# A Parallel Annealing Method For Automatic Color Cervigram Image Segmentation

Edward Kim, Wei Wang, Hongsheng Li, Xiaolei Huang

Image Data Emulation and Analysis Laboratory
Lehigh University

**Abstract**

The accurate and automatic segmentation of tissue regions in cervigram images can aid in the identification and classification of precancerous regions. We implement and analyze four GPU (Graphics Processing Unit) based clustering algorithms: k-means, mean shift, deterministic annealing, and spatially coherent deterministic annealing. From our results, we propose a novel parallel algorithm using the CUDA programming language for digital cervigram segmentation and clustering. The first step of our fully automatic method is to compute the number of modes in the feature space of a color cervigram image using the mean shift clustering algorithm. Next, we use the number of modes in a novel spatially coherent deterministic annealing optimization technique to produce an approximate optimal solution for the clustering problem. Our GPU based methods perform approximately 38x (deterministic annealing), 134x (mean shift), and 276x (spatially coherent deterministic annealing) faster than an equivalent CPU solution. Our implementation decreases the computational time of an annealing method on a 1280x872 pixel image from 5 hours 3 minutes to 72.12 seconds, enabling the use of this optimization method in clinical settings and on large cervigram datasets.

## Contents

## 1 Introduction

Clustering is an important technique in image segmentation. In consideration of an archive of 60,000 color uterine cervix images created by the National Library of Medicine (NLM) and the National Cancer Institute (NCI), we require a fast, accurate, and automatic segmentation method to analyze these images. Specifically in the area of cervigram image analysis, the most important observed area to segment is the Acetowhite (AW) region, which is caused by the whitening of potentially malignant regions of the cervix epithelium. The segmentation of this area and other tissue regions is particularly challenging due to high variability where tissue color distributions frequently overlap with different classes [16, 7]. Segmentation using simple clustering algorithms like K-means and fuzzy c-means [9], run quickly but are highly sensitive to initialization. Mean shift [4] is a more advanced algorithm able to detect the number of modes in an image, but, similarly cannot guarantee a globally optimal clustering. Other more complex methods can achieve a globally optimal clustering (e.g. deterministic annealing (DA) [13]), but are computationally expensive and require many iterations, preventing their practical use in clinical segmentation or for processing large medical image archives. Indeed, in image processing techniques, it is typical to have a trade-off between complexity, robustness, automation, and speed [8]. Fast segmentation methods are often not sufficiently robust or require large amounts of user interaction, whereas, complex methods are often robust, but are computation and time prohibitive.

Thus, some researchers have explored parallelization of clustering techniques to achieve fast execution times. Several implementations of a parallel K-Means clustering [14, 5, 2, 15] have been analyzed and have achieved good speed up results. Other research has attempted to exploit algorithmic parallelism of mean shift [1, 15, 18] and deterministic annealing [12] on grid computing or SIMD architectures. On medical images, Gammage *et. al* [6] has used fuzzy connectedness to perform a segmentation using OpenMP on a SMP system resulting in a speed up of nearly 50x using four processors. Although many grid or multi-core implementations scale fairly well, they become power, cost, and resource limited at high speedup factors. For instance, in [15], the speed up factor of several parallelized algorithms is linearly related to the number of processors. On the other hand, K-means [14] and mean shift [18] have been analyzed on GPU architectures, but unanswered questions remain regarding the adoption of complex clustering algorithms on the newest CUDA (Compute Unified Device Architecture) enabled GPU architectures.

In this paper, we use the CUDA [10] programming language and GPU infrastructure to implement four clustering algorithms: k-means, mean shift, deterministic annealing (DA), and a novel spatially coherent adaptive deterministic annealing method (spatial DA). Next, we analyze these methods to realize trends and identify where computational bottlenecks exist. Finally, we combine several of these methods together to produce a robust algorithm called *Parallel Mode Deterministic Annealing* (PMDA) and apply PDMA to our cervigram image segmentation problem. The proposed approach uses a mean shift algorithm to detect the modes of an image and a spatial DA method to render a globally optimal clustering, unaffected by cluster initialization. Our method resembles [3], but incorporates spatial continuity features and exploits the inherent parallelism of mean shift and spatial DA. Our spatial DA method has roots in [17, 9], but the previous methods presented a spatial constraint that is either too general (single scalar for the entire image) or too specific (constrained to small window). Thus, our contributions are to,

