GPU-accelerated, Gradient-free MI Deformable Registration for Atlas-based MR Brain Image Segmentation

Xiao Han, Lyndon S. Hibbard, Virgil Willcut
CMS Software, Elekta Inc.
13723 Riverport Drive, Maryland Heights, MO 63043, USA
{Xiao.Han,Lyn.Hibbard,Virgil.Willcut}@cmsrtp.com

Abstract

Brain structure segmentation is an important task in many neuroscience and clinical applications. In this paper, we introduce a novel MI-based dense deformable registration method and apply it to the automatic segmentation of detailed brain structures. Together with a multiple atlas fusion strategy, very accurate segmentation results were obtained, as compared with other reported methods in the literature. To make multi-atlas segmentation computationally feasible, we also propose to take advantage of the recent advancements in GPU technology and introduce a GPU-based implementation of the proposed registration method. With GPU acceleration it takes less than 8 minutes to compile a multi-atlas segmentation for each subject even with as many as 17 atlases, which demonstrates that the use of GPUs can greatly facilitate the application of such atlas-based segmentation methods in practice.

1. Introduction

Segmentation of brain structures from magnetic resonance (MR) brain images is an important task in neuroscience studies and can also benefit many clinical applications such as diagnosis of neurodegenerative and psychiatric disorders, treatment evaluation, and surgical planning. Manual delineation, although still the common standard for high quality segmentation, is tedious, time-consuming, and suffers from large intra- and inter-rater variability.

A large variety of methods have been proposed in the literature for automated segmentation of brain structures (cf. [22, 9, 21, 8, 7, 2, 1, 20, 3, 14] and references therein), among which atlas-based segmentation has become more and more popular especially for the simultaneous segmentation of a large set of structures. In atlas-based segmentation, a novel subject image is first aligned to an atlas image – an image that has the structures of interest already labeled – through image registration and the atlas labels are then mapped to the subject image using the computed image transformation. Due to its formulation, the two most critical components of such a method are the atlas registration method and the atlas selection or construction strategy. Different atlas-based methods most often differ by the registration methods they apply and the type of atlases used.

In this paper, we first introduce a new gradient-free mutual information (MI) deformable registration method, which serves as the basis of our atlas-based brain segmentation framework. MI-based deformable registration has been a popular registration technique due to its ability to handle both mono- and multi-modality images (cf. [11]). In this work, we use a non-parametric deformation field model as in [5, 12] where the registration transformation is directly modeled as a dense vector field with one displacement vector at each image voxel. But unlike [5, 12] where gradient-descent types of approaches have been used to estimate the optimal deformation field, we propose in this work an explicit local search as the optimization method. The explicit neighbor search avoids the difficulty in gradient-descent methods for determining the optimal time step size. It also eliminates the need to compute derivatives of either the images or the image intensity distributions, which makes it easily parallelizible and thus better suited for GPU implementation.We note that explicit search strategies have been applied in block-matching based image registration methods (cf. [3, 17]) and an explicit-search-like discrete sampling strategy has also been adopted in a recently proposed registration method that formulates image registration as global discrete labeling [6]. Surprisingly, the simple explicit local search strategy has not yet been applied for dense deformable registration according to our knowledge. We show through our atlas-based brain segmentation results that this strategy actually works very effectively in practice.

Atlas selection or construction is also a key component that highly determines the accuracy of atlas-based segmentation. Four different atlas-selection strategies are typically found in the literature (cf. [13]). One approach that has attracted large attention recently is to apply multiple atlases to generate multiple segmentations of the same image, which are then combined systematically in a multi-classifier framework to get an accurate final segmentation. A major advantage of this approach is that any image registration method can potentially be directly applied in this framework (unlike statistical or probabilistic atlases that typically require specially designed atlas registration methods). A limitation that may prevent their wide-spread use is that the computation time is directly proportional to the number of atlases used and can be prohibitive in practice. In this work, we propose to address this problem by taking advantage of the recent advancements in GPU technology and introduce a GPU-based implementation of our MI deformable registration method. As will be demonstrated later, the GPU acceleration can easily offer a speed-up of more than 20× and thus allowing segmentation using multiple atlases be finished within the same time-frame of a single atlas segmentation on high-speed CPU.

In the following, we first describe our MI deformable registration method and the new explicit local search optimization strategy. We then introduce its GPU implementation. Finally, we present validation results using real MR brain images from a public MR brain image database and compare with several other methods that were tested on the same set of data.

