Towards High Performance Cell Segmentation in Multispectral Fine Needle Aspiration Cytology of Thyroid Lesions

Edgar Gabriel, Vishwanath Venkatesan, and Shishir Shah

University of Houston
Department of Computer Science
Houston, TX 77204-3010
{gabriel,venkates,shah}@cs.uh.edu

Abstract. Thyroid nodule is a common cancer of the thyroid gland that affects up to 20% of the world population and approximately 50% of 60-year-old persons. Early detection and screening of the disease, especially analysis by fine needle aspiration cytology (FNAC), has led to improved diagnosis and management of the disease. Simultaneously, advances in imaging technology has enabled the rapid digitization of large volumes of FNAC specimen leading to increased interest in computer assisted diagnosis (CAD). This has led to development of a variety of algorithms for automated analysis of FNAC images, but due to the large scale memory and computing resource requirements, has had limited success in clinical use. In this paper, we present our experiences with two parallel versions of a code used for texture-based segmentation of thyroid FNAC images, a critical first step in realizing a fully automated CAD solution. An MPI version of the code is developed to exploit distributed memory compute resources such as PC clusters. An OpenMP version is developed for the currently emerging multi-core CPU architectures, which allow for parallel execution on every desktop system. Experiments are performed with image sizes ranging from 512×512 pixels with 31 channels to 8192×8192 pixels with 21 channels. Each parallelization is evaluated for performance and scalability.

1 Introduction

Cancer continues to remain a major health problem in the United States, with one of two men and one of three women developing cancer in their lifetime. Among various cancers, thyroid nodule is a common cancer of the thyroid gland. It has been estimated that up to 20% of the world population and approximately 50% of 60-year-old persons have palpable thyroid nodule or nodules [1]. The clinical spectrum ranges from the incidental, asymptotic, small, solitary nodule, in which the exclusion of cancer is a major concern, to the large, partly intrathoracic nodule that causes pressure symptoms, for which treatment is warranted regardless of cause. In spite of the growing incidences of thyroid lesions, the rate of thyroidectomies is on the decline. Early detection of the disease has been

partly responsible for improved outcomes. Among screening and detection procedures, fine needle aspiration (FNA) is believed to be a safe, inexpensive, and minimally invasive procedure to diagnose tumors [2]. For cytological evaluation of FNA samples, smears are appropriately prepared and stained. Typically, the stain changes the color of the cells and tissue so that examination of the smear under standard microscopes with moderate magnification (20–40x) is sufficient for clinical evaluation.

With the advances in imaging technology, there is considerable interest in automated analysis of FNA cell smears that could help to reduce the time required for manual screening and increase the detection rate of abnormalities [3]. Several commercial products such as ScanScope from Aperio Technologies, DX-40 from DMetrix, Inc., and iScan from BioImagene, Inc. have been developed to automate the process of digitizing microscope slides. They provide high throughput capabilities to digitize cell smears, resulting in a stitched image of scan areas of the order of 1.5cm x 1.5cm in less than 5 minutes. This provides a single image per smear that can be as large as 60,000 x 60,000 pixels under 40x magnification (resolution of 0.25μm/pixel). More recently, multispectral microscopes capable of acquiring spectral images under transmitted illumination have also been used to digitize and analyze cell smears [4, 5]. Spectral imaging allows for the simultaneously measurement of spectral and spatial information of a sample such that the measurement of the spectral response at any pixel of a two-dimensional image is possible. A spectral image consists of a series of images and each image is acquired under a narrow band wavelength of light. Studies have shown that biological tissue exhibits unique spectra in transmission. By exploring the spectral differences in tissue pathology, many chemical and physical characteristics not revealed under traditional imaging systems can be realized and used to improve the analysis efforts. Several efforts have already resulted in algorithms for cell segmentation, morphometric and karyometric feature analysis, as well as computer assisted diagnosis (CAD), with cell segmentation being the most challenging step for automated systems. However, to our knowledge, most of these efforts have been relatively limited in size due to the large data size and the computational bottlenecks. It is not uncommon to acquire anywhere from 5 to 31 spectral channels for each sample. Considering an average size of the smear to be 1.0cm x 1.0cm, the image cube to be analyzed would be approximately 8GB to 50GB in size. This creates difficulties in analyzing the entire data set on a standard desktop.

