Using Hardware Ray Transforms to Accelerate Ray/Primitive Intersections for Long, Thin Primitive Types

I. Wald, N. Morrical, S. Zellmann, L. Ma, W. Usher, T. Huang, V. Pascucci

I. INTRODUCTION

We propose and investigate a technique that "tricks" the AABB-based BVH traversal hardware on Turing into first doing a hardware-accelerated oriented-bounding box (OBB) rejection test before switching to software execution and calling the expensive user intersection program. We demonstrate that for a variety of different real-world models this technique results in speedups of up to 5.9× over the (also hardware-accelerated) reference BVH, with otherwise identical intersection programs.

II. OBJECTIVES

- Different acceleration structures and traversal methods for ray tracing are still an active area of research; of particular interest to this paper is the Turing architecture.
- It is about accelerating the ray tracing of predominantly long and thin primitives such as cylinders, rounded cone stumps, curves, ribbons, etc.

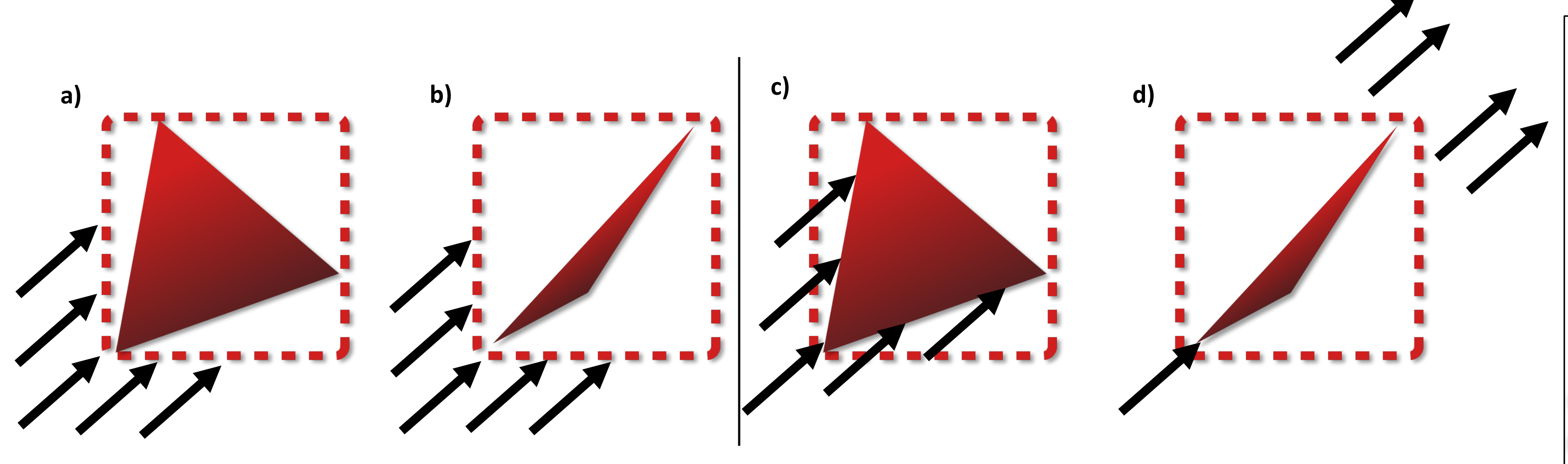

Fig. 1. Illustration of motivation.

- These primitive types all suffer from the same issue; namely, that the axis-aligned bounding boxes that modern ray tracers rely on often fail to tightly bound these shapes, leading to many costly ray-primitive intersections that mostly result in misses.