2. Gradient-free, Dense MI Deformable Registration

2.1. MI Similarity Metric and Spatial Transformation Model

Let I denote an atlas image and J a novel subject image to be segmented. The goal of atlas registration is to find a spatial transformation T that maps each point in image I to its corresponding point in J such that structure labels associated with each atlas image point can be transferred to the subject image. Mathematically, the atlas registration problem can be formulated as the following optimization problem:

$$T_{\text{opt}} = \text{argmax}_{T \in \Omega_T} S(I, J(T)), \tag{1}$$

where T_{opt} denotes the optimal transformation, S is a selected similarity metric, and Ω_T is the space of all possible transformations.

We adopt MI as the image similarity measure due to its ability to handle inter-subject intensity and contrast variations that are common in MR imaging. The MI of two images I and J measures the degree of dependence between I and J as the distance between their true joint intensity distribution $p_{IJ}(i,j)$ and the distribution when complete independence is assumed, as given by (cf. [18, 11]):

$$\begin{aligned} \text{MI}(I,J) &= H(I) + H(J) - H(I,J) & (2)\\ &= \sum_{(i,j)} p(i,j) \log \left[\frac{p_{I,J}(i,j)}{p_I(i) p_J(j)} \right], & (3) \end{aligned}$$

where $p_I(\cdot)$ and $p_J(\cdot)$ denote the marginal probabilistic intensity distributions of images I and J, and $H(\cdot)$ and $H(\cdot,\cdot)$ represent the marginal and joint entropies, respectively. The summation in Eq. (3) is computed over the set of image intensity value pairs.

Considering the discrete nature of digital images, it is shown in [12] that Eq. (3) can be approximated by:

$$\begin{aligned} \text{MI}(I,J) &\approx \sum_{(i,j)} \frac{N_{i,j}}{N} \log \left[\frac{p_{I,J}(i,j)}{p_I(i) p_J(j)} \right] & (4)\\ &= \frac{1}{N} \sum_{\mathbf{x}} \log \left[\frac{p_{I,J}(I(\mathbf{x}), J(T(\mathbf{x})))}{p_I(I(\mathbf{x})) p_J(J(T(\mathbf{x})))} \right], & (5) \end{aligned}$$

where $N_{(i,j)}$ is the number of occurrences of the intensity pair (i,j), and N is the total number of intensity pairs within the overlapped region of the two images. The final summation in Eq. (5) is now taken over the spatial image coordinates instead of intensity values.

Many different models of the transformation T has been proposed in the literature for MI-based deformable registration (cf. [11]). A popular choice is to apply parameterized models such as B-splines or radial basis functions, which allows direct control of the number of degrees of freedom that describe the transformation. But as pointed out by [16] (and also from our own experience), the computational speed of such parametric models is usually slow, especially for inter-subject deformable registration where high degrees of freedom is necessary to account for highly localized structural differences.

In this work, we adopt a non-parametric model where the image transformation is modeled directly by a vectorial displacement field $\mathbf{U}$, such that $T(\mathbf{x}) = \mathbf{x} + \mathbf{U}(\mathbf{x})$ for every image point (cf. [16]). Such a dense displacement field model has been used in [5] in a variational framework with a gradient-descent optimization strategy. Such a dense deformation field model is more flexible than a parameterized model, and can take into account all image information and allow detecting fine anatomical differences. In practice, extra regularization constraints usually need be imposed to get anatomically meaningful correspondences. But instead of adding an explicit regularization energy term to Eq. (1), we adopt a so-called "pair-and-smooth" strategy (cf. [4]) that alternates between optimizing the matching of the atlas and subject images and regularization of the estimated deformation field. In particular, the topology correctness and regularity of the computed deformation field is enforced through proper design of the update scheme and by the application of spatial smoothing filters, as will become clear in the next section. We would like to emphasize that, unlike the intra-subject case, there is no *true* physical model for the transformation between two different subjects.

2.2. Optimization Scheme

The optimal deformation field is computed in an iterative fashion. At each iteration step, the goal is to first find a correction field $\mathbf{u}$ that maximizes the image similarity criterion given the current estimate of the deformation field $\mathbf{U}$, and then smoothing or regularization can be applied if necessary. Unlike the additive correction scheme used in [5, 12], we adopt a compositive update strategy as in [16]. In particular, let $\mathbf{Id}$ denote the identity transformation and note that $T = \mathbf{Id} + \mathbf{U}$, then the transformation at the $(n+1)$-th iteration is given by

$$\begin{aligned} \mathbf{Id} + \mathbf{U}^{n+1} &= (\mathbf{Id} + \mathbf{U}^n) \circ (\mathbf{Id} + \mathbf{u}^n) \quad (6) \\ &= \mathbf{Id} + \mathbf{U}^n \circ (\mathbf{Id} + \mathbf{u}^n) + \mathbf{u}^n, \quad (7) \end{aligned}$$

where "$\circ$" denotes transformation composition. Thus, considering the deformation field $\mathbf{U}$, we end up with the following update equation

$$\mathbf{U}^{n+1} = \mathbf{U}^n \circ (\mathbf{Id} + \mathbf{u}^n) + \mathbf{u}^n. \quad (8)$$

One major advantage of the compositive scheme is that the total displacement $\mathbf{Id} + \mathbf{U}$ is guaranteed to be a homeomorphism if the correction $\mathbf{Id} + \mathbf{u}$ at each step is a homeomorphism. As pointed out in [16], guaranteeing that $\mathbf{Id} + \mathbf{u}$ is a homeomorphism is fairly easy: it suffices to limit the magnitude of $\mathbf{u}$ to half of the image voxel size.

