Optimising the Straylight Raytracer

Daniel Smith

March 20, 2014

1 Solution

The solution used in this optimisation is the utilisation of an R*-Tree. Generally R-Trees organizes the objects within a scene into a tree of axis-aligned bounding-boxes (AABB), where the children of each AABB are situated completely within the bounds of their parent's AABB, and the leaf nodes represent the bounds of each individual object in the scene. The R*-Tree, a specific type of R-Tree, tries to make these AABBs overlap each other as little as possible, thus minimising the amount of redundant intersection tests between each AABB. The aim is to reduce the number of intersection-tests done by removing all AABBs that do not intersect with each ray involved in the testing. R-Trees allow this to be done efficiently by pruning whole branches of the tree based on higher-level bounding boxes, removing many objects from the test with only a single test of their parent.

2 Experimental Setup

Experimentation was done by ray-tracing a variety of different scenes and measuring the time taken to complete the rendering of each scene. For the experiment, each scene was rendered twice: once using the original, unoptimised version of the program, and again using the new, optimised version. The time elapsed during each render was measured using the Unix "time" command. In order to generate a large enough sample size, the experiment was repeated several times. The scenes are divided up into "simple" scenes having less than five primitive objects, and "complex" scenes having more than five primitive objects. The experiment was carried out on a custom-built desktop with a 3.4GHz Core i5-3570, 8GB RAM and a 256GB Samsung 840 EVO SSD, running Ubuntu 13.10.

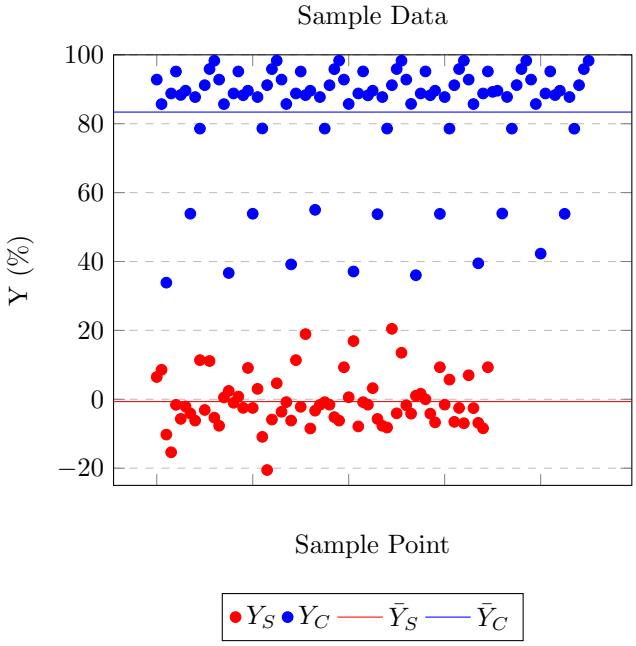
3 Analysis

The data is analysed to find out if the optimisation did actually bring about a significant increase in performance. In this case, a 10% increase in speed (i.e. a render time taking at least 10% less time) will be considered a significant performance improvement.

The experiment was executed seven times with ten simple scenes and thirteen complex scenes, yielding a total sample size of 161. The simple-scene sample

is denoted S, and the complex-scene sample is denoted C. Each sample point contains two variables, Y_1 and Y_2, the results gathered from the unoptimised and optimised versions of the program respectively. The random variable Y is defined as the percentage increase in speed from Y_1 to Y_2, and is calculated as follows:

$$Y = \frac{Y_1 - Y_2}{Y_1}$$



The hypothesis being tested is whether or not the mean speed increase due to optimisation is in fact greater than 10%. Thus the null and alternative hypotheses are defined as follows:

$$H_0 : \mu \leq 10\%$$
$$H_a : \mu > 10\%$$

A level of $\alpha = .01$ was chosen for the rejection region.

4 Results

The results of the hypothesis testing is presented in the following table.

Sample	S	C	$S \cup C$
Z	-11.624	40.960	10.622
H_0	Do not reject	Reject	Reject

5 Conclusion

The results show with almost no doubt that the optimisation resulted in a significant performance increase when rendering complex scenes, while also showing no performance increase when rendering simpler scenes. However, the test results do indicate that overall render performance was somewhat improved. Thus the conclusion is that the utilisation of an R*-Tree resulted in decidedly better performance than the original implementation.