In this paper, we present our experiences with two parallel versions of a code used for texture-based image segmentation. Our main interest here is not to define the best segmentation algorithm, but to define a set of processes that would be realistically required in a typical CAD system. Specifically, we use Gabor filters for texture measurement and combine it with absorption computed from the spectral image stack to generate a feature vector for each pixel. K-means clustering is used to group pixels into different classes resulting in the segmentation of thyroid cells. An MPI version of the code has been developed to exploit distributed memory compute resources such as PC clusters. An OpenMP version

has been developed for the currently emerging multi-core CPU architectures, which allow for parallel execution on every desktop system. The rest of the paper is organized as follows: section 2 provides a brief overview of multispectral microscopy and the digitization of thyroid lesion cell smears. The texture-based approach for segmentation of thyroid cells is presented in section 3. Section 4 presents the parallelization strategy. The experiments performed for large scale analysis of entire scans of smear samples and according results are presented in section 5. Finally, the paper is concluded in section 6.

2 Multispectral Microscopy

The core element of any spectral imaging system is the spectral dispersion component that separates light into its spectral components, and is coupled to a two-dimensional (2D) optical detector such as a CCD camera, or to an array of photomultiplier tubes (PMT). In our system, we use a quarter-meter class, Czerny-Turner type monochromator that provides a tunable light emission spectrum at 10nm resolution. We utilize a wavelength range from 400-700nm for this study. The monochromator is connected to an Olympus BX51 upright optical microscope such that the light output from the monochromator feeds in to the transmitted light path of the microscope. This allows for the use of conventional optical microscopy to acquire brightfield images at desired wavelengths (transmitted light). An Olympus UPlanApo 40X NA 0.9 was used for imaging. The Photometrics SenSys^{TM}CCD camera having 768 x 512 pixels (9x9μm) at 8-bit digitization is used which provides for high resolution low light image acquisition. Figure 1 (left) shows the system that has been assembled. To image each sample, the illumination from the monochromator was adjusted by achieving Köhler illumination for uniform excitation of the specimen. The condenser, aperture diaphragm, and the field stop were kept constant during measurements. Focusing was performed at the central wavelength of 550nm to minimize the chromatic aberration at all wavelengths. The system was calibrated as per the method proposed in [6]. Using a stepper controlled microscope stage, multiple images were acquired to cover the extent of the smear on the microscope slide. Resulting images were stitched to generate a composite mosaic.

A multispectral image allows the possibility to locate, discriminate, measure, and enumerate many entities within a specimen by detecting subtle differences among their individual spectral signatures[7]. Clearly, different stained cells will be spectrally distinct. However, spectral information in any cell can come from such optical processes as reflection and scattering. As long as the phenomenology is based on reproducible physical reality, classification of spectrally distinct species can be of great utility. Figure 1 (right) shows a subset of the spectral image (400nm, 500nm, 600nm, and 700nm) of a Papanicolaou stained cytological specimen. As seen the light absorption across various cellular constituents vary as a function of wavelength. This forms the spectral signature for each cellular entity.

To understand the spectral characteristics of biological samples, gray level image intensities may be used to determine the proportion of light transmitted

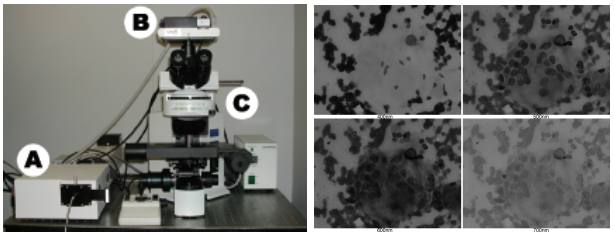

Fig. 1. Multispectral imaging system using a grating based spectral light source for transmitted illumination (left), and Four channels of a spectral image of a stained cytological smear (right).

by each cell across the exciting spectra. The transmission factor, T, is defined as:

$$T = I_t / I_i \tag{1}$$

where I_t is the intensity of the transmitted radiation at a point and I_i is the intensity of the incident light. Using the calibrated system, the incident light is fixed and known *a priori*. As such, one can then compute the absorption parameter for each pixel using the Beer-Lambert law[8] as:

$$A = \log(1/T) \tag{2}$$

For each pixel in the multispectral image, we measure the absorption parameter for all wavelengths to generate a feature vector representing the signature for that pixel.

