Summary of *Realtime Ray Tracing*

Victor Shu

The paper summarizes several acceleration techniques to improve the performance of ray tracing applications to make them run at interactive frame rates. The most successful ones among all techniques can efficiently accelerate the most expensive parts of the ray tracing algorithm. The paper introduces several spatial acceleration structures and coherent ray tracing techniques, then it comes to parallelization and caching to utilize the hardware.

The most expensive parts of a ray tracing application are searching for ray-geometry intersections and computing the intersection points. The spatial acceleration structures can efficiently accelerate the process by restricting the potentially visible set for a given ray to a minimal number of primitives. The k-dimensional tree (KD-tree) splits the space with axis-aligned planes, resulting in the search being efficient and easy to implement. However, memory management is always an issue during the construction of the tree, and another issue is that KD-tree is not suitable for dynamic scenes. The bounding volume hierarchy (BVH) is a tree of bounding volumes where each one of the volumes stores child volumes that are inside this one, and eventually, the leaf nodes store primitives. Unlike space partitioning data structures like KD-tree, BVH allow overlapping bounding volumes, and there's no explicit representation of the empty spaces, thus making BVH suitable for dynamic scenes, but it also inefficient because the child volumes are not spatially ordered, so there's no way to early exit a search. The Bounding Interval Hierarchy is a combination of the two, which uses two parallel axis-aligned planes instead of a full bounding volume, and it also allows overlapping. That means it can have an early exit in a search, and suitable for dynamic scenes.

Coherent ray tracing is another technique that can accelerate the ray tracing process. Rays that have relatively close directions and origins are considered as coherent, and they can be traced within a packet. Tracing a packet of rays allows reusing the loaded data for multiple intersection tests and utilizes the cache, and it can be easily parallelized. The ray-packet-frustum is a more optimized technique that represents a packet of coherent rays as a frustum. Like view frustum culling in rasterization, it can reduce the potentially intersecting primitives by culling the scene against the frustum. The multi level ray tracing takes advantage of ray packet frustums by tiling the output image into blocks and packing all primary rays in the block into one frustum. Depending on the complexity of the block, the frustum either can be highly effective, or it will be divided into smaller ones. Thus, it can effectively budget rays if the scene has low complexity.

Ray tracing algorithms can be even more efficient when utilizing modern CPUs. By using SIMD extensions, a CPU can process several floating-point numbers in parallel. Also, it is effortless to use a distributed system to execute the algorithm parallelly like a networked PC cluster due to the scalability of the ray tracing algorithm.

There are arguments about if ray tracing is superior to rasterization for a long time, but with the structures and techniques mentioned in the paper and modern hardware, realtime ray tracing is becoming more and more possible. There are many advantages of ray tracing over rasterization, and when the hardware gets better, the dominant technologies will turn into ray tracing instead of rasterization.