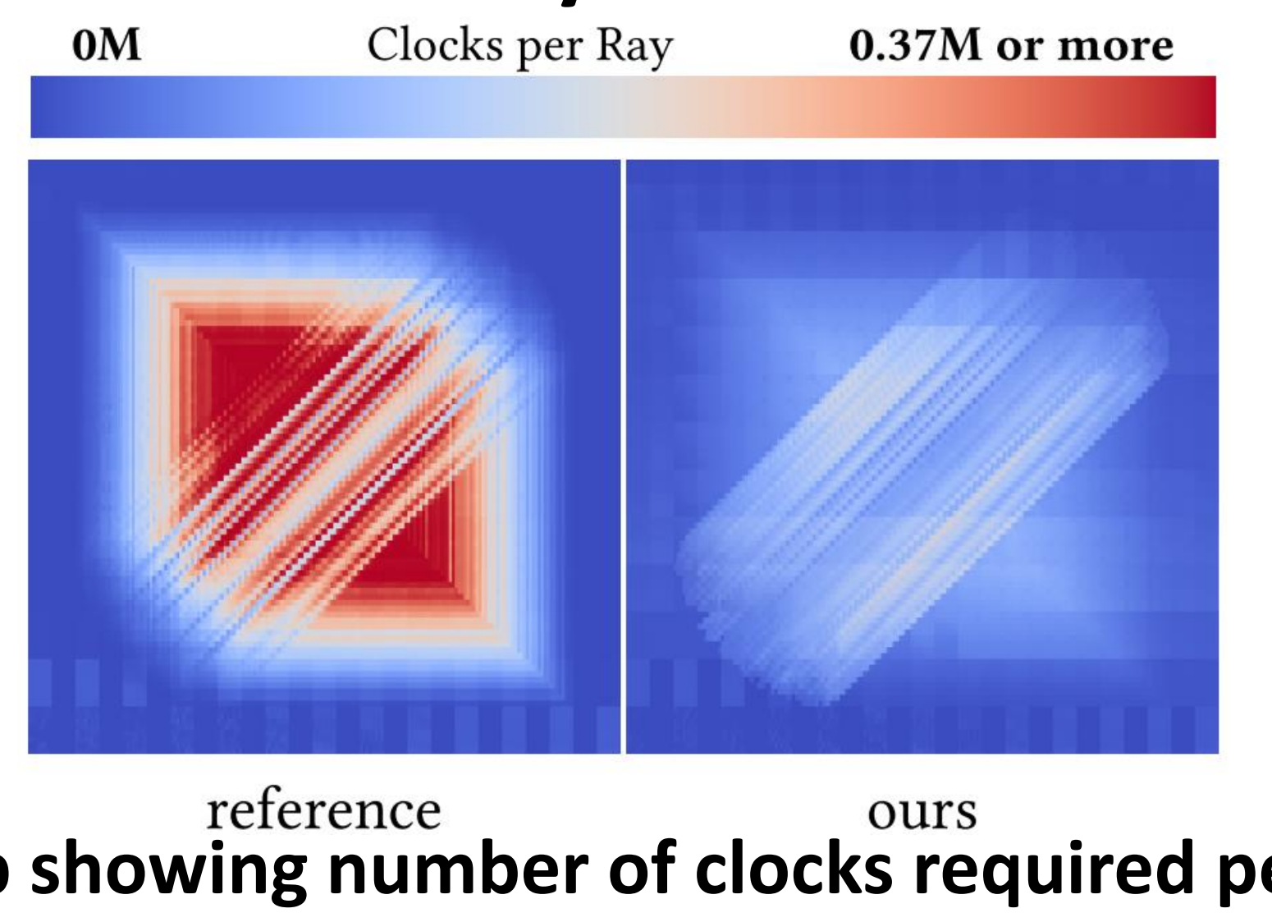

Fig. 2. Heat map showing number of clocks required per ray in a model.

III. REALIZING OBB TESTS

- The ray tracing hardware units on Turing can already do most of what a hardware OBB test would require.

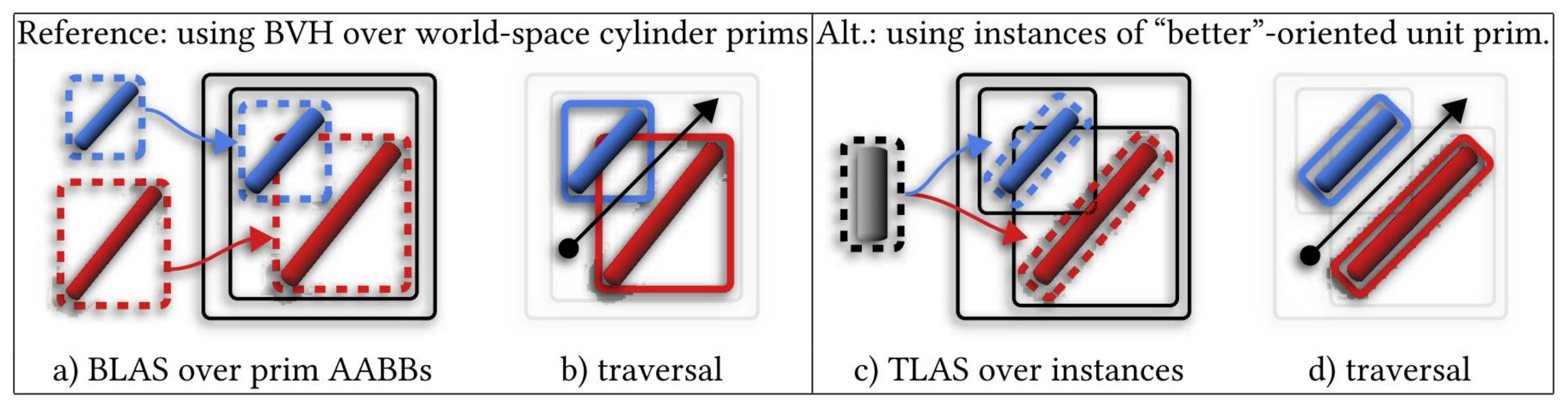
Reference: using BVH over world-space cylinder prims
Alt.: using instances of "better"-oriented unit prim.
a) BLAS over prim AABBs
b) traversal
c) TLAS over instances
d) traversal