1. provide a comparative analysis of four clustering algorithms on the GPU using CUDA
2. utilize the massively parallel architecture of GPUs to produce results nearly 38x (DA), 134x (mean shift), and 276x (spatial DA) faster than equivalent CPU implementations for a 1720x1172 pixel image
3. present PMDA, an unsupervised, automatic segmentation method for color medical images which is able to achieve an accurate globally optimal clustering in a time frame suitable for clinical use and on large scale medical image data sets
4. introduce an adaptive, spatially coherent clustering GPU extension to DA

## 2 Compute Unified Device Architecture

CUDA is a parallel programming model and software environment that is able to leverage the computational power of the latest generation Nvidia GPU processors.

*CUDA Hardware Model*: A GPU is a collection of stream processors that can be thought of as a massively multi-core processor. In the case of our experiment, one of the GPUs we used was a Nvidia Geforce 280 GTX. This card has a total of 240 streaming processing cores and 1 GB of total graphics memory. Each stream processor executes the same instruction but on different areas of memory. Additionally, each multiprocessor has local register memory, fast shared memory, and unrestricted access to texture and global memory. However, use of different types of memory incur different access times. One of the more recent modifications for GPUs is the addition of 64-bit support which has shown to be crucial when dealing with high precision algorithms (e.g. deterministic annealing).

*CUDA Software Model*: The CUDA software model and execution model can be most easily described from the bottom up. The most basic computational elements are *threads*. A collection of threads, called a *block*, runs on a multiprocessor and the block size is defined by the user. Threads within the block are able to share resources such as registers and shared memory. The combination of all the blocks makes up the *grid*. Typically, the grid covers the entire area of computation for your GPU program (in our case, the image size). A single GPU program is called a *kernel*. Kernel programs are frequently launched by a standard CPU program where many parallel operations can be executed. These kernel programs do the bulk of our computation which we describe in detail in later sections.

## 3 Clustering Methods on the GPU

In this section we first describe our implementation of four clustering methods on the GPU. Then, we present our proposed PDMA method for a fast, automatic segmentation of cervigram images. For all of our clustering methods, we use $L^*a^*b^*$ color space because the $L^*a^*b^*$ space has been shown to be better for clustering and classification [4]. Each color channel is stored in texture memory as a 2D texture. The reason for using texture memory rather than global or constant memory is because *texture fetches*, i.e. reads from texture memory, can exhibit higher bandwidth. The GPU architecture is better able to hide the latency of the addressing calculations and there is 2D spatial locality in texture fetches.

### 3.1 K-means and CUDA implementation

The K-means algorithm is a very popular clustering algorithm that is quite fast; however, it is sensitive to outliers, initialization of cluster centers, and needs to be given a predetermined number of clusters.

*K-means CUDA Implementation and kernel description*: The K-means CUDA implementation begins with a random initialization of cluster centers. Each pixel is assigned a thread on the GPU and each thread belongs to the kernel grid. The dimensions of the grid for the K-means kernel encompass the entirety of the cervigram image.

1. over the entire grid, compute the distance of each pixel to each cluster center in feature space
2. choose the minimum distance centroid (cluster center) and store all $width \times height$ labels in a 1D GPU global memory array of size $width \times height$
3. on the CPU, update the centroids based upon the minimum distance labels

### 3.2 Mean Shift Mode Detection and CUDA implementation

The mean shift [4] procedure is a non-parametric density estimation method that finds the *modes*, or local maxima of the empirical probability density function (p.d.f). This is a more complex clustering method that has the advantage of not needing to know the number of clusters *a priori*. It uses a density distribution kernel

to shift a window of size $h$ in the direction of higher density, where the direction is defined by the mean shift vector. Thus, the kernel will eventually converge at the local density maxima in feature space. Our radially symmetric profile at data point $x$ is defined as,

$$g(x) = -\frac{1}{2}\exp\left(-\frac{1}{2}x\right) \qquad x \geq 0 \tag{1}$$

Using $g(x)$, the kernel used in our mean shift procedure, $G(x)$ is defined as,

$$G(x) = c_{g,d}\, g(\|x\|^2) \tag{2}$$

where $c_{g,d}$ is the normalization constant. The mean shift procedure involves successive computation of the mean shift vector $m_{h,G}(x)$ and translation of the kernel window $G(x)$ by $m_{h,G}(x)$. This process repeats until the magnitude of the mean shift vector converges to 0. Given $n$ data points $x_i$, $i = 1,\ldots,n$, and window size $h$, the mean shift vector can be defined as,

$$m_{h,G}(x) = \frac{\sum_{i=1}^{n} x_i g(\|\frac{x-x_i}{h}\|^2)}{\sum_{i=1}^{n} g(\|\frac{x-x_i}{h}\|^2)} - x \tag{3}$$

where $x$ is the center of the kernel window.

