Single Image Haze Removal

Rama Mannava (rmannava)

Problem

Haze removal is the process of removing the effects of fog, smoke, and other atmospheric phenomena from an image. Many methods require the use of multiple images to provide the necessary information, but there exist strategies such as the one outlined in this paper that make use of a stronger assumption to supplement the under-constrained problem. A dark channel prior can be used to estimate the effect that haze has on light transmission, in addition to the atmospheric light in the scene. This information can then be used to reconstruct the original dehazed scene.

Solution Approach

The algorithm consists of several stages that must be performed sequentially. Each stage, however, is performed on individual pixels using context from a patch of surrounding pixels. The pixels of the final image are all constructed independently, so the image can be partitioned into blocks and computation can proceed for the entire image in parallel.

The use of CUDA will greatly improve the performance of this algorithm. The GPU on the GHC machines can support `46` blocks with `1024` threads each, which translates to `47,000` pixels that can be handled at once. If the entire algorithm consisted of calculations on pixels, this would lead to a `47,000x` speedup over a single threaded CPU-based implementation. However, the algorithm has sections restricted to sequential computation, and time spent computing pixel values will likely be negligible next to the bottleneck caused by transferring data around. Therefore, the actual speedup of a CUDA implementation will be much lower than `47,000x`. Although a precise estimate would require testing the separate sections of the algorithm, a speedup of `100x` should be achievable with judicious use of kernel launches and data transfer between host and device. Speedup will be measured on a set of images chosen with differing backgrounds and levels of haze.

Timeline

Week 1: Write and test sequential implementation

Week 2: Begin CUDA implementation

Week 3: Finish CUDA implementation and begin testing

Week 4: Finish testing CUDA implementation and begin performance tuning

Week 5: Finish performance tuning and begin comparison testing

Week 6: Finish comparison testing and write final report