The goal now is to find the correction field $\mathbf{u}^n$ that maximizes $\mathrm{MI}(I, J \circ ((\mathbf{Id} + \mathbf{U}^n) \circ (\mathbf{Id} + \mathbf{u}^n)))$. To simplify further, let J^n denote the transformed subject image by applying the current deformation field estimate $\mathbf{U}^n$:

$$J^n(\mathbf{x}) = (J \circ (\mathbf{Id} + U^n))(\mathbf{x}) = J(\mathbf{x} + \mathbf{U}^n(\mathbf{x})), \quad (9)$$

then the optimal $\mathbf{u}^n$ should maximize

$$\begin{aligned} \mathrm{MI}(I, J^n &\circ (\mathbf{Id} + \mathbf{u}^n)) \\ &= \frac{1}{N} \sum_{\mathbf{x}} \log \left[\frac{p_{I,J^n}(I(\mathbf{x}), J^n(\mathbf{x} + \mathbf{u}^n(\mathbf{x})))}{p_I(I(\mathbf{x})) p_{J^n}(J^n(\mathbf{x} + \mathbf{u}^n(\mathbf{x})))} \right] \\ &\triangleq \frac{1}{N} \sum_{\mathbf{x}} \mathrm{SMI}(I(\mathbf{x}), J^n(\mathbf{x} + \mathbf{u}^n(\mathbf{x}))), \quad (10) \end{aligned}$$

where SMI can be called a point-wise MI, a notation initially introduced in [12].

So far in the literature, the update $\mathbf{u}^n$ is always approximated by computing the gradient or first variation of the metric MI (or SMI) at each voxel location. A major difficulty of a gradient-descent approach is in the determination of the optimal time step size or the magnitude of the update. Although line-search is possible, it is rarely used in practice due to its large computation cost. Another popular choice is to choose a global time step such that the maximum update magnitude is less than a pre-defined threshold. Such a choice usually leads to slow convergence due to the large dynamic range across the different components (one for each image point) of the high-dimensional gradient vector. In addition, there is little correlation expected between the displacements at points far-away from each other. Hence, a global time step is hardly an optimal choice.

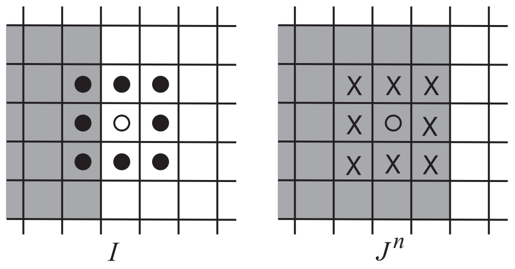

Figure 1. A 2D illustration of explicit local search. The open circles indicate the image point under consideration and its current correspondence in the other image. The crosses indicate the neighbors involved in the forward search. The solid circles indicate the neighbors involved in the backward search.

From Eq. (10), it is clear that the correction field $\mathbf{u}^n$ at different image locations is largely decoupled. To maximize $\mathrm{MI}(I, J^n \circ (\mathbf{Id} + \mathbf{u}^n))$, we can try to maximize each of the summation terms (SMIs) individually. We hence propose to use an explicit local search to find the optimal $\mathbf{u}^n(\mathbf{x})$ at each image location $\mathbf{x}$. In particular, as illustrated in Fig. 1, at each atlas location $\mathbf{x}$ (as indicated by the open-circle in the figure), we consider the possible matching of $\mathbf{x}$ to each of its immediate neighbors (including $\mathbf{x}$ itself, which implies a zero update) in image J^n. We consider the 26-neighborhood in 3D. For each pair of potential matches, we evaluate the SMI value and find the pair $(\mathbf{x}, \mathbf{x}')$ that gives the largest value. The local update is then computed as $\mathbf{u}^n(\mathbf{x}) = \mathbf{x}' - \mathbf{x}$. Since the comparison is performed at exact image voxel locations, no interpolation is needed assuming that J^n has already been computed.

