31st Annual International Conference of the IEEE EMBS
Minneapolis, Minnesota, USA, September 2-6, 2009

Accelerated Detection of Intracranial Space-occupying Lesions with CUDA Based on Statistical Texture Atlas in Brain HRCT

Wei Liu, Huanqing Feng, Chuanfu Li, Yufeng Huang, Dehuang Wu and Tong Tong,

***Abstract*—In this paper, we present a method that detects intracranial space-occupying lesions in two-dimensional (2D) brain high-resolution CT images. Use of statistical texture atlas technique localizes anatomy variation in the gray level distribution of brain images, and in turn, identifies the regions with lesions. The statistical texture atlas involves 147 HRCT slices of normal individuals and its construction is extremely time-consuming. To improve the performance of atlas construction, we have implemented the pixel-wise texture extraction procedure on Nvidia 8800GTX GPU with Compute Unified Device Architecture (CUDA) platform. Experimental results indicate that the extracted texture feature is distinctive and robust enough, and is suitable for detecting uniform and mixed density space-occupying lesions. In addition, a significant speedup against straight forward CPU version was achieved with CUDA.**

I. Introduction

THERE has been growing interests in developing Computer-Aided Diagnosis (CAD) techniques with high-resolution Computer Tomography (CT) for detecting brain structural and anatomical abnormities[1][2]. Among these techniques, the construction of brain atlases has been central to the understanding of the variabilities of brain anatomy. Large sets of images are mapped into a common coordinate system, such as a deterministic or probabilistic atlas, to study intra-subject and inter-subject variabilities, and to provide voxel-wise statistical analysis of the resulting scalar or vector fields. Subsequently regions in which there are significant group or condition differences yielded by brain abnormities could be determined. Commonly used examples for these approaches include Statistical Parametric Mapping (SPM), RAVENS map[3], voxel-based morphometry (VBM)[4], and tensor-based morphometry (TBM)[4]. Promising detection results have been achieved in a variety of disease including, semantic dementia, Alzheimer's disease, schizophrenia, autism, bipolar disorder, and primary progressive aphasia. However, these methods were originally devised to identify subtle neuroanatomical changes associated with neurological and psychiatric dysfunction. When dealing with intracranial space-occupying lesions, such as tumor and stroke, satisfying lesion segmentation results cannot be easily achieved [5]. It is because such type of brain lesion usually induces relatively large displacements and complicated intensity patterns rather than subtle local morphological changes.

Recent studies advocate that intrinsic intensity patterns, especially regional texture signatures are much more reliable for mixed density lesion detection [6][7]. In the literature, methods for extracting image texture feature often rely on intensity gradient, moments, Gabor features, local frequency representations and wavelet coefficients. These methods have different detection performance and computational costs.

The purpose of this paper is to propose an effective detection scheme for intracranial space-occupying lesions, which uses statistical texture atlas to characterize brain local anatomical features. And our goal is to make the texture feature distinctive and robust enough. In addition, to make the detection method more applicable, we have implemented the pixel-wise texture extraction procedure on Nvidia 8800GTX GPU with Compute Unified Device Architecture (CUDA) platform. It should be noted that this study has examined only 2D CT images, and further implementations on 3D volumes will be summarized in our next study.

II. Methodology

Fig.1 provides a summary of the proposed intracranial space-occupying lesion detection scheme.

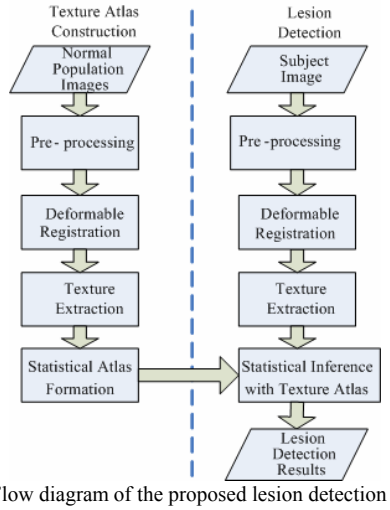

Fig.1. Flow diagram of the proposed lesion detection scheme

Wei Liu, Department of Electronic Science and Technology, University of Science and Technology of China (USTC), Hefei, China. (phone: +86-1-551-3601800, fax: +86-1-551-3601806; e-mail: bmewliu@mail.ustc.edu.cn)

Huanqing Feng (hqfeng@ustc.edu.cn), Yufeng Huang, Dehuang Wu and Tong Tong are with the Department of Electronic Science and Technology, University of Science and Technology of China, Hefei, China.