Fig. 3. Conceptual illustration of using instance transforms to reduce number of primitive intersections.

- Replacing the primitives with instances of a "better oriented" unit primitive results in a top-level acceleration structure (TLAS) whose leaf nodes (black) look almost the same as in a), but the root nodes of the BLAS's they are instantiating (dotted red and blue rectangles in c) are now the equivalent of OBBs.

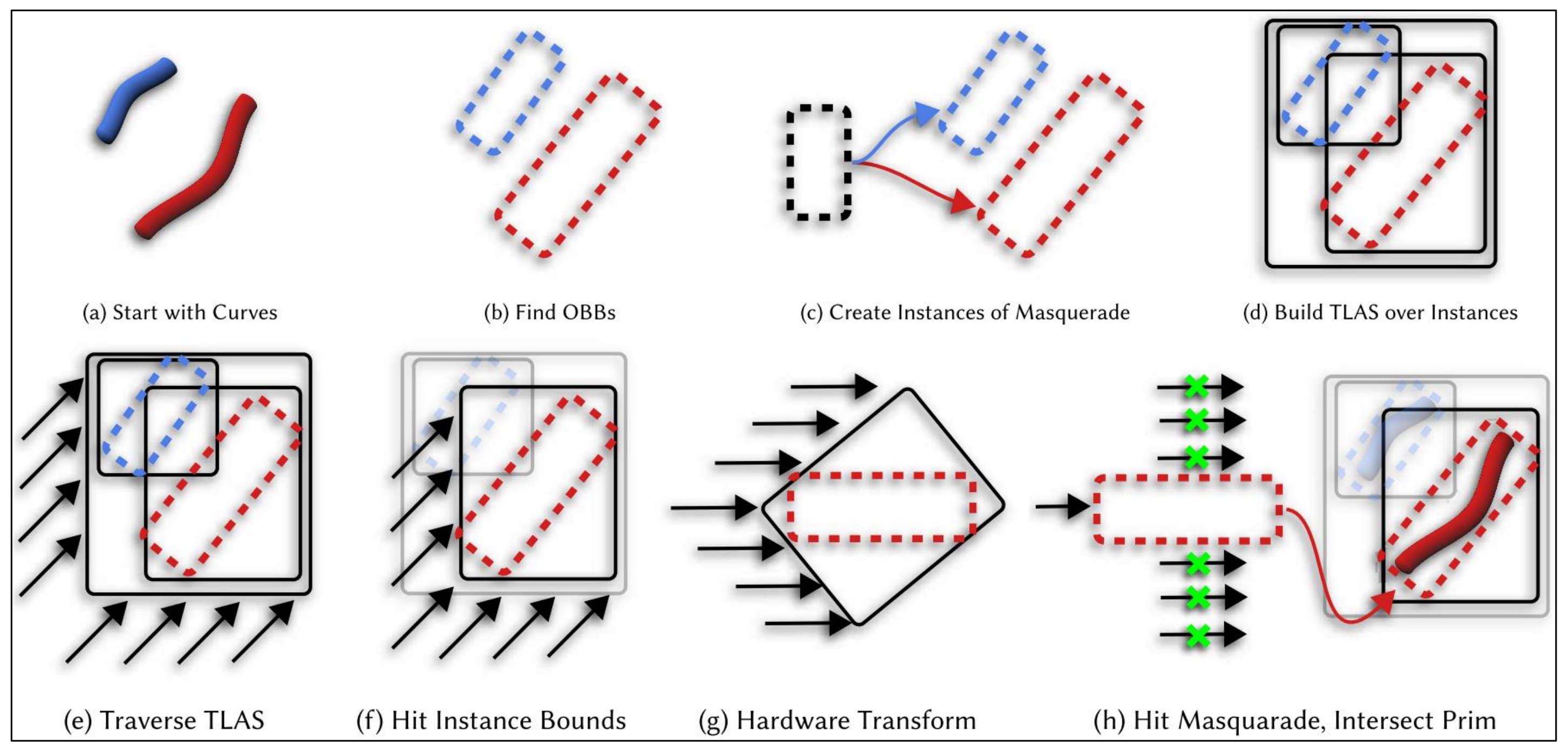
(a) Start with Curves
(b) Find OBBs
(c) Create Instances of Masquerade
(d) Build TLAS over Instances
(e) Traverse TLAS
(f) Hit Instance Bounds
(g) Hardware Transform
(h) Hit Masquarade, Intersect Prim

Fig. 4. Illustration of our complete model.

