Evaluation of the Multiplaten Z-Buffer Algorithm for Electromagnetic Ray Tracing in High-Frequency Electromagnetic Scattering Computations

Yong Zhou* and Hao Ling

Dept. of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712-1084 USA

The shooting and bouncing ray (SBR) technique, proposed in the 1980s, has become a standard approach for high-frequency radar signature prediction. In applying the SBR technique, the total computation time is comprised of the ray-tracing time and the time for carrying out the electromagnetic calculations. A number of techniques have been implemented in the mid-1990s to accelerate the latter. As a result, the total computation time of applying the SBR technique is currently dominated by the ray-tracing time. A faster ray tracer can significantly speed up the SBR algorithm for signature prediction. In this paper, we evaluate the multiplaten Z-buffer ray-tracing algorithm proposed by J.-L. Hu et al (*Elec. Lett.*, **33**, 825-826, 1997) as an alternative to the traditional Binary Space Partition (BSP) tree algorithm. Although it was claimed that the multiplaten Z-buffer algorithm should be superior to the traditional ray-tracing process, no evaluation about its time performance was reported.

In the standard BSP-tree algorithm, a BSP tree is first built based on the facet model of the target by recursively cutting the bounding box of the object along a spatial plane. Ray tracing is then performed by traversing the BSP tree. The BSP-tree based ray tracer is considered the fastest among all of the spatial subdivision approaches. In the multiplaten Z-buffer approach, a multi-layered Z-buffer is first generated from the scan conversion process. Instead of just storing the z-coordinates of the visible pixels as in the traditional Z-buffer process, multiple Z-buffers are created to store the z-coordinates of all of the facets within each pixel during the scan conversion. During the ray trace, a ray is tracked by moving along the ray direction pixel-by-pixel. Within every pixel, the z-depth of the ray is compared to all of the Z-buffer values for that pixel to check for possible intersections. Once an intersection is found, the hit point and the reflection direction can be calculated, and the tracing process is then iterated until the ray departs from the bounding box. We evaluate the computation time performance of the multiplaten Z-buffer ray tracer against that of the BSP tree-based algorithm. Results for a wide range of targets are tested to determine the computational as well as memory complexity as functions of the number of facets and the complexity of the target.