Chuanfu Li is with Medical Imaging Center of the First Affiliated Hospital, Anhui College of Traditional Chinese Medicine, Hefei, China.

A. Preprocessing

Preprocessing consists of two stages: *skull-stripping* and *rigid registration*. 1) In the first stage, a priori-based region growing algorithm is performed on 2D CT slices to extract brain regions, and then morphological operator is used to modify the initial segmentation results. 2) In the second stage, mutual-information-based rigid registration, provided by ITK, is employed in order to compensate for possible motion and position differences between individuals.

B. Deformable Registration

Demons algorithm, originally proposed by Thirion[8], is applied in the deformable registration stage. Demons is based on the optical flow method, which is used to determine small deformations in temporal sequences. The optical flow methods finds a displacement field that deforms the moving images, *M*, so that it is matched with the subject images *S*. The basic hypothesis of optical flow is that intensities are constant between *M* and *S*.

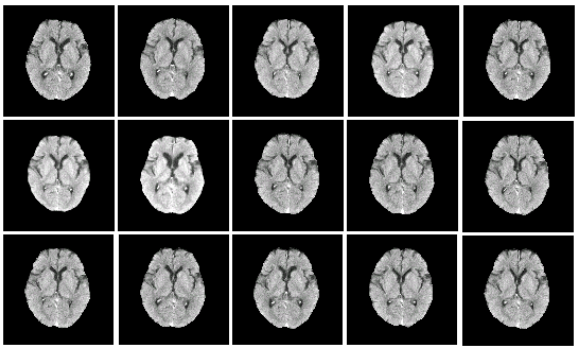

Fig.2. 2D Brain CT images after pre-processing and Demons registration

C. Texture Feature Extraction

Although a large number of related literature have focused on sophisticated texture extraction methods, it is still necessary to mention that more features will not always produce better results in characterizing brain anatomy[9][10], because detailed features vary dramatically across individual brains. To draw a balance, we introduce a relatively simple but discriminative and robust texture extraction method. Each point in the image is represented as a texture attribute vector, which uses multi-scale local intensity histogram and edge information to characterize the geometric features around the point at different resolutions.

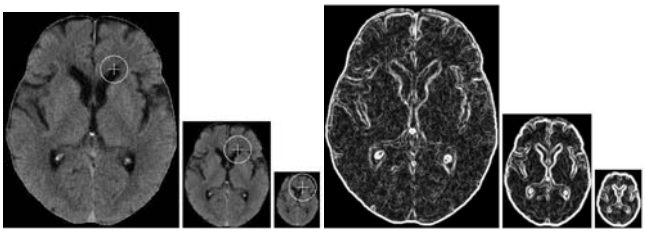

Fig.3. Schematic demonstration of texture extraction

1) Down-sampling: The original image $f(\boldsymbol{x})$, $\boldsymbol{x} \in \boldsymbol{R}^2$, is down-sampled by a factor of s, resulting in several down-sampled images at different levels, $f_s(\boldsymbol{x}_s)$, where $\boldsymbol{x}_s = [\boldsymbol{x}/s]$, and when $s=1$, $f_s(\boldsymbol{x}_s)$ is the original image. Gaussian filter is used to down-sample an 512×512 size image, and three resolution levels, i.e., $s=1,2,4$, are used, resulting in 512×512, 256×256, and 128×128 images.

2) Computing local histogram: For each resolution image $f_s(\boldsymbol{x}_s)$, a local histogram $h_s(\boldsymbol{x}_s)$ of intensities is calculated from a spherical region around point $\boldsymbol{x}_s$. The radius of spherical region is set to be identical across different resolutions, as shown in Fig. 3. Therefore, for each point $\boldsymbol{x}$ in the original image, we can obtain several local histograms from the multi-scale images, which capture different levels of spatial intensity distribution information around point $\boldsymbol{x}$.

3) Computing geometric moments: The statistical features are, respectively, extracted from each histogram $h_s(\boldsymbol{x}_s)$, by calculating its regular geometric moments.

$$m_s(\boldsymbol{x}_s, p) = \sum_i i^p h_s(\boldsymbol{x}_s, i), \quad p = 0,1,2 \qquad (1)$$

4) Adding Canny edge strength: In addition to using the intensity histogram as attributes, boundary information is also extracted in order to provide more accurate anatomical features. In this study, Canny edge detector is used to quantify the strength of boundary on each point.

