doi:10.1017/S1431927616004797
Microsc. Microanal. 22 (Suppl 3), 2016

Three Dimensional Parallel Automated Segmentation of Neural Soma in Large KESM Images of Brain Tissue

Laila Saadatifard[1], David Mayerich[1]

[1] Department of Electrical and Computer Engineering, University of Houston, Houston, USA.

Recent advances in high-throughput imaging allow the researchers to acquire terabyte-scale volumes of image data describing the three-dimensional structure of tissue. Many analyses are currently bottlenecked by the time required to process such large data sets. Fully automated techniques are required in order to realistically process large volumetric data sets. In addition, the size of the data introduces additional constraints, including the need to process data with minimum user feedback.

In this paper, we describe an automated method for finding the position of cell soma in images stained using thionin (Nissl) and imaged using Knife-Edge Scanning Microscopy (KESM) [1]. This method can be used to acquire data on the order of terabytes per day. As the speed of imaging techniques improves, faster segmentation algorithms are needed. Furthermore, the large variability of cell types and cell densities make an accurate automated cell segmentation difficult. In this work, an iterative voting technique is used to generate a binary segmentation of the volumetric data, which is then refined to determine the location of cell soma [2]. This technique requires minimal user input and no user feedback, making it amenable to fully automated processing of terabyte-scale data sets. We demonstrate the effectiveness of this algorithm on volumetric images of the rat somatosensory cortex obtained using KESM by comparing to a manually segmented ground truth. The proposed algorithm is parallelized in order to provide efficient evaluation using heterogeneous (GPU-based) hardware commonly available in commercial computer systems. The high level of parallelism built in to our algorithm also means that the algorithm is scalable, allowing the allocation of additional computational resources to improve segmentation speed.

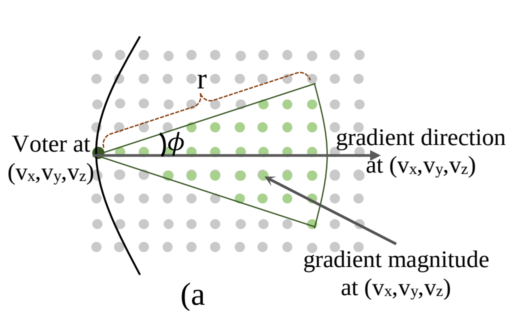
Iterative voting is based on radial symmetry, so it is able to detect cells with different sizes and shapes. Every pixel of the dataset generates a conical *voting field* based on the pixel gradient magnitude and direction (Fig. 1). All voting fields are integrated to produce a *vote image*, where the magnitude of each pixel reflects the probability of a cell centroid at that position. This vote image is then used to refine the direction of each voting field. The vote image is then iteratively refined and local maxima of high intensity values in the vote image are calculated and labeled as cell centroids. The details for the two-dimensional algorithm are described in the work by Pravin et al. [2]. We extend the iterative voting algorithm to three dimensional data sets and develop a GPU-based algorithm to calculate the vote image and update the voting fields in a practical time frame. For validation, a 128×128×128 chunk of KESM data was selected, and the precision and recall of features are calculated. Figure 2a shows the precision-recall curves after 1, 8, and 30 iterations of voting.

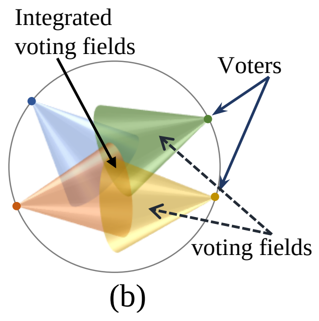
The iterative voting algorithm includes a significant amount of computation, making it generally impractical for 3D data sets using a single thread. However, these calculations can be highly parallelized by (a) computing the vote image for each pixel independently and (b) updating the voting field for each pixel independently. Both of these calculations can be represented as a 3D template function. However, the significant number of data fetches creates a memory latency bottleneck when GPU global memory is utilized. Our algorithm takes advantage of on-chip shared memory available to GPU streaming multiprocessors to significantly reduce memory latency for redundant fetches. The profiling result shows a much better performance by parallelizing the code and using shared memory. Figure 2b shows the performance for running our algorithm with the same parameters on the CPU (using a Matlab implementation) and GPU.

Conclusion: Validation results demonstrate that our algorithm works well for detecting cell nuclei in volumetric data sets that include cells with varying sizes and shapes. Iterative voting has also been shown to be robust in images where cells are densely packed, which is often seen in KESM brain data. By parallelizing the iterative voting algorithm and running on the GPU, performance significantly improves, allowing further refinement of the vote image and making the algorithm practical for 3D data sets. This algorithm requires two input parameters, but executes on entire data sets with no user interaction.

References:

[1] D Mayerich, L Abbot and B McCromick, Journal of Microscopy Volume **231** (2008), p. 134.
[2] B Parvin *et al*, IEEE transactions on image processing (2007) p.615.


r
ϕ
Voter at (vx,vy,vz)
gradient direction at (vx,vy,vz)
gradient magnitude at (vx,vy,vz)
(a

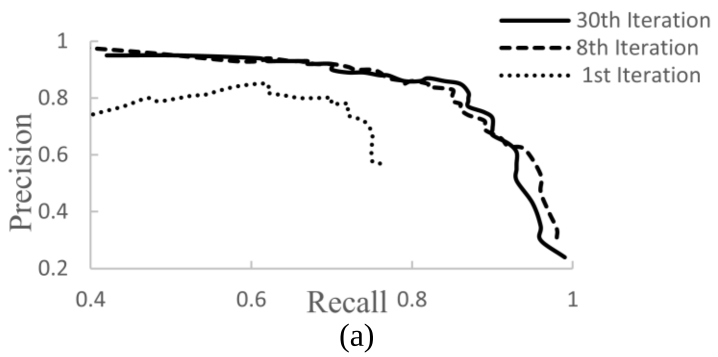


$$V(p) = \sum_i M(v_i)$$
(1)

$$v_i \in \{\angle(p - v_i) < \phi \text{ and } |p - v_i| < r\}$$
(2)

(c)

Figure 1. (a) The voting field for each voter is based on the image gradient. (b) Voting fields are integrated to create the vote image. (c) Equation 1 shows the integrated vote image value for a pixel at pixel position $p = (p_x, p_y, p_z)$, where M is the gradient magnitude for the voter at position $v_i = (v_{xi}, v_{yi}, v_{zi})$. Equation 2 defines the relevant voters that contribute to p in the vote image. These voters contain p within their voting fields, where r and ϕ define the voting field angle and distance.



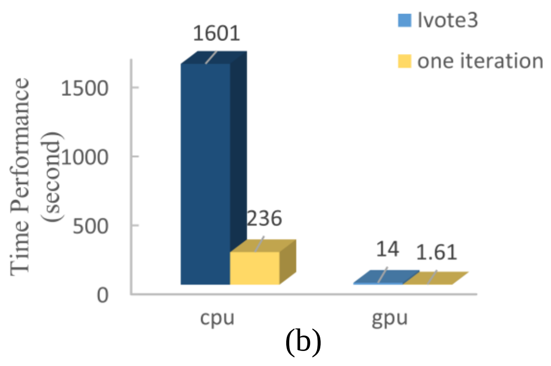


Figure 2. Precision-Recall curve after different iterations (a), time performance for running our algorithm sequentially (CPU, Matlab) and parallel (GPU, CUDA) for 1 iteration (yellow) and until convergence at 8 iterations (blue).