15-418 Project Proposal: Analysis of Reduce Sweep Image Segmentation

William Qian, Joseph Gnehm

October 31, 2019

1 Disclaimer

This project proposal has two separate proposals as we are currently in the middle of deciding which topic to pursue. We have written the proposal below for the image segmentation algorithm, but we also have an idea to collaborate with a robotics lab on campus that is working on parallel code for generative sensor design. We still have to look into the second idea more to see how feasible it is, but please continue reading for more details.

2 Project URL

The url that will keep track of the project is:

https://geejoseph.github.io/qian-gnehm/

3 Summary

This project will focus on the implementation and analysis of the Reduce Sweep strategy for graph based image segmentation. We found a paper which implemented this same algorithm in four different ways, but was not able to achieve much speedup. The authors left some ideas for improving on the algorithm or approaching it in a different way. We will implement a CUDA and OpenMP version of the algorithm, determine potential bottlenecks in the code, and attempt to different optimization techniques to create a competitive image segmentation algorithm.

4 Background

Graph based algorithms have recently become an interesting area of research for image segmentation, due to the potential for large parallel speedup gains. Many of the current graph algorithms reduce the problem to an MST or K-means problem. In a relatively recent paper, the authors describe a novel algorithm called the reduce sweep method which seems to be more easily parallelizable. At a high level, the algorithm partitions the image into chunks, in which the algorithm is recursively called on. These chunks are then merged based on some criterion (i.e. color). The authors provide different implementations of the algorithm exists along with some empirical testing, showing that even though it seemed promising they were able to get relatively little speedup. It seems possible that the algorithm can be more specifically tuned to the different architectures (CUDA, OpenMP) as well as to the particular images that are going to be segmented.

5 Challenges

Some challenges that we anticipate is the relative amount of synchronization that appears to occur with the algorithm. We will experiment with ways to reduce amount of synchronizations required and minimize communication to computation ratio.

The authors also note that the algorithm does a lot of expensive random memory accesses and it may be promising to

> *modify the Hybrid implementation to keep the data in the CPU's memory (zero-copy), eliminating the memory allocation and copy overheads. Another possibility would be to devise new sweep orders that mesh better with the GPU architecture, or new merge criteria that aren't based on color.*

Finally, depending on the image, there can be a very heavy work imbalance for this algorithm. This means that using different scheduling heuristics and granularities could help to increase speedup.

6 Resources

We will be implementing everything from scratch, including the sequential implementation. We will start be using "Parallel Image Segmentation Using Reduction-Sweeps On Multicore Processors and GPUs" as this paper inspired our project, to be the initial guide in our implementation, before extending it.

Farias, Ricardo & Marroquim, Ricardo & Clua, Esteban. (2013). Parallel Image Segmentation Using Reduction-Sweeps on Multicore Processors and GPUs. Brazilian Symposium of Computer Graphic and Image Processing. 139-146. 10.1109/SIBGRAPI.2013.28.

7 Goals and Deliverables

At the very minimum, we plan to have both the CUDA and OpenMP implementations running, having tried a couple different ways of implementing each algorithm for both CUDA and OpenMP. We aim to clearly identify areas where speedup is most obvious and bottlenecks, as well as areas for potential improvement.

We hope to ultimately achieve performance that is better than what was given in the reference paper, at least on a subset of the images. We also may experiment with extending the model to 3D image segmentation.

For the demo specifically, we will display the results of our image segmentation algorithm, along with speedup graphs, workload graph, and any another visuals that we construct during our analysis.

8 Platform

We will use the GHC cluster, as the GHC cluster supports CUDA and OpenMP development.

9 Schedule

Week 1: Continue researching topic, update and finalize proposal
Week 2: Finish sequential implementation, start parallel implementation
Week 3: Complete Parallel Implementation, and start optimization
Week 4: Finish checkpoint, Continue with optimizations
Week 5: Complete optimizations and analysis and begin writing report
Week 6: Finish report, and prepare for demo

15-418 Alternative Project Proposal: Generative Sensor Design

William Qian, Joseph Gnehm

October 31, 2019

1 Disclaimer

This is the second (and more tentative) of two proposals which we are still working on.