*Mean Shift CUDA Implementation and kernel description*: The dimensions of the grid for the mean shift kernel encompass the entirety of the cervigram image. Each pixel is assigned a thread on the GPU and each thread runs the mean shift procedure to convergence;

1. over the entire grid, compute the mean shift vector $m_{h,G}(x)$ as in equation (3) and translation of the kernel window $G(x)$ in the color feature space
2. store all $width \times height$ modes in a 1D GPU global memory array of size $width \times height$
3. on the CPU, merge all $width \times height$ modes within a provided bandwidth parameter, $h$

## 3.3 Deterministic Annealing (DA)

The DA method [13] is a globally optimal clustering method that introduces an element of randomness to a clustering problem to avoid being stuck in local minima. Similar to simulated annealing, the procedure does not always take the greedy decision and is able to jump out of local minima. The amount of jumping can be thought of as the temperature and as the temperature cools and decreases the randomness, the clustering should converge to a global minimum. The mass-constrained clustering DA method that is independent of cluster center initializations is defined as the minimization of,

$$F = D - TH \tag{4}$$

where $H$ is defined as the Shannon entropy in equation (5), $T$, temperature, is the Lagrange multiplier, and $D$ is the distortion in equation (6). $H$ is defined as,

$$H(X,Y) = -\sum_{x}\sum_{y} p(x,y)\log p(x,y) \tag{5}$$

where $x \in X$ is the set of data points and $y \in Y$ is the set of clusters. $D$ is the expected distortion,

$$D = \sum_{x}\sum_{y} p(x,y)d(x,y) = \sum_{x} p(x)\sum_{y} p(y|x)d(x,y) \tag{6}$$

Our distortion measure $d(x,y)$ was defined as the Euclidean sum of squared differences between $x$ and the center of cluster $y$ in each color channel. Similar to fuzzy membership, our $x$ data points are associated with

(a) Original (b) Deterministic Annealing (c) Spatially Coherent DA

Figure 1: *Example of DA clustering without (b) and with (c) spatial constraints.* $\sigma_2 = 1.0$.

some probability to every cluster $y$ with probability magnitude $p(y|x)$. When $T$ is a large value, the equation maximizes the entropy; however, as $T$ is lowered, the equation minimizes $D$. Given $y_j$ as the distinct cluster center, the association probabilities are calculated by,

$$p(y_j|x) = \frac{p(y_j)\exp\left(-\frac{d(x,y_j)}{T}\right)}{Z_x} \tag{7}$$

where $Z_x$ is defined as the normalization constant,

$$Z_x = \sum_i^k p(y_i)\exp\left(-\frac{d(x,y_i)}{T}\right) \tag{8}$$

Here, $k$ is the number of cluster centers. To minimize the distortion with respect to the cluster center, $y$, the gradients can be set to zero, yielding,

$$\sum_x p(x)p(y|x)\frac{d}{dy}d(x,y) = 0 \qquad \forall y \in Y \tag{9}$$

### 3.4 Spatially Coherent Deterministic Annealing (spatial DA)

In many image clustering problems, including our cervigram application, regions of similar features tend to appear together spatially [17]. However in the basic DA method, these spatial features are ignored in the global optimization problem. To incorporate spatial coherency to the DA method, we introduce a weighting function that influences the DA distortion measurement to incorporate the membership of the current pixel as well as its surrounding 3x3 neighborhood. The idea is that neighboring pixels can influence the current pixels' cluster membership based on their feature space similarity. Thus, neighboring pixels with similar features have a greater probability to belong to the same cluster. Let $(r,s)$ represent a pixel location and $(r_n,s_n)$ represent the neighbor. The weighting function of the neighboring pixels on $(r,s)$ is defined as,

$$\lambda^{r,s}(\delta) = \frac{1}{1+\exp\left(\frac{-(\delta-\mu)}{\sigma_1}\right)} \tag{10}$$

