Accelerated Ray Tracing Report

I used C++ to implement my ray tracer. I ran it on a MacBook Pro with a 2.7 GHz Intel Core i5 processor. I used multi jittering in my algorithm to smooth out the edges. For that, I shot out 4 rays per pixel. Since my image size was 400x400, there were 640,000 rays shot per image. The color and position of each sphere was chosen randomly, and the size of the sphere was uniform based on the number of spheres.

Number of Spheres	Render Time in Seconds		Grid Speed Up Factors
	With BVH	Exhaustive	
10	1.991	2.164	1.09
100	2.303	6.052	2.63
1,000	3.593	41.401	11.52
10,000	7.126	442.157	62.05
100,000	166.920	3944.030	23.63

The BVH definitely sped up the render time considerably, but there was a drop in the speed up factor between 10,000 spheres and 100,000 spheres. I also noted down the time it took to construct the binary tree for the BVH structure in the table below and it showed that the time it took to create a 100,000 sphere tree was a lot more than it took for the 10,000 tree. I'm planning on modifying my algorithm and trying out different k values in the future to see if I can reduce the time.

Number of Spheres	Time to Construct BVH	Render Time after constructing BVH	Total Render Time
10	0.00014	1.991	1.991
100	0.000627	2.302	2.303
1,000	0.023158	3.569	3.593
10,000	1.59123	5.535	7.126
100,000	158.125	8.795	166.920