5) Generating texture attribute vector: For a single resolution, texture attribute vector is defined as:

$$\boldsymbol{V}_s(\boldsymbol{x}_s) = [m_s(\boldsymbol{x}_s,0)\ m_s(\boldsymbol{x}_s,1)\ m_s(\boldsymbol{x}_s,2)\ c_s(\boldsymbol{x}_s)] \qquad (2)$$

Here, $c_s(\boldsymbol{x}_s)$ denotes Canny edge strength on point $\boldsymbol{x}_s$ at resolution s. Therefore, the final texture attribute vector can be represented as:

$$\boldsymbol{V}_T(\boldsymbol{x}) = [\boldsymbol{V}_1(\boldsymbol{x})\ \boldsymbol{V}_2(\boldsymbol{x})\ \boldsymbol{V}_4(\boldsymbol{x})] \qquad (3)$$

D. Statistical Texture Atlas Construction

Given a collection of normal brain images, we treat the texture attribute vector at the same point across the images as normally distributed. Specifically, the set $\boldsymbol{\Omega_v(x)}$ of the attribute vector of the corresponding points at a particular location $\boldsymbol{x}$ ($\boldsymbol{x} \in \boldsymbol{R}^2$) in a spatial normalized coordinate is approximated by a single multi-dimension Gaussian distribution with mean vector $\boldsymbol{V}$ and covariance matrix $\boldsymbol{\Sigma}_v$. Therefore, the value of probabilistic distribution function of a given attribute vector $\boldsymbol{V(y)}$, at location $\mathbf{y}$ belonging to $\boldsymbol{\Omega_v(x)}$ is defined as:

$$f\left(\boldsymbol{V}(\boldsymbol{y}) \mid \boldsymbol{\Omega}_v(\boldsymbol{x})\right) = \eta \cdot \exp\left(-\frac{1}{2}\boldsymbol{E}_v\right) \qquad (4)$$

Where, η is a normalization coefficient, and

$$\boldsymbol{E}_v = \left(\boldsymbol{V}(\boldsymbol{y}) - \overline{\boldsymbol{V}}\right)^T \boldsymbol{\Sigma}_v^{-1} \left(\boldsymbol{V}(\boldsymbol{y}) - \overline{\boldsymbol{V}}\right) \qquad (5)$$

As we know, $\boldsymbol{E}_v$ is actually the Mahalanobis distance between $\boldsymbol{V}(\boldsymbol{y})$ and $\overline{\boldsymbol{V}}$ in vector space. Apparently, the construction procedure is pixel-wise, and each point in the co-registered reference coordinate across individuals has its mean vector $\overline{\boldsymbol{V}}$ and covariance matrix $\boldsymbol{\Sigma}_v$, respectively.

E. Lesion Detection

So far, we have constructed the statistical texture atlas, which is parameterized by mean texture attribute vector $\overline{\boldsymbol{V}}$ and covariance matrix $\boldsymbol{\Sigma}_v$ of normal individuals. In this

sense, the problem of lesion detection can be restated as measuring how the texture attribute vectors of points in the input subject image are similar or belonging to the corresponding locations in the atlas. For simplicity, we use Mahalanobis distance $\boldsymbol{E}_v$ as the measurement.

III. IMPLEMENTATION

Graphics Processing Units (GPUs) have emerged as a powerful platform for high-performance computation. They have been successfully used to accelerate many scientific workloads[11][12]. As mentioned in Section I, the construction of statistical texture atlas is typically time-consuming and the implementation should be optimized. In practice, the former three operations, including skull-stripping, rigid registration and Demons registration are implemented in ITK platform, and only texture extraction has been transplanted on GPU with CUDA. It is because: 1) ITK is thread-parallel architecture and the program would also be accelerated; 2) texture feature extraction is naturally data independent and suitable for GPU implementation.

A. Programming Models on CUDA

CUDA is a general purpose parallel computing architecture that leverages the parallel compute engine in Nvidia graphics processing units (GPUs) to solve many complex computational problems in a fraction of the time required on a CPU. In CUDA, programs are expressed as kernels, which have a Single-Program, Mutiple-Data(SPMD) programming model. Applications that are executed many times, but independently on different elements of the dataset, can be isolated into a kernel that is running on GPU in the form of many different threads. The CUDA Kernel runs asynchronously on the GPU and executes many data-parallel threads simultaneously.