We note that the local update scheme ignores the dependency of the joint image intensity distribution on the update field, and hence the computed update is sub-optimal in theory. But since we re-estimate the intensity distributions before each iteration and since the update is only local, such approximation works well in practice, as verified by our atlas segmentation results.

The above update scheme is non-symmetric with respect to the two images. For example, if $\mathbf{x}$ is near an object boundary in image I but within a homogenous region of J^n as illustrated in Fig. 1, the local neighbor search described above will not return a useful match. To improve the convergence property of the algorithm, we perform a *backward* search as well. In this pass, we fix the voxel $\mathbf{x}$ in image J^n and search all its neighbors (including $\mathbf{x}$ itself) in image I, and found the pair that produces the largest score $\mathrm{SMI}_{\mathrm{backward}}(I(\mathbf{x}-\mathbf{u}(\mathbf{x})), J(\mathbf{x}))$. Finally, we take the update as

$$\mathbf{u}^n(\mathbf{x}) = \mathrm{argmax}\left(\mathrm{SMI}_{\mathrm{forward}}(\mathbf{u}(\mathbf{x})), \mathrm{SMI}_{\mathrm{backward}}(-\mathbf{u}(\mathbf{x}))\right).$$

The backward search is equivalent to finding the optimal inverse update since $(\mathbf{Id}+\mathbf{u}^n)^{-1} \approx \mathbf{Id}-\mathbf{u}^n$. It has been also shown by others (e.g. [12]) that such a symmetric "force" computation helps improve the capture range and convergence of a deformable registration method.

To make sure that the update field is invertible, we can scale it so that its magnitude is less than half the voxel size. After the update field is found, we smooth it with a spatial Gaussian filter. In all experiments presented later, we used a Gaussian filter with the kernel size equal to the size of an image voxel in each dimension. We then update the total field according to Eq. (8), and regularize it using another Gaussian filter. The iteration is repeated until a user-specified number of steps or until the global MI stops increasing. The overall method can be summarized in the following algorithm.

Algorithm 1 (Gradient-free Dense MI Deformable Registration):

```
0. Given an initial estimate $\mathbf{U}^0$, set $n = 0$
1. Compute deformed subject image $J^n = J \circ (\mathbf{Id}+\mathbf{U}^n)$
2. Update the marginal and joint intensity distributions in
   Eq. (5) using normalized image intensity histograms
3. Compute $\mathbf{u}^n$ using the symmetric explicit local search
   scheme described in Section 2.2
4. Regularize $\mathbf{u}^n$ with a spatial Gaussian filter
5. Compute $\mathbf{U}^{(n+1)}$ using Eq. (8)
6. Regularize $\mathbf{U}^{(n+1)}$ with another Gaussian filter
7. If converged, stop; if not, set $n = n + 1$, goto Step 1.
```

To improve computational efficiency, the traditional multi-resolution scheme is applied in all experiments presented later, which runs the above algorithm in a coarse-to-fine fashion using a pyramid representation of both the atlas and subject images. Since the algorithm involves no computation of image gradients or derivatives of the intensity distributions, it is extremely easy to parallelize. In the next section, we will describe a CUDA implementation of the above algorithm on NVIDIA graphic cards.

3. Implementation on GPU with CUDA

3.1. An Overview of CUDA

In recent years, commodity graphics processors (GPUs) have rapidly evolved into an attractive hardware platform for general purpose computations due to their extremely high floating-point processing performance and their comparatively low cost. Many applications that are not yet able to achieve satisfactory performance on CPUs can directly benefit from the massive parallelism provided by such devices. But until recently, GPU programming required the use of graphics APIs such as OpenGL or high-level languages layered on top of them, hence was non-intuitive and cumbersome. NVIDIA's *C*ompute *U*nified *D*evice *A*rchitecture (CUDA) aims to solve these problems by introducing a new hardware and software architecture, allowing a much more flexible programming model and allowing developers to implement a wider variety of data-parallel applications. The details of the CUDA programming model are available in NVIDIA's CUDA Programming Guide [10]. Here we only provide a brief overview.

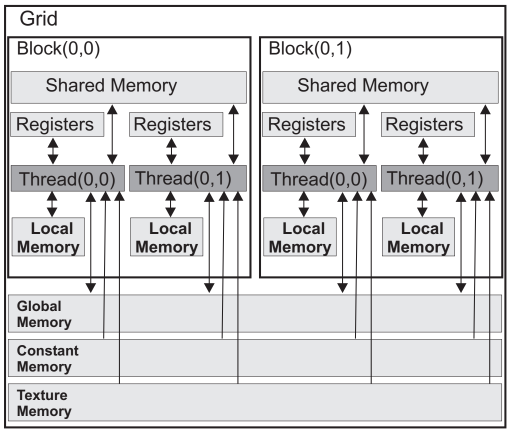

Figure 2. CUDA programming model and memory hierarchy [10].