Two input requirements:

- for each of those primitives, an oriented bounding box(specified through an affine transform of a unit bounding box) that is guaranteed to cover the primitive in world space;
- a CUDA intersection program that, given an integer primitive ID, computes a (world-space) intersection with a ray and the specified primitive.

IV. PERFORMANCE EVALUATION

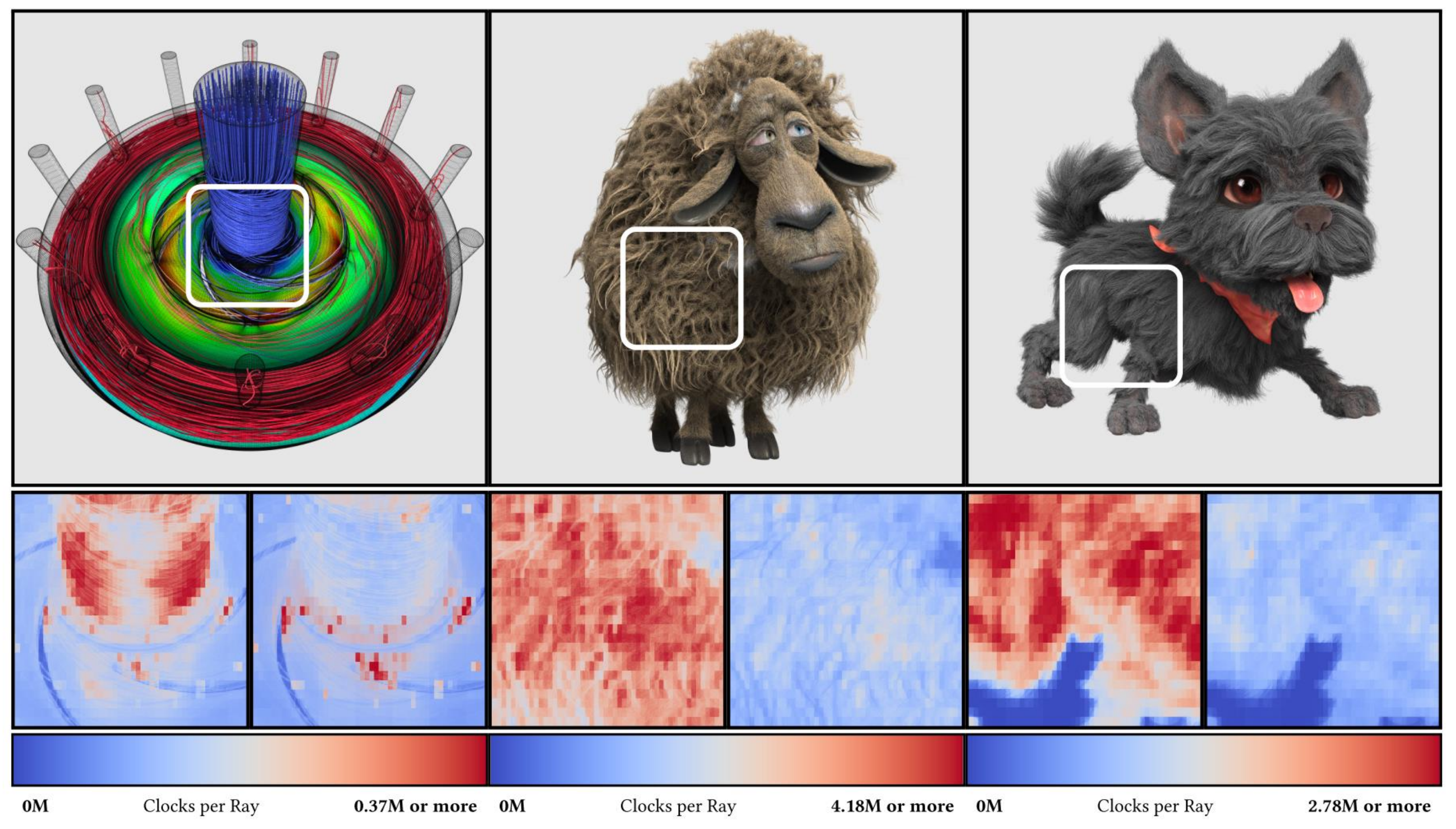

Fig. 5. Three of the models we used for evaluating our model. For these three models, our method leverages hardware ray transforms to realize a hardware-accelerated OBB culling test, achieving speedup of 1.3×, 2.0×, and 2.1×, respectively, over a traditional (but also hardware-accelerated) AABB-based BVH (both methods use the same primitive intersection codes). Bottom: heat map of number of intersection program evaluations for the two methods, respectively.

