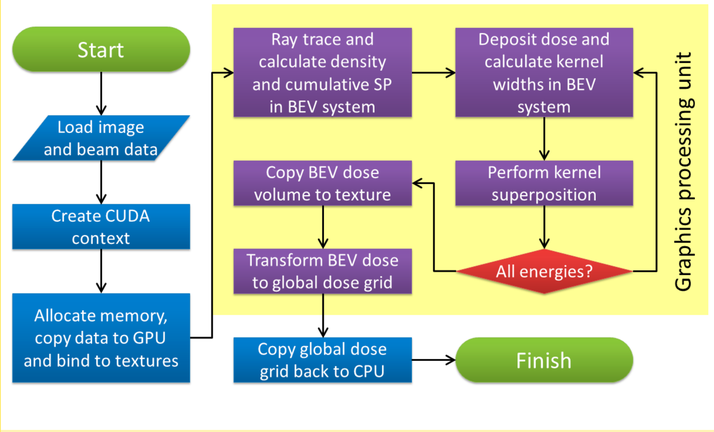

Figure 4. Flow chart over the CUDA implementation of the PBA.

Conclusion

We present a novel CUDA implementation of a 2D kernel superposition, which forms the backbone of a PBA for APT dose calculation. To the best of our knowledge, the complete implementation constitutes the first PBA for proton therapy to run entirely on GPU and it outperforms the partial GPU implementation found in the literature. Preliminary results for a real treatment plan suggest that near real-time dose calculation for APT could soon be possible on standard desktop computers.

Acknowledgements

We would like to thank Silvia Molinelli (CNAO, Italy) for providing clinical data and Till Böhlen (MedAustron, Austria) for help with MC simulations. ENTERVISION is a Marie Curie ITN funded by the European Commission FP7.

References

1. L. Hong, M. Goitein, M. Bucciolini, R. Comiskey, et al., Phys Med Biol, vol. 41, no. 8, pp. 1305–30, 1996.
2. R. Fujimoto, T. Kurihara, and Y. Nagamine, Phys Med Biol, vol. 56, no. 5, pp. 1319–28, 2011.