where $\mu$ shifts $\lambda$ and is defined in equation (13), and $\sigma_1$ specifies the steepness of the sigmoid curve. The $\delta$ value is computed by,

$$\delta_{(r,s),(r_n,s_n)} = (x_{r,s} - x_{r_n,s_n})^T(x_{r,s} - x_{r_n,s_n}) \tag{11}$$

where $x_{r,s}$ is the feature vector and $x_{r_n,s_n}$ is the neighbor's's feature vector. This equation (11), computes the Euclidean distance between the two feature vectors. We can now use the following distortion term as a substitute for $d(x,y)$ in equations (7),(8), and (9),

$$d'(x,y) = \frac{1}{8}\sum_{l_1=-1}^{1}\sum_{l_2=-1}^{1}\left[d(x_{r,s},y)\lambda^{r,s}_{l_1,l_2} + d(x_{r+l_1,s+l_2},y)(1-\lambda^{r,s}_{l_1,l_2})\right] \tag{12}$$

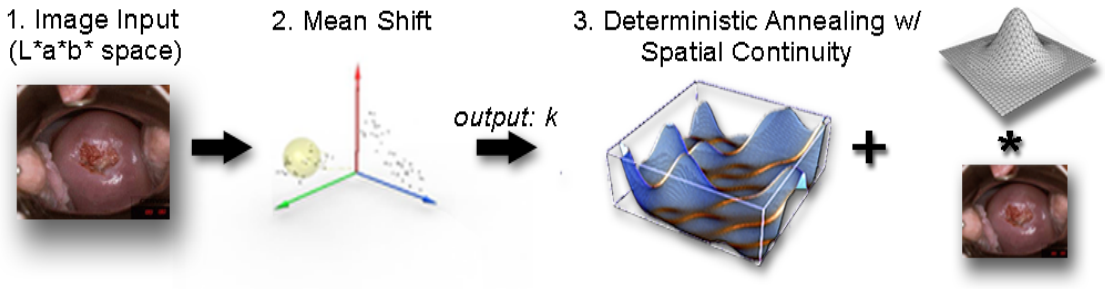



Figure 2: *PMDA Method*

where $\lambda_{i,j}^{r,s} = \lambda(\delta_{(r,s),(r_i,s_j)})$.

Research presented in [17, 9], describes spatial constraints either too general (single scalar for the entire image) or too specific (constrained to small window). Our method approximates $\mu$, while parameterizing a Gaussian smoothing technique to encompass as large an area as desired. This term is able to be precomputed and stored in texture memory for fast fetching on our GPU implementation. Our $\mu$ is defined as,

$$\mu_{(r,s)} = 1 - || \bigtriangledown G_{\sigma_2} * I_{(r,s)} ||_2 \tag{13}$$

where our image is convolved with a smoothing Gaussian,$G_{\sigma_2}$. There is an inverse relationship between pixel busyness and image gradient. If the gradient at a pixel is high, this can be thought of as high busyness and thus the $\mu$ would be small. The high gradient would be present in non-homogeneous regions and would shift the weight influence on the center pixel to a smaller value. Given our parameterized method, we can adjust the amount of spatial continuity desired at each pixel. Fig. 1 contrasts the results between the basic DA method and the new spatial DA method.

### 3.5 DA and spatial DA CUDA Implementation

Because the framework of the DA and spatial DA methods differ only by their distortion terms, we represent both methods in this section. At each temperature state, the DA algorithm consists of three steps,

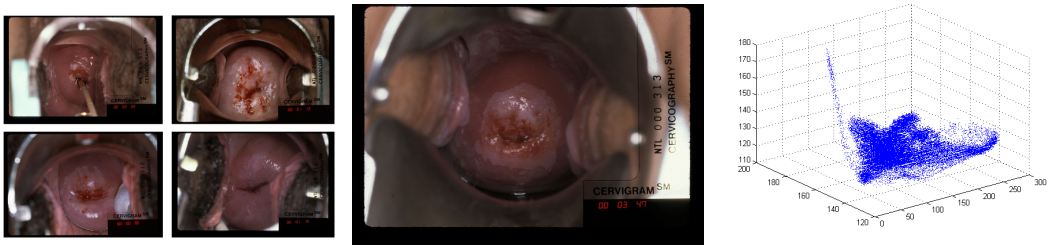
1. fix the cluster centers and use equation (7) in a GPU kernel to compute the association probabilities.
2. fix the association probabilities and optimize the cluster centers using (9). The maximum association probabilities are stored in a 1D GPU global memory array of size $width \times height$, one array for each color channel. The current individual cluster centers are stored in a 1D GPU global memory array of size $k$.
3. use the CUBLAS libraries (CUDA Basic Linear Algebra) to compute the parallel reduction of the arrays to find the final vector positions and probabilities.