3 Texture-based Cell Segmentation

Image segmentation is probably the most widely researched topic in image analysis and many attempts have been made to develop algorithms for segmentation of biomedical images. This is a critical problem since it forms the first step in identifying cells and tissue structures relevant for subsequent analysis. It is also the most challenging task due to the variabilities present in the images to be segmented. The most common approach for cytological image segmentation has been thresholding with the purpose of separating cells from the background. In some cases, learning algorithms coupled with clustering techniques have been used where pixels are assigned to either "cell" or "non-cell" class [9, 10]. Other mathematical formulations ranging from active contours, Fourier and Hough transforms, neural networks, and others [11, 12] have also been developed. An overview of segmentation techniques, specifically for cell segmentation can be found in [13].

More recently, textural features have been exploited for cell segmentation, especially for cytological and histological samples [14, 15]. Texture in an image can be obtained using a multitude of methods ranging from gray-level co-occurrence

matrices (GLCM), fractal measures, Law's texture measures, gradient structure tensors (GST), and Gabor filters [16–18]. In this work, we use a bank of Gabor filters to extract a measure of texture at each pixel followed by clustering to group pixels belonging to the same class. Specifically, we generate a bank of self-similar filters through appropriate dilations and translations of the basis Gabor function as defined by Manjunath et al. [19]. We use 3 scales and 4 orientations resulting in a total of 12 filters in the bank. To efficiently compute a measure of texture for each pixel in the multispectral image, we generate an average image from the multispectral stack. For each pixel in the average image, the magnitude response of each filter, the mean, and standard deviation is computed and stored as a feature vector. In addition, the absorption is measured according to equation 2 for each pixel in each channel of the multispectral image. Hence, we generate a 45-dimensional feature vector for each pixel belonging to a spectral image with 31 channels. The extracted features are clustered using the standard k-means algorithm [9] which results in effective grouping of pixels belonging to the thyroid cells and partitioning of the image. Figure 2 (left) shows an example of a thyroid cell smear image at 520nm (1 of 31 channels) and the result of clustering (right) with the cluster of pixels detected as thyroid cells overlayed in black.

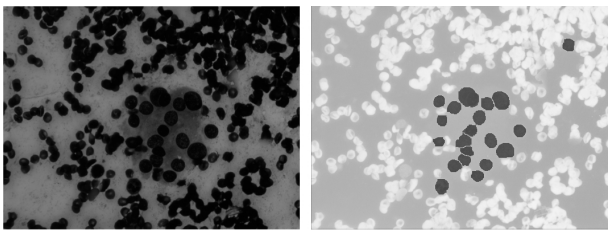

Fig. 2. One channel of a multispectral cell smear image (left) and result of clustering overlayed to highlight the cluster representing the cells of interest (right).

4 Parallelization Methods

In order to meet the challenge posed by the large images and the number of channels, we created two different parallel versions of the code: one version based on the Message Passing Interface (MPI) [20], which is targeting clustered environments; a second version based on OpenMP [21], the de-facto standard for shared memory programming, targeting the new generation of multi-core processors. The sequential code used as the basis for both parallel versions is implemented in C, using the FFTW library [22] for the Fast-Fourier Transform (FFT) operations. The parallel versions described here achieve the same results with the same accuracy as their sequential counterparts.

4.1 MPI parallel version

As with most MPI applications, the parallelization strategy used in this version of the code relies on data decomposition: each process holds one part of the overall image, and all processes execute the same code on different data items. FFTW version 2.1.5 supports MPI parallel FFT operations. However, the library mandates a one-dimensional data decomposition, i.e. each process holds a certain number of rows of the overall image. The size of the image passed to the FFTW routines have to be padded by the number of rows/columns of the Gabor filters, resulting in a slightly uneven distribution of rows for the unpadded image across the processes. I/O operations are implemented using collective MPI I/O routines.

The most challenging part of the MPI version was the parallelization of the k-means clustering routine. Assuming that the number of clusters are small compared to the number of pixels, we replicated the information about the clusters on all processes. Each process determines locally for each pixel in its domain the cluster whose center is closest to each pixel, and assigns that pixels to the appropriate cluster. The code also determines the number of pixels assigned to each cluster and the weight of each cluster. Following these local calculations are three global reduction operations, which determine the overall number of pixels assigned to each cluster across all processes, the overall weight of each cluster, and the global error, defined as the sum of the squared distance of each pixel to the center of the closest cluster. Using these global values, each process can independently determine the new center of each cluster for the next iteration of the algorithm. This iterative procedure is terminated when the error between two iterations is smaller than a predefined threshold.

