Parallel Web Based Lesion Border Detection for Dermoscopy with Heterogeneous Computing

Participants: James Lemon, Matthew Reigada, Benjamin J. Kruger
Mentors: Drs. Sinan Kockara and Tansel Halic

Background

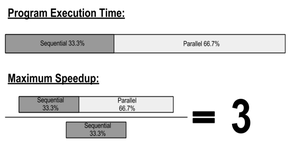
Melanoma is a swift killer. However, early detection often leads to successful treatment through excision. The most effective means of detection, short of biopsy, is dermoscopy. This involves using a tool called a dermatoscope to image the suspect lesion in high resolution.

Melanoma Diagnosis

Rule	Score	Weight
Asymmetry	0-2	1.3
Border	0-8	0.1
Color	1-6	0.5
Diameter	1-5	0.5

The ABCD rule is the most common technique for naked eye examination of a skin lesion to determine if it is melanocytic. The border is used for 3 portions of the diagnosis. An automated border detection process would remove some subjectivity in Melanoma diagnosis.

Parallel Processing



Amdahl's law states the maximum speedup achievable from parallel processing. It assumes that all tasks that can be done in parallel are done in parallel. In reality we are limited by the number of processors we can use. In GPGPU programming, the number of processors is in the hundred, and provides a very high level of parallelism

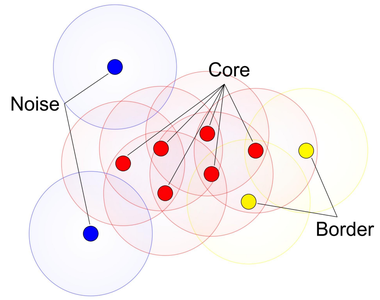
Web Interface

Thanks to WebCL, the parallel processing can take place within a browser. Because most people are fairly comfortable interacting with the Web, user training is simplified. Concerns over upgrades and version numbers, as well as compatibility are gone.

Methodology

DBSCAN is a density based clustering algorithm requiring two arguments: epsilon (ε) and minimum number of points (MinPts). DBSCAN works by initiating or expanding a cluster around a point only if its ε-neighborhood contains the minimum number of points. Such a point is considered "core". Points within a core point's ε-radius whose ε-neighborhoods do not meet MinPts are labeled "border". Points which meet neither of the previous criteria are deemed to be "noise".

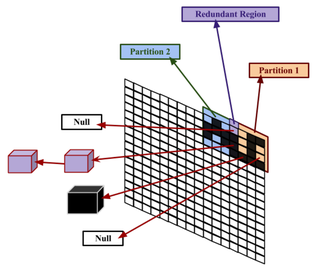
To allow the algorithm to run in a reasonable amount of time for high resolution images, we implemented it in WebCL, an emerging technology which allows for OpenCL kernels to be loaded and executed on a Web browser.



Novel Algorithms

Partition Merging

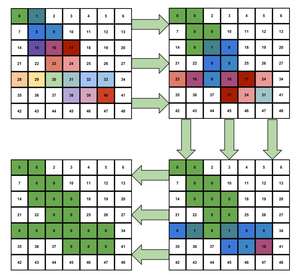
WebCL has an upper bound limit for the amount of data that may be bused to the GPU for processing. Therefore we partition our image into chunks which may be bused to the GPU, processed and merged into the global image state .



The above diagram depicts the global state of the image. Each pixel corresponds to a linked list (of sizes 0, 1, or 2). Linked list size 0 means the pixel is not in a cluster. Linked List size 1 means the pixel is in a cluster and not in conflict. Linked list size 2 means that the pixel is in 2 separate clusters in 2 separate partitions, and by the transitive property of clusters, these clusters should be merged across the partition boundaries.

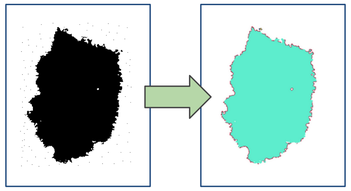
Reduce Root

During parallel processing each pixel contains its own node ID as its root. The lowest node ID is broadcast to all pixels within epsilon range until no roots are percolated further out within the image.

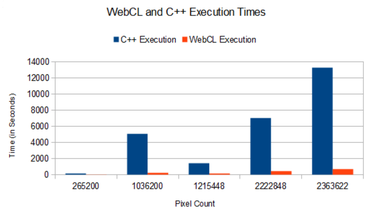


The above diagram demonstrates the operation of the reduce root algorithm for epsilon = 1. Each pixel initially contains its own ID as its root, and rewrites its root as lower roots are broadcast via the reduce root algorithm. For the example diagram epsilon = 1, and the image is shown after 0, 3, 6, & 10 iterations of reduce root.

Results



A printable image of a lesion is created. Each identified cluster is colored and their borders are highlighted. The implementation is loaded on a Web browser. The OpenCL code is compiled from source as needed. The input is collected using Javascript. The output is drawn onto an HTML5 canvas.



Execution time is not entirely dependent on image size. The number of pixels being consumed by a cluster and the number of clusters found also play a role in execution time. These values can be approximated by image size.

Future Work

The current algorithm works on Binary Images. This algorithm can be modified to consider colors in the clusters by adding additional dimensions or modifying the way distance is calculated.

Noisey pictures return multiple clusters besides the skin lesion. More clear output can be created by detecting and returning only the largest cluster found by DBSCAN.

Conclusion

We were able to deliver efficient code over the Web browser that performs complex algorithms faster than serial implementation. WebCL is an enabling technology that allows for computationally expensive tasks to be integrated into a Web browser. DBSCAN is a parallelizable algorithm. The speedups available to DBSCAN from parallelization can make it better suited for applications with large datasets or that require quick computation. Our implementation provides fast results, quality output, and can be easily delivered over a network.

Contact Information

Ben Kruger, Undergraduate Researcher: kruger@nsuok.edu
James Lemon, Graduate Researcher: jlemon2@uca.edu
Matt Reigada, Undergraduate Researcher: mreigad1@binghamton.edu
Tansel Halic, Faculty Mentor: tanselh@uca.edu
Sinan Kockara, Faculty Mentor: skockara@uca.edu

Acknowledgments

This research was partially supported by NSF Award# 1062838: "REU Site: HIT@UCA: Applied Research in Health Information Technology."