This kernel is computed $\forall y \in Y$ until the cluster centers converge. Upon convergence, the cooling schedule will determine the next value of $T$ and will continue until $T < \varepsilon$.

### 3.6 PMDA Methodology

Our *Parallel Mode Deterministic Annealing* (PMDA) method is a combination of a mean shift mode detection followed by a spatial DA clustering algorithm. Since we cannot assume a fixed number of clusters for our image segmentation, we use a mean shift algorithm to detect the number of color modes and we use this characteristic, $k$, to initialize our spatial DA clustering method. The method consists of three steps (Fig. 2),

1. input the image in $L^*a^*b^*$ color space
2. compute the modes, $k$, of an image by a mean shift clustering algorithm.
3. compute the optimal clustering using spatial DA with parameter, $k$.



(a) Several image examples (b) Cervigram image (c) Feature space

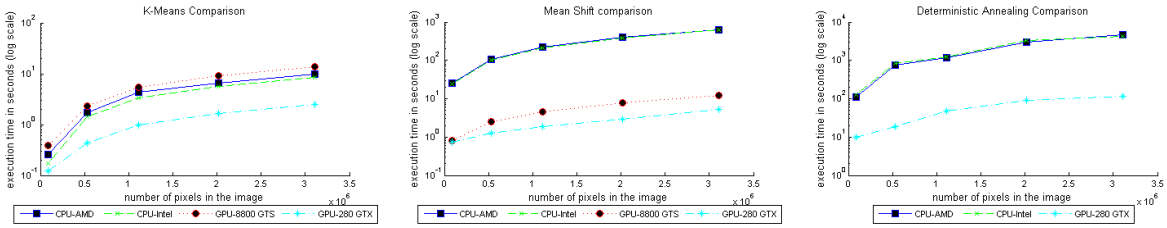
Figure 3: *(a) Example cervigram images used in our experiments depicting the high variability present in the data set. (b) a sample image with the corresponding 3D feature space plot in (c).*

## 4 Experimental Results

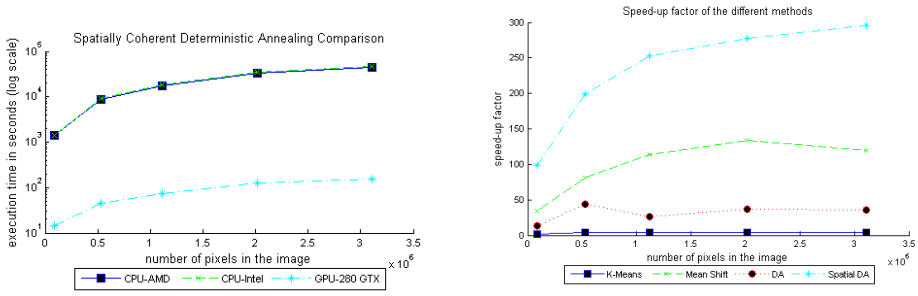
For our experiments, we evaluate our method on 8 uterine cervix images obtained from the NLM/NCI (Fig. 3). The hardware for these experiments consisted of several machines equipped with different GPUs. Our first machine is an AMD 4600+ dual core with a Nvidia 8800 GTS. The second machine is an Intel Xeon 3.0 Ghz dual core with a Nvidia 280 GTX. We evaluate our results on several aspects: computational speed, analysis, optimality, and validity.

### 4.1 Computational Speed and Analysis

We present our timing and speed up results in Fig 4. Similar to the work by Cao[2] *et. al*, we find a comparable speed up of 3-8x when implementing K-means on the GPU. With mean shift, we are able to greatly outperform other parallelized methods [1, 15], mainly because these methods are implemented for multi-core CPUs. Because their performance scales linearly to the number of processing cores, to match the speed up factor of our GPU implementation, these systems would need nearly two hundred cores (based on 1720x1172 image). Similarly with DA, the performance of [12] scales approximately linearly. Our GPU based method dominates these other methods in performance because of the number of streaming processors and high memory bandwidth that are present on GPUs.



