

Hybrid Volumetric Ray-Casting

Wei Hong

Feng Qiu

Arie Kaufman

Center for Visual Computing (CVC) and Department of Computer Science
Stony Brook University, Stony Brook, NY 11794-4400, USA
<http://www.cs.sunysb.edu/weihong/hybrid/>

Both CPU-based and GPU-based volumetric ray-casting algorithms have their advantages and limitations. The CPU can access very large memory, and programming on the CPU is much more flexible than that on the GPU. Meanwhile, the GPU is specially designed for local illumination computations, and it has several fragment processing units working in parallel on different input data elements in a SIMD manner. Thus, we devised a hybrid volumetric ray-casting algorithm using both the CPU and the GPU. We use the CPU for ray traversal and empty space skipping, and use the GPU for local illumination and ray integration. We also exploit the parallelism between the CPU and the GPU to obtain further acceleration.

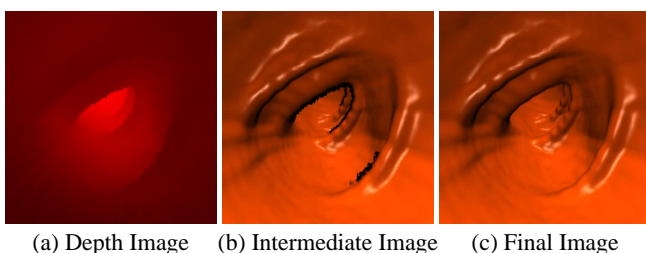


Figure 1: Overview of the hybrid volumetric ray-casting algorithm.

Our hybrid volumetric ray-casting algorithm consists of four steps: step 1 and 3 are implemented on the CPU, while step 2 and 4 are implemented on the GPU.

1. **Ray Determination:** At this step, the entry point and the normalized direction are computed for each ray. The entry point is defined as the first intersection point of the ray with the object. The entry points and normalized directions are stored in two 2D floating point textures (see Figure 1(a)).
2. **Slab Rendering:** We render a viewport size rectangle N times to make each ray travel N steps forward. We use a fragment program to implement the ray integration (see Figure 1(b)).
3. **Hole Predicting:** Since we only composit N sampling points along each ray during the slab rendering, there are some rays that have not saturated the opacity. Pixels corresponding to these rays are called *hole pixel*. We use the CPU to predict these hole pixels when the GPU is performing the slab rendering task. For each ray, we first check if the ray has already saturated in its first N sampling points, in which case the compositing result is already the final color. Otherwise, the corresponding pixel is a hole pixel on the intermediate image.
4. **Hole Filling:** During the prediction of the hole pixels, we generated a vertex array which stores the 2D image space coordinate and skip distance for each sampling point. We use OpenGL `GL_POINTS` command to do hole filling using the same fragment program as the slab rendering (see Figure 1(c)).

The key of our hybrid volumetric ray-casting algorithm is to exploit the parallelism of the CPU and the GPU. The GPU can only begin to work after the ray entry points and ray directions textures are loaded into video memory. In order to make the GPU start to work as early

as possible, the image plane is divided into small tiles. When the rays of a tile determined, the corresponding sub-textures are loaded into the GPU memory, the GPU start to do the ray integration for these rays immediately.

The trade-off between the CPU and the GPU is also crucial to our algorithm. We can adjust the workload of the CPU and the GPU by controlling how many steps are processed in the slab rendering and how many steps are processed in the fragment programs.

We implemented our algorithm on a 2.4GHz Intel Pentium IV PC with 1G RAM and a NVIDIA Geforce FX5950 graphics card. Figures 2(a) and (b) show two different images inside a human colon in the virtual colonoscopy system. The size of the CT dataset is 512^3 , and the image resolutions are both 256×256 . For the human colon dataset, we use the GPU to estimate the gradient on the fly, because its pre-computed gradients are too large to be loaded into GPU memory. Figures 2(c) and (d) show two different images of the engine dataset. The size of the engine dataset is $152 \times 256 \times 220$, and the image resolutions are both 512×512 . The opaque engine is rendered much faster than the semi-transparent engine, because the early ray termination is not effective when the semi-transparent transfer function is applied.

In conclusion, the hybrid volumetric ray-casting method can provide high quality and good performance for versatile direct volume rendering applications.

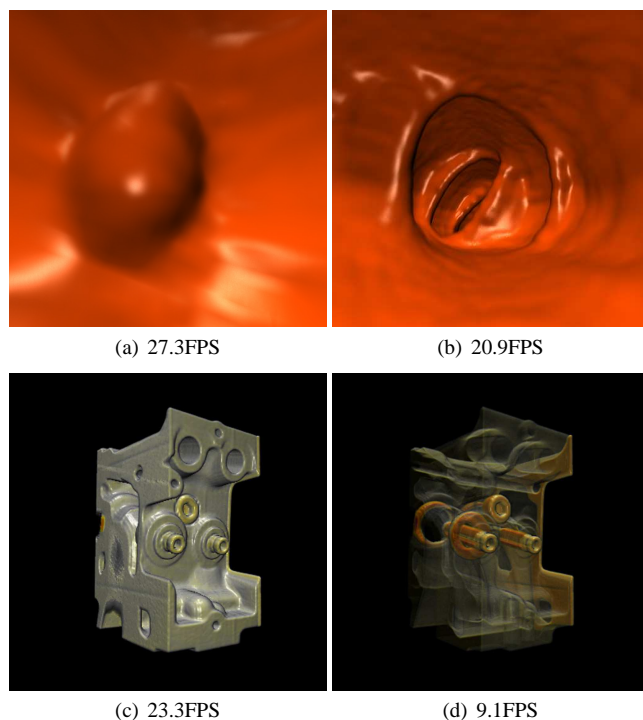


Figure 2: Average rendering frame rates per second for human colon and engine datasets.