The code than performs a smoothing operation by comparing the cluster assigned to a particular pixel with the clusters assigned to its neighboring pixels. In the MPI version access to information owned by another process is realized by introducing 'ghostcells', i.e. introducing copies of information (in this case which cluster does a pixel belong to) owned by another process. The size of the ghostcells is determined by the number of neighboring cells analyzed.

4.2 OpenMP parallel version

The OpenMP parallel version of the code also follows a data decomposition approach. Each of the three main modules (convolution, clustering, smoothing) has been parallelized individually and invoked in a sequential fashion in order to avoid nested parallelism. FFTW version 1.2.5 also supports OpenMP type thread level parallelism, which eased the parallelization of the filtering and convolution, as it only required an additional parameter (number of threads) to be passed to the functions. However, the initial version of this code section did not perform well. Profiling the application with the performance analysis tool *TAU* [23] revealed, that OpenMP directives inserted for a loop performing the padding of the image degraded the performance due to a large number of cache misses between the different threads. Thus, the performance of this code section could be improved significantly by *not* parallelizing it.

The k-means clustering revealed a series of other challenges. Since the code is organized in a large `while` loop, the very first idea was to parallelize this outer loop. However, OpenMP as of version 2.5 does not support parallelization of loops with unknown number of iterations at compile time. Following a similar approach as in the MPI version, OpenMP directives were then inserted to parallelize the access to the individual pixels. Determining the number of pixels assigned to a cluster and the weight of each cluster across all threads posed however some problems, since OpenMP does not support reductions over arrays in C in the current specification. Different alternatives have been explored to overcome this limitation including introducing critical regions around the clustering arrays, atomic updates, and locking only a particular value of the clustering data. The version leading to the best performance privatized the clustering arrays and performed global updates by executing element-wise reduction operations.

5 Evaluation

In the following, we present the performance of the code described in the previous section. The MPI measurements have been executed on the *shark* cluster at the University of Houston. The cluster consists of 24 single processor, dual core AMD Opteron nodes running at 2.2 GHz, each node equipped with 2GB of main memory. Nodes are connected by a 4xInfiniBand and a Gigabit Ethernet network. The compute nodes have access to an NFS mounted home file system as well as to a PVFS2 file system. The OpenMP measurements have been executed on *zeola*, a shared memory system consisting of eight dual core 2.6 GHz AMD Opteron processors with 64GB of main memory. We used gcc v4.2 and Open MPI v1.2.5Tests have been executed for image sizes ranging from 512×512 pixels with 31 spectral channels, to 1024×1024, 2048×2048, 4096×4096 and 8192×8192 pixels, all using 21 spectral channels. Since the image is stored in a raw, uncompressed format, the largest image analyzed was close to 1.5 GB. Each test presented in this paper has been repeated three times, and the minimum time over the three runs has been used.

Both parallel versions of the code show all-in-all a good scalability. The left part of figure 3 shows the execution time for each image on various number of processes for the MPI parallel version. The number of processes used was restricted by the main memory required for a given image and a minimum number of rows per process. This version shows very good scaling on shark when using the InfiniBand network, i.e. doubling the number of processes for a given image reduces the execution time in most instances by a factor of 1.6-1.9. The main exception from that behavior is observed for the largest image when utilizing 32 and 44 processes. The reason for this deviation is discussed in later sections.

The OpenMP version, shown in the right part of fig 3 scales similarly well. For the smaller images, using more than four threads typically does not further reduce the execution time of the code. For the largest image, the code achieved however a speedup of 5.9 when using eight threads, and 9.7 for 16 threads. Please note, that the OpenMP version of the code does not write the results of the convolution operation for each filter into output files. The comparison of the

execution times of the MPI and the OpenMP versions can only be performed on selected code sections.

An in-depth analysis of the application reveals, that the most time consuming parts are the k-means clustering followed by the convolution for both parallel versions. Only the 8192×8192 image reverses the order of costs for these two operations. Figures 4 and 5 analyze the scaling behavior for both, the MPI and the OpenMP version of these two code sections. Both algorithms scale well for large problem sizes. Most notably, the deviation from the very good scaling behavior of the MPI code for the 32 and 44 processor test cases mentioned above is clearly not due to the performance of these two routines. Comparing the execution time of the MPI and the OpenMP version for the same image and number of processes/threads reveals, that the MPI version usually outperforms the OpenMP version by approx. 30% for these two operations. This difference stems mainly from the overhead introduced by the runtime environment when managing an OpenMP application, which is also observable when comparing the performance of a sequential application vs. the OpenMP version using a single thread. Although this effect might be stronger for the gcc compiler used in these tests, this is effect is well understood and documented in the compiler research community [24].