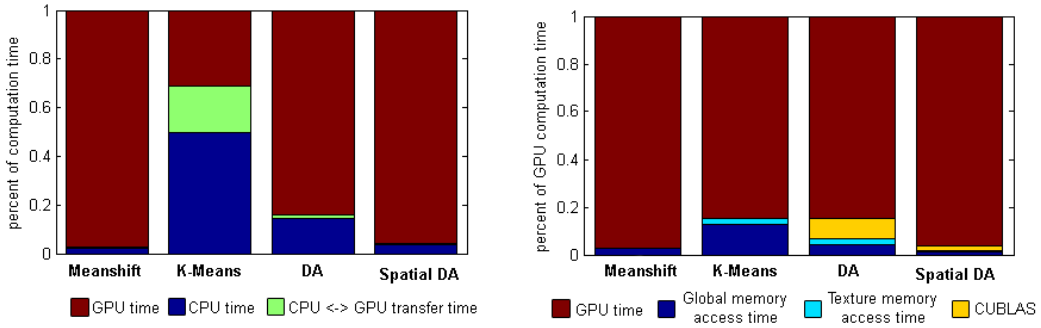

(a) K-means performance (b) Mean shift performance (c) DA performance




(d) Spatial DA performance (e) GPU Speed-up

Figure 4: *Execution times of the four clustering implementations when mean shift determined 11 modes, k=11 for all other methods. (e) Speed-up factor determined from slowest CPU to fastest GPU.*

Next, we analyze each method at each stage of the execution in a 880x600 pixel image to identify where the computational bottlenecks exist. The CPU and GPU execution times of the individual methods are compared in Fig. 5(a). From our results, we can deduce several key observations when parallelizing these clustering algorithms. First, it can be seen from the utilization graphs of both the mean shift and spatial DA methods that when the GPU time is highly utilized, the algorithm is offloading more computations to the GPU and can attain an overall greater speed-up factor. Second, highly iterative methods can perform extremely well on GPU architectures when data transfer between the CPU and GPU can be minimized. For example, in our K-means implementation, this data transfer is unavoidable, which limits our speed up factor. Third, the utilization of specific memory structures, such as shared memory and texture memory rather than global memory can help in lowering the total execution time. We frequently benefit from texture memory in our implementations but would like to explore shared memory in future work. A detailed breakdown of the GPU executions is presented in Fig. 5(b).




(a) Comparison of Computation Time

(b) Breakdown of GPU execution time

Figure 5: *Percentage of execution time for each process*

## 4.2 Optimality of Cluster Centers

We define an optimal clustering as the minimization of the Euclidean sum of squared differences (SSD) between the pixel cluster assignment and location of the final cluster center in feature space. We compare different clustering methods and present our results in Table 1. From our mean shift clustering with window sizes, $h_l = 0.08, h_a = 0.03, h_b = 0.03$, we detect 17 modes in a 1280x872 cervigram image. In 250 K-means trials we randomly initialized the cluster centers, set the $k = 17$ and the convergence criteria to $\varepsilon < 0.001$. For the DA method, we ran 50 randomized inputs, set the $k = 17$ and the cooling schedule to $0.8T$, and computed the annealing steps until $T < 5 \times 10^{-5}$. We also ran our spatial DA method on 50 randomized inputs with $\sigma_1 = 0.25$ and $\sigma_2 = 1$ to demonstrate its optimality. The presented SSD energy is slightly higher than the spatial distortion energy, but for direct comparability, the SSD energy is reported.

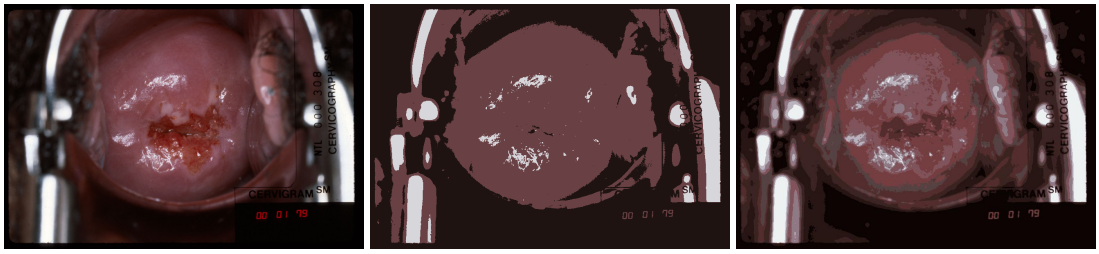
Because the mean shift method takes every pixel as an input, there is no initialization randomization to the method and the average energy is static. From our results, the simplistic K-means method has very high variance, and is highly sensitive to its initialization as seen in Fig. 6(b). Conversely, our DA algorithm always produces an approximate globally optimal result regardless of initialization (Fig. 6(c)).