Fig. 2 illustrates how the GPU appears to the developer under CUDA. The computation is distributed in a grid of thread blocks and executed on the GPU (called *device* in CUDA). All blocks contain the same number of threads that execute a program on the device in parallel, which is known as the *kernel function* or *kernel*. The number of threads within a block that can execute simultaneously is limited by the number of cores in a multiprocessor of the GPU, which are referred to as a *warp*. Threads in a block can communicate with each other via a small but fast per-block shared memory space (16 KB) and be synchronized at developer specified barriers. There is, however, no means of synchronization among threads in different blocks.

Each block in a grid has a unique two-dimensional block ID and each thread has a three-dimensional ID that is unique within one block. The unique ID allows each thread to be identified with an element or a piece (e.g., a single row or a single column) of the data for the thread to process. CUDA allows developers to specify the grid and block dimensions at run-time (the so-called *execution configuration*) to effectively distribute the computation load across all processors and to hide memory access latency. While tools are available from NVIDIA for estimating the optimal execution configuration, it is typically necessary to fine tune the configuration experimentally for each kernel function.

CUDA threads may access data from multiple memory spaces during their execution. Aside from the per-block shared memory mentioned above, each thread has a private local memory. Finally, all threads have access to the same global memory. There are also two additional read-only memory spaces accessible by all threads: the constant and texture memory spaces. The global, constant, and texture memory spaces are optimized for different memory usage, which can be leveraged to improve memory performance. In addition, CUDA supports a subset of the texturing hardware capabilities that GPU typically uses for graphics. For example, texture memory access supports hardware-accelerated linear interpolation between texels, which is very useful for implementing image warping.

3.2. CUDA Implementation of the MI Method

The proposed deformable registration method is ideal for GPU acceleration due to the high level of parallelism exists between the processing of individual image voxels, but care need be taken to achieve the best efficiency. Since data transfers between the CPU and the GPU are much slower than transfers within the GPU memory, the efficiency of an iterative algorithm can be improved if all computation can be performed on the GPU. Considering the limited memory space of available graphics cards, we design our GPU program to compute Algorithm 1 on one level at a time for our multi-resolution atlas registration. At the beginning of each resolution level, the down-sampled images and the initial estimate of the deformation field are copied over to the GPU memory, and the iterative updates of Algorithm 1 are then computed on the GPU until convergence. The main program then up-samples the field to the next resolution level, and re-initializes the GPU for the next level of computation. The task remains in the implementation of Algorithm 1 using CUDA kernel functions, which we divide into 4 major groups as summarized in the next.

A. Estimate joint and marginal intensity distributions. We use image histograms (joint and marginal) to estimate the intensity distribution functions needed in evaluating the MI in Eq. (5) or the SMI in Eq. (10). Although trivial to compute on CPU, histograms had been quite difficult to compute efficiently on GPU due to the required data-dependent memory access and update [15]. A couple of efficient CUDA-based histogram calculation methods have been recently proposed and we use a method introduced in [15] in this work. In this method, each warp of a thread block maintains a subset of histogram bins and uses the fast per-block shared memory to store and update the sub-histogram. In the end, the per-block sub-histograms are accumulated and written to the global memory using atomic global memory operations. Since the per-block share memory has very limited size (16KB), it is necessary to run the algorithm multiple times to cover the entire bin range of a large histogram (as typically needed if the number of histogram bins is greater than 1024). As also pointed out in [15], computing a 2D joint histogram with $B_1 \times B_2$ bins can be formulated as computing a 1D histogram of $B = B_1 \times B_2$ number of bins by properly concatenating the data. After the joint histogram is computed, the marginal histograms are computed using a parallel reduction strategy.

B. Compute image warping. Image warping is needed both to compute the deformed subject image J^n as indicated by Eq. (9) and to compute the deformation field update as given by Eq. (8). For the vector-valued deformation field $\mathbf{U}$, the warping $\mathbf{U}(\mathbf{x}+\mathbf{u}(\mathbf{x}))$ can simply be computed for each of its three scalar components separately. As mentioned earlier, the CUDA texture memory is faster than global memory and provides hardware-accelerated linear interpolation capability. In addition, the 2.0 version of CUDA added direct support for 3D textures, which makes using CUDA for 3D image warping very efficient. For example, to compute the warped image J^n, we first create a texture reference object, $textJ$, and bind it to the global 3D image array that stores the original image J. Then each thread can compute the value of J^n at one image point $\mathbf{x}(=(x,y,z))$ using a single texture fetching:

$$J^n(\mathbf{x}) = texFetch3D(texJ, x{+}U(\mathbf{x}), y{+}V(\mathbf{x}), z{+}W(\mathbf{x})),$$

where (U,V,W) denotes the three scalar components of the deformation field $\mathbf{U}^n$.