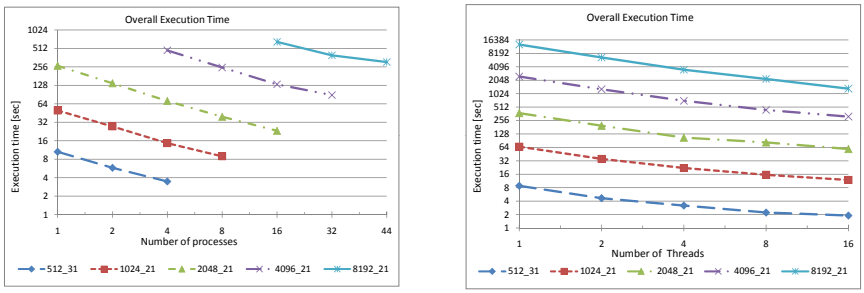

Fig. 3. Performance analysis of the MPI version of the overall code (left) and the OpenMP version (right).

5.1 Influence of I/O and the network interconnect

In order to understand the performance behavior of the MPI code for the largest image and processor size, we further dissect the execution time of the code. Beyond the convolution and the k-means clustering, the next most time consuming code sections are the routines dealing with reading input files and writing output files. The MPI version of the code has the option to write the texture data into

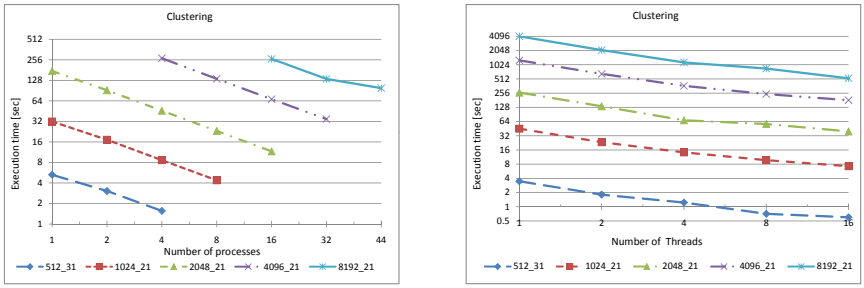

Fig. 4. Performance of the MPI (left) and OpenMP (right) version of the k-means clustering.

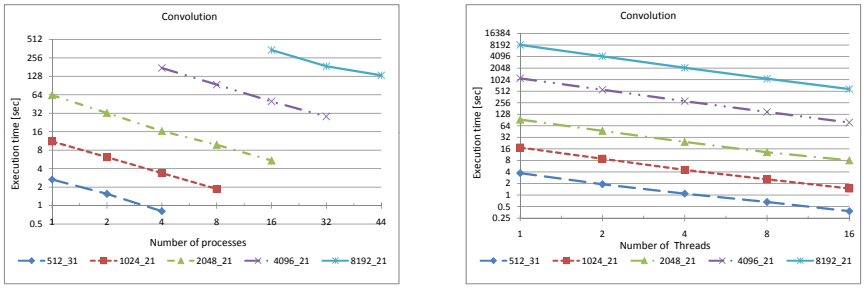

Fig. 5. Performance of the MPI version (left) and OpenMP (right) version of the convolution.

output files. This is mostly to facilitate future processing steps in realizing a complete CAD solution.

As described in section 4, I/O operations are performed using collective MPI I/O routines. For the 8192×8192 image, the size of the texture information stored in different files sums up to 6GB. Table 1 shows, that in most instances of the 2048×2048, the MPI I/O version of the code outperforms a version of the MPI parallel code, which uses regular POSIX style I/O operations. However, as shown in the left part of fig 6, the time spent in I/O operations increases dramatically for all 32 and 44 processes test cases. The reason probably is, that other test cases have only a single MPI process running per node, while the 32 and 44 processes test cases run two MPI processes on each node, one per CPU core. The implementation of the collective I/O routines within Open MPI seem to generate for these instances a significant overhead, and, as shown in

the lower part of Table 1, a POSIX I/O based parallel version outperforms in this case the MPI I/O version. The last column in the table also documents the performance penalty one would pay by using a standard NFS file system instead of the parallel PVFS2 file system. For the largest image, the time spent in I/O operations doubles to 115 seconds, with some runs taking up to 350 seconds spent in writing 6 GB of data. Thus, if the code would be executed using an NFS file system, I/O could be in fact the most time consuming portion of the code for large images.