Table 1: Energy comparisons between different methods, ($\sigma$ = standard deviation)

| Method | # of Trials | Min Energy | Max Energy | Average Energy | $\sigma$ = sd |
|---|---|---|---|---|---|
| Mean Shift | n/a[1] | n/a[1] | n/a[1] | 9,019.59 | n/a[1] |
| K-Means | 250 | 1,521.43 | 10,824.08 | 3,325.92 | 1,695.15 |
| Determinstic Annealing | 50 | 884.51 | 1,264.13 | 1,044.14 | 86.28 |
| Spatially Coherent DA[2] | 50 | 1,276.59 | 2,780.63 | 1,791.61 | 375.01 |

[1] *Mean shift is initialized with every pixel, thus, there is no deviation between trials*

[2] *Spatially Coherent DA distortion energy is calculated differently than the other methods. Basic SSD shown here.*



(a) Original (b) K-Means (c) Spatially Coherent DA

Figure 6: *K=13 for the different methods. (b) K-Means result from a poor initialization. (c) Spatial DA method with the same initialization as K-Means but reaches optimal clustering.*

### 4.3 Validity

To measure the validity of our clustering results, we compared selected AW tissue clusters with expertly segmented ground truths obtained from the NLM/NCI. We evaluated the p and q (sensitivity and specificity) and also the dice similarity coefficient (DSC) for 8 randomly chosen samples. The calculations of p, q, and dsc can be found in Popovic *et. al* [11]. Depending on the variation within each image, the number of modes detected by our PMDA method varied from 4 to 19. On average, our results are as follows: for the basic DA method, p = 0.723, q = 0.875, and dsc = 0.731. When we applied spatial coherency, our averages improved across all measurements. This was typically because the spatial method eliminated outliers and better delineated boundary conditions as seen in Fig. 7(c)(d). The most significant improvement was a 4.3% increase in the sensitivity. Our spatial coherent DA results were p = 0.756, q = 0.885, and dsc = 0.738.

## 5 Conclusion

In conclusion, we implemented and analyzed four clustering algorithms on the GPU using the CUDA programming language. We combined several of these methods and introduced a robust method, PMDA, for automatically segmenting cervigram images. We exploited the parallelism inherent in two algorithms, mean shift and deterministic annealing, using the GPU to achieve significant speed increases. We were able to achieve an approximate globally optimal clustering with relatively few resources (time and hardware). Also, a new spatially coherent extension to the deterministic annealing algorithm was introduced. Although we used cervigram images for our process, this clustering method can be extended for use on many different applications (e.g. histology images). In our future work, we would like to extend our algorithm to automatically classify cluster results and explore the use of *a priori* knowledge to more accurately segment and cluster image regions.

## References

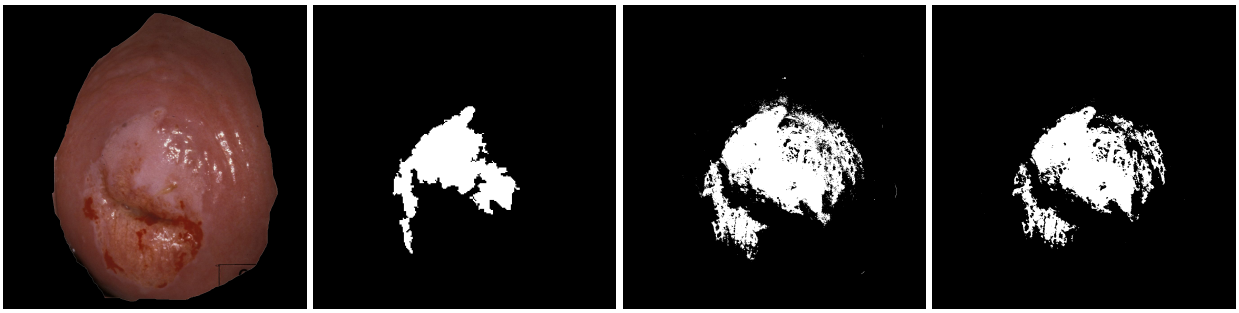
[1] J.G. Allen, R.Y.D, Xu, and J.S. Jin. Mean shift object tracking for a SIMD computer. *Third Int'l Conference on Information Technology and Applications*, 1:692–697, 2005. 1, 4.1

