Parallelizing Anisotropic Sampling
Mid Project Deliverable
CS 470 – Parallel & Distributed Systems
Charlie McCrea, Georgia Corey
Jason Zareski, and Sam Kiwus

# Discussion Paper

Overall, our project is coming along well. We've been able to create a reliable serial implementation of the isotropic sampling to use as a baseline test. This will give us something of a baseline with which to measure the performance of parallel (isotropic and anisotropic) sampling, as well as the skeleton for our serial anisotropic sampling. We'll be working on these programs in the next phase of the project and this serial implementation has given us a strong start. We were also able to create a python program to convert images into our own special gray scale image storage file type, `.grey`, as well as a testing script, multiple testing files, and basic instructions on how to install the necessary libraries and components. The main java program also outputs the results of each test as a `.html` file, so it can be viewed as an overlay of the original image in any html enabled web browser. This is very helpful for seeing how dense the distribution is and helped us calibrate our distribution method discussed below.

A potential issue that we've run across is determining when each simulation should stop running. Theoretically, it's possible to keep running the program until there isn't a single open space for the radius of a sampling ray (visualized as random number generator "throwing darts") to land successfully. That said, there is also the possibility that any random distribution could be more ordered, so running the tests more times with random distribution will result in discovering better and better coverage patterns. We eventually decided on having ten times as many failures to land as darts thrown, and this gives a reasonably dense population of sampling rays. Another problem we ran into was accidental early exit of the loop do to arithmetic overflow, which was easily solved once we noticed but previous test results had to be thrown out. The biggest obstacle that we faced was actually getting

the python ‘Pillow’ library to import on MacOSX, but once we ironed out the steps for success we included them in the readme file.

For inclusion in our final project, we have a wonderful way to display the sampling rays via the `.html` files. We’ll likely include the sampling results of some simple, geometric reference images and the associated runtime data, as well as some larger and more complex photorealistic images that take significantly longer to sample. Obviously, we will compare the performance between all there or four types of sampling that we run, dependent on how many we get working. On the poster, we’ll also include a description of our project, what we were attempting to achieve, what the results tell us and how we managed to make it happen. We will also discuss obstacles we ran into as a team and how we overcame them. It would also be a good idea to use some of the poster to explain what anisotropic sampling is, so the layperson looking at it understands the real-world application of the methods used. It’s important to emphasize real-world connections in research for undergrad CS students as well to give them a glimpse of their future. Of course, the poster will include good graphic design and information display principles and should include the professional headshots of the team so that viewers can put names to faces.

According to our current progress, we shouldn’t face many issues moving forward. We appear to be well on our way to having a completely finished project by the final due date. We expect to complete all the types of sampling that we attempt to produce, with both parallel versions scheduled to be completed in the next steps of the project. After meeting with Dr. Bowers and using his expertise as a resource and inspiration moving forward, it’s clear that fully exploring this research topic should be well within the ability of our group and we expect to deliver an interesting and thoroughly researched report for display in the CS department.

Some testing results are included below. Test 5 was our largest test at this phase of the project. After making the corrections to the arithmetic overflow, instead of being a one-to-one ratio of darts thrown to dart failures, the numbers bumped up to one-to-ten as intended. However, this also meant the sampling takes significantly longer. These corrections resulted in about twice the density of sampling, but about twice to four times the sampling runtime. In the last case, runtime was multiplied by a factor of 77. The usual increase rounded out to about 70 times longer for twice the sampling coverage on the largest image.

**Before corrections to sampling overflow:**

```
=== TEST 5: Earth Image ===
WxH = 8192 4096
Done!
Number of conflicts: 57321
Creating SVG file...
Using grey file: earth.grey and picture file: earth.jpg
Number of darts: 57320
Runtime: 6321 ms
Earth image file sampled
```

**After corrections to sampling overflow:**

```
=== TEST 1: Puppy greyscale file ===
WxH = 332 300
Done!
Number of misses: 3450
Creating SVG file...
Using grey file: puppy.grey and picture file: puppy.jpg
Number of darts: 345
Runtime: 35 ms
Puppy file sampled

=== TEST 2: Gradient greyscale file ===
WxH = 116 116
Done!
Number of misses: 500
Creating SVG file...
Using grey file: gradient.grey and picture file:
gradient.png
Number of darts: 50
Runtime: 14 ms
Gradient file sampled

=== TEST 3: Desaturated Bars file ===
WxH = 900 600
Done!
```

```
Number of misses: 17710
Creating SVG file...
Using grey file: desaturatedBars.grey and picture file:
desaturatedBars.png
Number of darts: 1771
Runtime: 105 ms
Desaturated bars file sampled


=== TEST 4: Color References file ===
WxH = 421 220
Done!
Number of misses: 3170
Creating SVG file...
Using grey file: ColorValueRef.grey and picture file:
ColorValueRef.png
Number of darts: 317
Runtime: 29 ms
color ref file sampled


=== TEST 5: Earth Image ===
WxH = 8192 4096
Done!
Number of misses: 1079860
Creating SVG file...
Using grey file: earth.grey and picture file: earth.jpg
Number of darts: 107986
Runtime: 491006 ms
Earth image file sampled
```