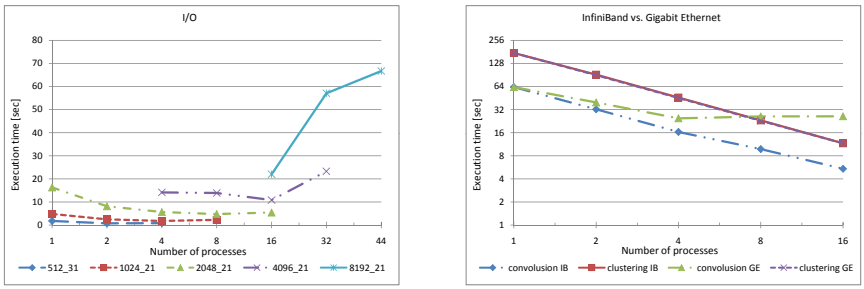

Fig. 6. Analyzing the I/O behavior of the MPI version (left) and comparing the performance of convolution and clustering over InfiniBand and Gigabit Ethernet (right).

Table 1. Comparing MPI I/O with POSIX I/O over PVFS2 and NFS.

Image	No. of processes	MPI I/O on PVFS2	POSIX I/O on PVFS2	POSIX I/O on NFS
2048×2048	2	8.1	11.2	-
2048×2048	4	5.6	8.5	-
2048×2048	8	4.7	5.9	-
2048×2048	16	5.4	3.9	-
8192×8192	16	21.9	49.0	115.9
8192×8192	32	57.0	51.0	114.4
8192×8192	44	66.7	50.4	115.9

Lastly, we evaluate the influence of the network interconnect on the two most time consuming routines in the code, namely the convolution and the k-means clustering. The right part of figure 6 compares the execution time of these two routines for the 2048×2048 image over 4xInfiniBand (latency: $3.5\mu s$, bandwidth: $1GB/s$) and Gigabit Ethernet (latency: $55\mu s$, bandwidth: $80MB/s$). The k-means clustering is nearly insensitive to the difference in the quality of the networks due to the low communication overhead introduced by the global

reductions. However, the convolution, which consists of a large number of FFTs, does not scale beyond four processes on the Gigabit Ethernet network, while it still shows a performance improvement for 16 processes for the very same test case when using InfiniBand. This can be explained with the communication pattern of a parallel FFT, which involves a large number of (small) messages, and therefore shows a sensitivity to the latency of a network.

6 Summary and Conclusions

This paper presented an MPI and an OpenMP parallel version of a code used for texture-based image segmentation applied to multispectral fine needle aspiration cytology of thyroid lesions. We evaluated the performance of both code versions using a series of images of increasing sizes, with 31 and 21 spectral channels. Both versions show good scalability all-in-all. A comparison of the most time consuming parts of the code between the two parallel versions shows a performance advantage for the MPI version. However, a detailed analysis of the MPI application reveals, that performance and scalability for this application strongly depends on state-of-the-art technology for the network and the file system. Off-the-shelf network interconnects such as Gigabit Ethernet and the popular NFS file system clearly do not show the technical capability to keep up with the requirements of this highly resource intensive application.

We plan to further extend this analysis by integrating additional functionality required for the overall goal, such as including classifiers or comparing texture information of an image to a data base of known cases. Since the final goal is to analyze images of up to 20GB sizes, we also plan to make further scalability tests beyond the resources currently available, e.g. through a Teragrid Grant. Our experiments provide useful insights into the ability to scale and parallelize typical image analysis algorithms. While the results presented here cannot be generalized for all segmentation algorithms, the use of convolution and k-means clustering is common in variety of image processing tasks, and the respective parallelization results of the two modules would generalize well characterized by the available hardware.

Acknowledgments. We would like to thank Oscar Hernandez for his support with the OpenMP version of the code.