[2] F. Cao, A. Tung, and A. Zhou. Scalable clustering using graphics processors. *Advances in Web-Age Information Management*, 4:372–384, 2006. 1, 4.1

[3] W. Cho, S. Park, and J. Park. Segmentation of color image using deterministic annealing EM. *15th Int'l Conference on Pattern Recognition*, 3:646–649, 2000. 1

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach towards feature space analysis. *IEEE Trans. PAMI*, 24:603–619, 2002. 1, 3, 3.2

[5] E. Gabriel, V. Venkatesan, and S. Shah. Towards high performance cell segmentation in multispectral fine needle aspiration cytology of thyroid lesions. *HP-MICCAI Workshop*, 2008. 1



(a) Original (b) Ground Truth (c) DA result (d) Spatial DA result

Figure 7: *(a) Original image with (b) ground truth and selected regions (c)(d) identified as Acetowhite.*


[6] C. Gammage and V. Chaudhary. On optimization and parallelization of fuzzy connected segmentation for medical imaging. *International Conference on Advanced Information Networking and Applications*, 2:623–627, 2006. 1

[7] S. Gordon, G. Zimmerman, R. Long, S. Antani, J.Jeronimo, and H. Greenspan. Content analysis of uterine cervix images: Initial steps towards content based indexing and retrieval of cervigrams. *In Proc. of SPIE medical imaging*, 6144:1549–1556, 2006. 1

[8] C. Kulikowski and L. Gong. Progress in high performance medical imaging. *International Conference on Multimedia and Expo*, pages 284–287, 2007. 1

[9] A.W.C. Liew, S.H. Leung, and W.H. Lau. Fuzzy image clustering incorporating spatial continuity. *IEE Proc. Visual Image Signal Processing*, 147:185–192, 2000. 1, 3.4

[10] Nvidia. Cuda: Compute unified device architecture programming guide. *Technical Report*, 2008. 1

[11] A. Popovic, M. Engelhardt D. La, and K. Radermacher. Statistical validation metric for accuracy assessment in medical image segmentation. *International Journal of Computer Assisted Radiology and Surgery*, 2:169–181, 2007. 4.3

[12] X. Qiu, G. Fox, H. Yuan, S. Bae, and G. Chrysanthakopoulos. Performance of multicore systems on parallel data clustering with deterministic annealing. *ICCS*, 5101:407–416, 2008. 1, 4.1

[13] K. Rose. Deterministic annealing for clustering, compression, classification, regression and related optimization problems. *Proc. IEEE*, 86:2210–2239, 1998. 1, 3.3

[14] H. Takizawa and H. Kobayashi. Hierarchical parallel processing of large scale data clustering on a pc cluster with GPU co-processing. *Journal of Supercomputing*, 36:219–234, 2006. 1

[15] H. Wang, J. Zhao, H. Li, and J. Wang. Parallel clustering algorithms for image processing on multicore CPUs. *International Conference on Computer Science and Software Engineering*, 2008. 1, 4.1

[16] W. Wang, X. Huang, Y. Zhu, D. Lopresti, L.R. Long, S. Antani, Z. Xue, and G. Thoma. A classifier ensemble based on performance level estimation. *In Proc. of the IEEE ISBI*, 2009. 1

[17] Z.M. Wang, Q. Song, and Y.C. Soh. Mri brain image segmentation by adaptive spatial deterministic annealing clustering. *In Proc. of the IEEE ISBI*, pages 299–302, 2006. 1, 3.4, 3.4

[18] L. Zhu, C. Wang, G. Zhu, B. Han, H. Wang, P. Huang, and E. Wu. Image spatial diffusion on GPUs. *Circuits and Systems, IEEE Asia Pacific Conference*, pages 610–613, 2008. 1