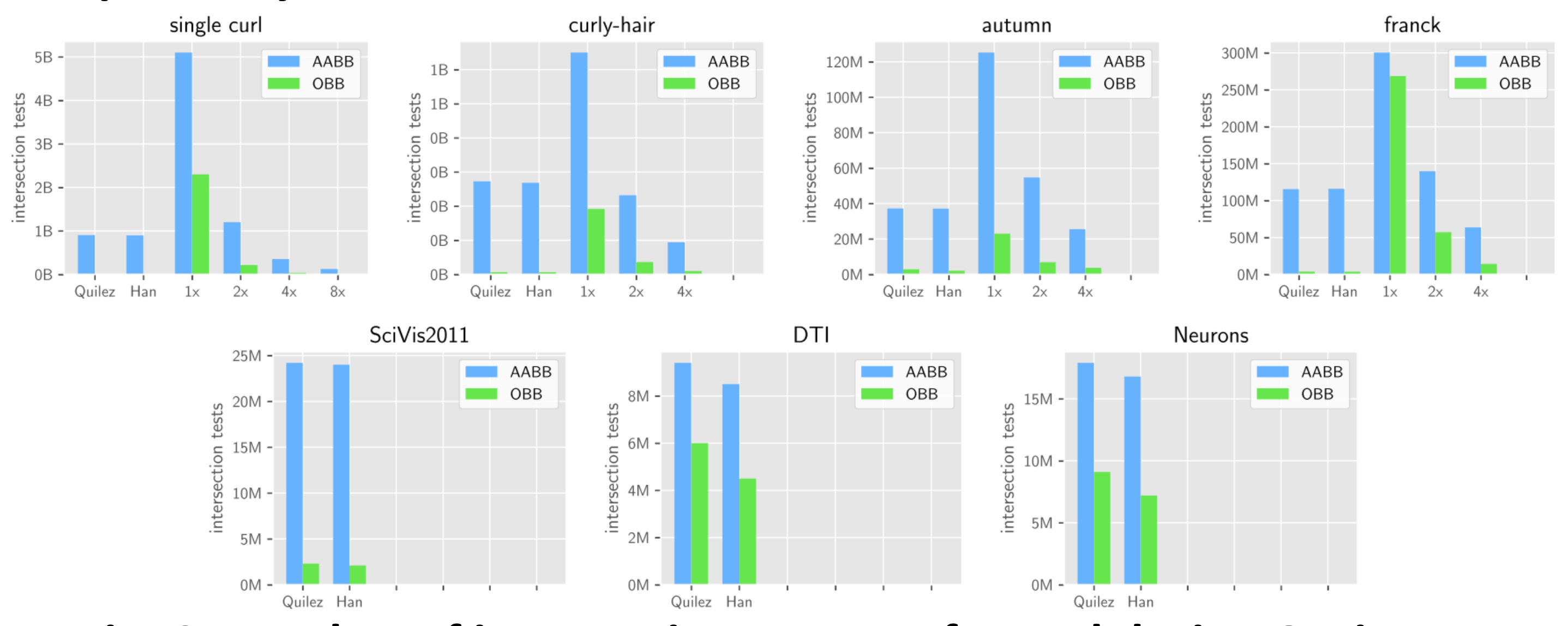

Fig. 6. Number of intersection tests performed during OptiX BVH traversal, both with the AABB-based reference implementation and with our OBB-based masquerading technique.

V. CONCLUSIONS

- We have presented a method that leverages existing ray tracing hardware units to realize an accelerated OBB culling test
- Our approach works by treating the root bounds of a BLAS as an OBB which we place into a TLAS through instancing
- Unlike traditional instancing, we do not create multiple affine copies of the same primitive

Environment: Ubuntu Linux 18.04.3, OptiX 7.0, CUDA 10.1

Hardware: Intel Xeon CPU (8 cores, 2.2 GHz), 128 GB of RAM, and two NVIDIA Quadro RTX 8000 GPUs