References

1. Hegedus, L.: The thyroid nodule. New Eng J Med. **351** (2004) 1764–1771
2. Gharib, H., Goellner, J.R.: Fine-needle aspiration of the thyroid: an appraisal. Ann. Intern. Med. **118** (1993) 282–289
3. Association, T.A.T.: Thyroid fine needle aspiration (FNA) and cytology. In: Consensus guidelines for thyroid testing in the new millennium. Volume 6. (2003) 1–80
4. Shah, S., Schwartz, M.R., Mody, D.R., Scheiber-Pacht, M., Amrikachi, M.: The role of multispectral microscopy in differentiating benign and malignant thyroid

nodules: A pilot study of 24 cases. In: Proceedings of the Annual Meeting of the United States and Canadian Academy of Pathology. (2008)
5. Feng, C., Shuzhen, C., Libo, Z.: New abnormal cervical cell detection method of multi-spectral pap smears. Wuhan University Journal of Natural Sciences **12** (2007) 476–480
6. Shah, S., Thigpen, J., Merchant, F., Castleman, K.: Photometric calibration for automated multispectral imaging of biological samples. In Metaxas, D., Whitaker, R., Rittscher, J., Sebastian, T., eds.: Proceedings of 1st Workshop on Microscopic Image Analysis with Applications in Biology (in conjunction with MICCAI, Copenhagen). (2006) 27–33
7. Farkas, D., Ballou, B., Fisher, F., Fishman, D.: Microscopic and mesoscopic spectral bio-imaging. In: Proceedings of SPIE. Volume 2678. (1996) 200–209
8. Ornberg, R.L., Woerner, M., Edwards, D.A.: Analysis of stained objects in histopathological sections by spectral imaging and differential absorption. J. Histochem. Cytochem. **47** (1999) 1307–1313
9. Hartigan, J.A., Wong, M.A.: A K-means clustering algorithm. Applied Statistics **28** (1979) 100–108
10. Faugeras, O.D., Pratt, W.K.: Decorrelation methods of texture feature extraction. IEEE Trans. Pattern Analysis and Machine Intelligence **2** (1980) 323–332
11. Bamford, P., Lovell, B.: Unsupervised cell nucleus segmentation with active contours. Signal Processing **71** (1998) 203–213
12. Kurugollu, F., Sankur, B.: Color cell image segmentation using pyramidal constraint satisfaction neural network. In: IAPR Workshop on Machine Vision Applications. (1998) 85–88
13. Ablameyko, S., Nedzved, A., Lagunovsky, D., Patsko, O., Kirillov, V.: Cell image segmentation: review of approaches. In: Proc. ICPR. Volume 2. (2001) 26–34
14. Ferrer-Roca, O., Gomez, J.A.P., Estevez, M.: Chromatin texture from hematoxylin stained thyroid lesions. Anal Cell Pathol. **17** (1998) 209–217
15. Yogesan, K., Jorgensen, T., Albregtsen, F., Tveter, K.J., Danielsen, H.E.: Entropy-based texture analysis of chromatin structure in advanced prostate cancer. Cytometry **24** (1996) 268–276
16. Laws, K.I.: Textured Image Segmentation. PhD thesis (1980)
17. Tourassi, G.D., Frederick, E.D., Vittitoe, N.F., Coleman, R.E.: Fractal texture analysis of perfusion lung scans. Comput Boimed Res **33** (2000) 161–171
18. Shah, S., Aggarwal, J.K.: A Bayesian segmentation framework for textured visual images. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. (1997) 1014–1020
19. Manjunath, B.S., Ma, W.: Texture features for browsing and retrieval of image data. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI - Special issue on Digital Libraries) **18** (1996) 837–42
20. Message Passing Interface Forum: MPI: A Message Passing Interface Standard. (1995) http://www.mpi-forum.org/.
21. Board, O.A.R.: OpenMP Application Program Interface. (2005) Version 2.5.
22. Frigo, M., Johnson, S.G.: The Design and Implementation of FFTW3. Proceedings of IEEE **93** (2005) 216–231
23. Malony, A.D., Shende, S., Bell, R., Li, K., Li, L., Trebon, N.: Advances in the tau performance system. Performance analysis and grid computing (2004) 129–144
24. Huang, L., Eachempati, D., Hervey, M.W., Chapman, B.: Extending global optimizations in the OpenUH compiler for OpenMP. In: Open64 Workshop at CGO 2008, In Conjunction with the International Symposium on Code Generation and Optimization (CGO), Boston, MA, USA (2008)