C. Gaussian smoothing. We apply 3D Gaussian smoothing filters to regularize the update and the total deformation field, as indicated in Steps 4 and 6 of Algorithm 1. For the vector-valued deformation field, the Gaussian smoothing is performed for each of the three scalar components separately. We adopted the strategy of the design of a 2D recursive Gaussian filtering in the CUDA development kit and extended it to 3D. In particular, since the Gaussian filtering is separable, three one-dimensional Gaussian filtering kernels are designed that apply Gaussian smoothing along each coordinate direction separately. Taken for example the 1D filtering along the row direction, each thread is identified with one row of the image and applies the recursive Gaussian filtering to it.

Special care is needed when processing along the column direction (the fastest changing index direction for a stored image in the GPU global memory). If each thread is identified with one column of the image, then consecutive threads within a block will try to address non-continuous memory locations at the same time when they are executed in parallel. This "non-coalesced" memory access has a large latency and is thus slow. To improve efficiency, the volume is first transposed by switching the row and column indexes, and then the row-filtering kernel can be applied, after which the 3D volume is transposed back. The volume transpose is performed with another kernel function.

D. Compute the update field. Computing the update field at each voxel requires accessing image values at neighboring voxels. Thus, there is an overlap between data needed at neighboring threads. To make use of this overlap and reduce the amount of global memory access, we split the image domain into small 3D sub-blocks, corresponding to the block-wise organization of threads in the GPU architecture. At the start of the kernel function, each thread block loads into the shared memory the voxels to be processed and the neighboring voxels of the whole block. Given that the shared memory has a limited size of 16 KB, we currently use a block size of $16 \times 10 \times 5$ (this choice may not be optimal and may be optimized in the future). Thus, after the data loading, seven planes (five plus the top and bottom neighbors) of data from both the image I and the deformed image J^n are kept in the shared memory. Each thread can then perform the explicit forward and backward searches and find the optimal displacement at the voxel location identified with the thread.

4. Experiments and Results

4.1. Data and Experiment Design

For experimental data, we use the MR brain images and their manual segmentation provided by the Center for Morphometric Analysis at Massachusetts General Hospital, which are publically available at *http://www.cma.mgh.harvard.edu/ibsr/*. This public data set allows us to run comparisons with several other recently reported brain segmentation methods that were tested on the same set of data. This data set has 18 $256 \times 256 \times 128$ brain images with a resolution around $0.93 \times 0.93 \times 1.5\text{mm}^3$. The subject age ranges from 7- to 71- years old. These images were "positionally-normalized" into the Talairach orientation (rotation only) and processed with intensity bias field correction. The manual segmentation for each subject contains near 70 structures (left and right hemispheres are counted separately) covering both the cerebrum and the cerebellum. In this study, we focus on the brain-stem (B) and the following subcortical structures: lateral ventricles (LV), caudate (Cau), thalamus (Th), putamen (Put), pallidum (Pal), hippocampus (H) and amygdala (Amy). The shortened notations will be used later for figure labels.

We use a leave-one-out strategy to evaluate our atlas-based segmentation method: for each subject, the remaining 17 subjects are considered as possible atlases. Before running the MI deformable registration to align a test subject with an atlas, we run a quick linear registration step first to correct for gross position and size differences between them. The 6-degree linear transformation is computed by maximizing the global MI between the two images using a multi-resolution stochastic gradient-descent optimization algorithm [18]. The quick linear registration takes only 3 seconds on our desktop computer, and thus no GPU-acceleration is needed.

As described earlier, our multi-atlas segmentation strategy performs first an atlas-based segmentation for each subject using each of the candidate atlases separately. The final segmentation is then computed using the STAPLE algorithm [19]. In the results presented later, the STAPLE algorithm was applied for each structure separately. Note that the STAPLE algorithm actually provides a probabilistic estimation of each voxel belonging to the structure under consideration. We typically threshold the probability map at 0.5 to get a binary structure segmentation. At voxels where overlapping occurs in the final label map between adjacent structures due to the independent STAPLE computation, we re-assign each voxel to the structure label that gives the highest probability as produced by the STAPLE algorithm. For comparison, we also implemented a single "optimal" atlas segmentation strategy where each subject image is segmented using a single atlas that is most "similar" to it. We used the MI value after the linear registration step as the similarity measure. Segmentation accuracy is evaluated against the available manual segmentation using the widely used *Dice* similarity coefficients (cf. [8]).