B. Texture Extraction

As explained previously, in order to map an algorithm to the CUDA programming environment, we should identify data-parallel portions of the application, and isolate them as CUDA kernels. According to the algorithm description below, texture extraction operations are executed iteratively and independently on different elements of the dataset. The target dataset is stored in device Texture memory.

```
1: Down-sample images by applying Gaussian smoothing
2: Initialize V_s(x_s) to zero vector
3: repeat
4:    for each pixel, i do
5:       Compute local histogram
6:       Compute geometric moments
7:       Compute Canny edge strength
8:    end for
9: until all resolutions are done
```

The GPU hardware allows for fast performance of pixel-wise operations. To our delight, the major operations of computing geometric moments of local histograms are naturally pixel-wise and spatially independent. Meanwhile, many steps in Canny edge detection procedure, such as gradient finding, convolution, and non-maxima suppression can be performed in parallel, too.

For our implementation, we give one thread to every pixel. Algorithm 1 runs on the CPU while algorithm 2 runs on the GPU.

Algorithm 1
CUDA_Feature(Image Data V, Width, Height)

```
1: Create point array A and bind it to two dimensional texture memory T
2: Allocate Global memory F for saving GPU results
3: for every point in parallel do
4: invoke CUDA_Feature_Kernal(F) on the grid
5: end for
6: Send results to CPU memory
```

Algorithm 2 CUDA_Feature_Kernal(F)

```
1: tid←getThreadID, bid←getBlockID
2: Compute point offset←bid*width+tid
3: for all the neighbors in the window of point [bid,tid] do
4: Compute four features of point and put into temp
5: end for
6: F[offset]←temp
```

Usually, a completed version of CUDA implementation includes host code in CPU and kernel code in GPU, corresponding to algorithm 1 and algorithm 2, respectively. In practice, the host code is responsible for realizing IO interface and procedure control, which is very similar with ordinary standard C code. The major difference is that host code might use some CUDA APIs to communicate with GPU and invoke the computing procedure. In our implementation, testing image is firstly read into CPU memory, and bind to the texture memory, which could visit the image data by 2 dimensional index. Meanwhile, the attached cache of texture memory tremendously reduces IO cost when accessing the pixel data under read mode. When data is transferred to GPU, kernel code is invoked for parallel computing. The most important action in kernel code is to determine the thread allocation. In our implementation, in order to optimize the thread parallelism in the target GPU device, the original size of testing images is 512×512. Then, 512 blocks are used, each of which has 512 threads.

IV. EXPERIMENTAL RESULTS

A. Lesion Detection Results

In practice, we have collected 147 normal individuals' brain HRCT volumes for statistical texture atlas construction, and the slice thickness varies from 1mm to 3mm. With the help of radiologists, one single slice is picked out at the

similar location from each volume. After rigid and deformable registration, these 2D slices have been transformed into a normalized reference coordinate. Table I shows the registration accuracy, in terms of average correlation coefficient, average signal-to-noise ratio, and average mean squared error. Fig4 shows the segmentation results of three typical intracranial space-occupying lesions, which are generated according to the Mahalanobis distances of the corresponding pixels between atlas and sample images.

TABLE I REGISTRATION ACCURACY

	$\overline{R_{cc}}$	$\overline{SNR}$	$\overline{MSE}$
ORIGINAL	0.8243	1.5074	3.94E+005
REGISTERED	0.9637	12.4518	40.5371

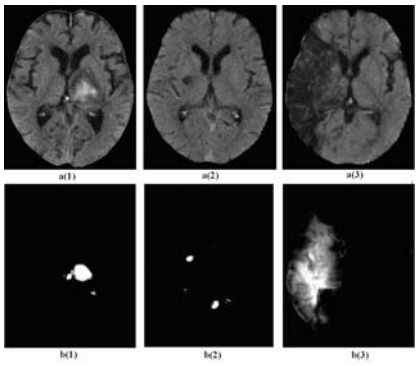

Fig.4. Lesion detection results

B. CUDA performance of Texture Extraction

Our implementation is running on a 1.2GHz Pentium 4 Dual Core, 1 GB RAM with Nvidia 8800GTX graphics chip with 768MB on board, and using CUDA 1.1 under Microsoft Windows XP SP2. For testing, we compared the performance of texture extraction procedure of all the 147 sample images on GPU and CPU. GPU implementation is written in CUDA, while CPU version is written in standard C. Performance results are shown in Table II. Detailed time consumption is profiled in Fig. 5. Comparisons between CUDA and standard C implementation demonstrate a significant speedup in GPU against straight forward CPU functions.