2 Project URL

The url that will keep track of the project is:

https://geejoseph.github.io/qian-gnehm/

3 Summary

This project would help a lab that designs soft force sensors. It would parallelize the exploration of different designs, including where to put Hall-effect sensors in the rigid part of the robot and magnets in the foam part to be able to sense forces and torques on the foam as accurately as possible.

4 Background

This is part of a project in Lu Li's team in the Biorobotics Lab. They are trying to design robots that can help with recycling. In particular, this project is for a robot that can take the caps off of plastic bottles and sense what kind of plastic an object is made out of and where to grip it/cut it by squishing it.

The lab currently has prototypes of these foam sensors attached to a rigid moving part. The foam can be in all different shapes and sizes, which would be helpful to work with different parts of the waste stream (like big pieces of cardboard vs. small electronic parts). Under the foam part is at least one magnet. Then on the rigid part are Hall-effect sensors which detect part of the magnetic field. As the foam is perturbed, the magnet moves, and the Hall-effect sensors can pick it up. From this information we can infer what forces (and potentially torques) are hitting the foam.

But then the question becomes, where is the best place to put the magnets and Hall-effect sensors? Their range is limited and it is important to be accurate, they can overlap each other to provide more accuracy or spread out to get more coverage. Right now in the lab, there is a prototype with a simple and symmetric configuration as a proof of concept, which seems reasonably placed. But when the foam pieces are more irregular or we can use more magnets and sensors it is not as obvious.

So, the lab is looking at doing a generative design of this sensor apparatus, exploring lots of ways the pieces could be arranged. This can be done at various levels of sophistication - it is possible to do very realistic physics simulations for a configuration, but it is expensive. So the question is, how can you do this efficiently?

The basic idea is to explore this space of configurations by just chopping it up. But we can be more clever, by first placing magnets, then sensors, or one sensor at a time - and ruling out whole branches of bad configurations. We can also be clever by doing the simulations in parallel for similar configurations. We can be even more clever by using a kind of "granularity" in the simulations themselves - choosing for promising branches to do very realistic simulations, and for not-so-promising branches only a basic simulation that does not cost too much.

5 Challenges

Unknowns - we do not even know how long the sequential version takes at this point, is it really that bad?
Scope - what can we really help with?
Range of sensors - how do they impact each other?
Simulation fidelity - how easy is it to implement a realistic simulation?
Testing and calibration - do we need to physically try some of these configurations to learn something?

6 Resources

Primer on GPU in Matlab: `https://www.mathworks.com/solutions/gpu-computing.html`.

Here are two recent papers trying to reach a similar goal:

Geometry Optimisation of a Hall-Effect-Based Soft Fingertip for Estimating Orientation of Thin Rectangular Objects

An Adjustable Force Sensitive Sensor with an Electromagnet for a Soft, Distributed, Digital 3-axis Skin Sensor

Lu Li and Evan Schindewolf in the lab have experience with parallelism and hardware and would be willing to help us.

7 Goals and Deliverables

The most basic goal would just be to get a simple parallel version working that would help the lab. Then we could do an analysis of what sources of parallelism there might be and go from there.

Hopefully, we could take advantage of our knowledge about the sensors (and intuitive ideas like symmetry) to get much more advanced parallelism.

We can compare all of these attempts against a sequential version, which we can run until completion.

8 Platform

The current code (and most of the code in the lab) is in Matlab, so that's what we have to work with. Matlab supports both CUDA and some parallel for loops, along wiht traditional threads.

9 Schedule

Week 1: Meet with PI of lab and also with Profs. Mowry or Railing to see how feasible this is, decide what to do!
Week 2: If taking this route, finalize goals and expectations, read existing code and pseudocode, get it up and running
Week 3: Complete sequential implementation to be used as a check, begin basic parallel implementation
Week 4: Complete basic parallel implementation and basic analysis and move on to more advanced ideas
Week 5: Finish out advanced parallelizations, test, begin writing report
Week 6: Finish report, and prepare for demo