For comparison between CPU and GPU, a multi-threaded version of the gradient-free MI deformable registration method was also implemented on CPU with the help of the ITK software libraries (www.itk.org). The CPU implementation was run on a HP xw8400 Workstation with an Intel Xeon Quad-core 2.66 GHz processor and 4 GB memory. The GPU code was run on a NVIDIA GeForce GTX 280 graphics card installed on the same computer.

4.2. Results

Fig. 3 demonstrates the deformable registration and atlas-based segmentation using two of the test images (subjects #7 and #8 from the IBSR database). Fig. 3(a) shows a 2D cross-section of the atlas image and Fig. 3(b) the subject. Fig. 3(c) shows the subject image mapped to the atlas using the computed deformable registration. Fig. 3(d) displays the atlas structures overlaid on the original subject image. As can be seen, the ventricles and putamen are clearly mismatched. Fig. 3(e) shows the transformed atlas structures by applying the registration result, which gives the auto-segmentation of the subject based on the given atlas. Fig. 3(f) is a magnified view of the result together with the manual segmentation overlaid as the yellow curves. From Fig. 3(f), it can be seen that the auto-segmentation result matches the manual segmentation very well.

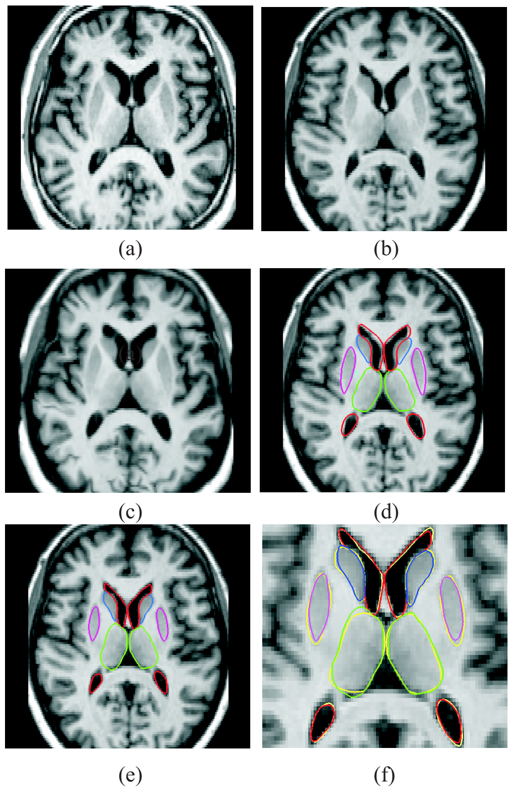

Figure 3. Illustration of the deformable registration and atlas-based segmentation results (see text for details).

The Dice coefficient results are summarized in the box plots of Fig. 4, which shows the mean and std of the Dice values for each of the 10 structures computed over the 18 subjects. Fig. 4 also compares the multi-atlas results against the single "adaptive" atlas. Due to lack of space, the Dice coefficients for the left and right parts of the same structure are averaged together for each subject. The use of multiple atlases largely improves the accuracy over using a single atlas: five of the eight structures have a mean Dice coefficients well above 0.8 when multiple atlases are used and all structures are above 0.7. Using paired-*t* tests, it was also found that the improvements are statistically significant at the 0.01 level for all structures except for the brain-stem.

Table 1 compares the segmentation accuracy of our atlas-based segmentation method with several other recently re-

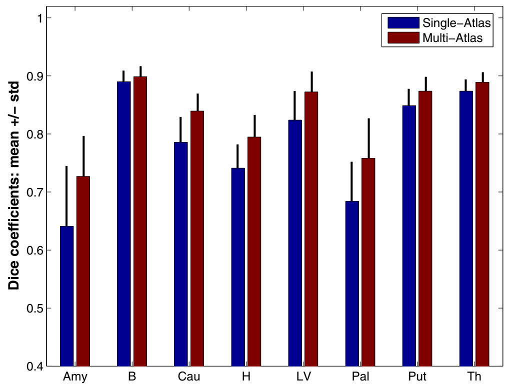

Figure 4. Comparison of two atlas-selection strategies. Bar plots show average Dice coefficients over 18 subjects for each structure, and error bars indicate one standard deviation.

Table 1. Comparison with other methods on the IBSR data sets.