TABLE II CUDA SPEEDUP

RESOLUTION	GPU(S)	CPU(S)	SPEED UP
512×512	470.4	10772.16	22.9
256×256	264.6	5662.4	21.4
128×128	161.7	3411.9	21.1
TOTAL SPEEDUP	22.13		

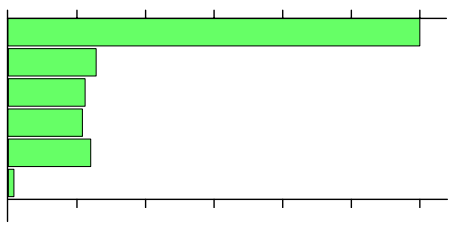

Fig.5. CUDA profile output

V. CONCLUSIONS AND DISCUSSIONS

We present a GPU-based accelerated detection method for intracranial space-occupying lesions, using statistical texture atlas. In our approach, local intensity histogram distribution and Canny edge information are selected to characterize statistical texture patterns of brain anatomy. Experimental results support that Mahalanobis distance in feature space effectively depicts the difference between the cross-region mixed density lesions and normal tissues. Further more, we have implemented a parallel computing scheme on CUDA to accelerate the texture extraction procedure, which makes it more applicable for large scale clinical applications. However, it should be noted that this study has examined only 2D CT images, and further implementations on 3D volumes will be summarized in our next study.

ACKNOWLEDGMENT

The authors appreciate the support of Nature and Science Foundation of China (project No. 60771007) and Natural Science Foundation of Anhui Education Department (project No. 2006KJ097A).

REFERENCES

[1] J. E. Bradley, B. Brian, "Computer-aided detection and diagnosis at the start of the third millennium", Journal of Digital Imaging, vol. 15, no. 2, pp. 59-68, 2002.

[2] A. L. Gholipour, N. Kehtarnavaz, R. Briggs, et al, "Brain function localization: A survey of image registration techniques", IEEE Trans on Medical Imaging, vol. 26, no. 4, pp. 427-451, 2007.

[3] C. Davatzikos, A. Genc, D. Xu, et al, "Voxel-based morphometry using the RAVENS maps: Methods and validation using simulated longitudinal atrophy", NeuroImage, vol. 14, no. 6, pp.1361-1369, 2001.

[4] J. Ashburner, K. Friston, "Voxel-based morphometry- The methods", NeuroImage, vol. 1, no. 11, pp. 805-821, 2000.

[5] D. Shen, C. Davatzikos, "Very high-Resolution morphometry using mass-preserving deformations and HAMMER elastic registration", NeuroImage, vol. 18, no. 1, pp. 28-41, 2003.

[6] K. Friston, A. P. Holmes, J. P. Poline, et al, "Statistical parametric maps in functional imaging: A general linear approach", Human Brain Mapping, vol. 2, no. 4, pp. 189-210, 1995.

[7] S. G. Metha, J. Thomas, Y. Trivedi, et al, "Evaluation of voxel-based morphometry for focal lesion detection in individuals", NeuroImage, vol. 20, no. 3, pp. 1438-1454, 2003.

[8] J. P. Thirion, "Image matching as a diffusion process: an analogy with Maxwell's demons", Medical Image Analysis, vol. 2, no. 3, pp. 243–260, 1998.

[9] K. A. Ganser, H. Dickhaus, R. Metzner, et al, "A deformable digital brain atlas system according to Talairach and Tournoux", Medical Image Analysis, vol. 8, no. 1, pp. 3-22, 2004.

[10] X. Zhong, D. Shen, C. Davatzikos, "Determining correspondence in 3D MR brain images using attribute vectors as morphological signatures of voxels", IEEE Trans on Medical Imaging, vol. 23, no. 10, pp. 1276-1291, 2004.

[11] Pawan Harish, P. J. Narayanan, "Accelerating large graph algorithms on the GPU using CUDA", in IEEE International Conference on High Performance Computing, Springer, pp.197-208, December 2007.

[12] S.S. Stone, J.P. Haldar, S.C. Tsao, et al, "Accelerating advanced MRI reconstructions on GPUs", Journal of Parallel and Distrivuted Computing, vol. 68, pp. 1307-1318, 2008.