Method	Amy	B	Cau	H	LV	Pal	Put	Th
Ours – Multi-atlas	0.73	0.90	0.84	0.79	0.87	0.76	0.87	0.89
Akselrod-Ballin et al. [1]	0.63	0.84	0.80	0.69	-	0.74	0.79	0.84
Bhattacharjee et al. [3]	0.60	-	0.75	0.66	0.64	0.76	0.82	0.82
Bazin & Pham [2]	-	-	0.80	-	0.83	-	0.76	0.78
Gouttard et al. [7]	0.64	-	0.76	0.66	0.85	0.71	0.78	-
Sdika [14]	-	-	0.79	-	0.86	-	0.81	-
Wu et al. [21]	0.71	-	0.86	0.79	-	0.79	0.86	0.88
Wu & Chung [20]	-	-	0.78	-	-	-	0.76	0.77
Zhou et al. [22]	0.65	-	0.81	0.71	-	-	0.83	0.84

ported methods tested on the same set of IBSR data[1]. Only mean Dice values are reported in the table. From Table 1, it can be seen that our method with the multi-atlas scheme compares favorably with all the other methods. The only method that offers similar accuracy to ours is the work by Wu et al. [21], which also applied a multi-atlas-based segmentation method. But instead of using STAPLE to fuse the multiple segmentation results, Wu et al. [21] proposed to adaptively select the "best" atlas for each structure separately based on a local registration accuracy estimation. Computation-wise, the method still requires a full image registration of each new subject to every candidate atlas. The deformable registration method used in [21] is totally

[1]Some of the methods reported results in terms of *overlap ratio*. We have converted them to Dice values using the formula in [9].

different from what we proposed in this paper. In addition, the method of [21] performs extra pre-processing steps including skull-stripping and intensity normalization.

For computation speed comparison between GPU and CPU, it was found that the deformable registration takes about 8 minutes for each pair of images on the above-mentioned desktop computer, whereas it only takes about 19 seconds for the GPU version. Thus, the GPU offers a speed-up of $25\times$ comparing with the quad-core CPU. As a result, it takes less time to run multi-atlas segmentation with 17 atlases on the GPU than running a single-atlas segmentation on the CPU.

5. Conclusion

In this paper, we have proposed an atlas-based brain segmentation method based on a novel gradient-free MI dense deformable registration. We also showed that using multiple atlases is a good approach to achieve high segmentation accuracy. Test results on a set of publically available data showed that the proposed method compares favorably with other recently reported methods. We also proposed making use of GPU to make multi-atlas segmentation a more feasible approach in practice. With GPU acceleration, segmentation using 17 atlases takes less than 8 minutes, whereas it would have take more than 2 hours even on a fast desktop computer. In addition, the GeForce GTX280 card used in this study fits into any modern desktop computer and currently only costs \$350. Thus, GPU is a very low cost way to achieve such high performance comparing against a computer cluster. Future work will investigate the dependency of segmentation accuracy with the number of atlases used, and investigate further improvement of hippocampus and amygdala segmentation.

References

[1] A. Akselrod-Ballin, M. Galun, J. Gomori, A. Brandt, and R. Basri. Prior knowledge driven multiscale segmentation of brain MRI. In *MICCAI 2007, Part II*, LNCS 4792, pages 118–126. Springer-Verlag, 2007.

[2] P.-L. Bazin and D. Pham. Statistical and topological atlas based brain image segmentation. In *MICCAI 2007, Part I*, LNCS 4791, pages 94–101. Springer-Verlag, 2007.

[3] M. Bhattacharjee, A. Pitot, A. Roche, D. Dormont, and E. Bardinet. Anatomy-preserving nonlinear registration of deep brain ROIs using confidence-based block matching. In *MICCAI 2008, Part II*, LNCS 5242, pages 956–963. Springer-Verlag, 2008.

[4] P. Cachier and N. Ayache. Regularization in image non-rigid registration: I. trade-off between smoothness and similarity. Research report RR-4188, INRIA, 2001.

[5] M. Cuadra, M. D. Craene, V. Duay, B. Macq, C. Pollo, and J.-P. Thiran. Dense deformation field estimation for atlas-based segmentation of pathological MR brain images. *Comput. Meth. Prog. Biomed.*, 84:66–75, 2006.

[6] B. Glocker, N. Komodakis, G. Tziritas, N. Navab, and N. Paragios. Dense image registration through MRFs and efficient linear programming. *Med. Image Anal.*, 12(6):731–741, 2008.

[7] S. Gouttard, M. Styner, S. Joshi, R. Smith, H. Cody, and G. Gerig. Subcortical structure segmentation using probabilistic atlas priors. In *Proc. SPIE Med. Imag. Conf.*, volume 6512, pages 65122J1–65122J11. SPIE, 2007.

[8] X. Han and B. Fischl. Atlas renormalization for improved brain MR image segmentation across scanner platforms. *IEEE Trans. Med. Imag.*, 26(4):479–486, 2007.

[9] R. Heckemann, J. Hajnal, P. Aljabar, D. Rueckert, and A. Hammers. Automatic anatomical brain MRI segmentation combining label propagation and decison fusion. *NeuroImage*, 33:115–126, 2006.

[10] NVIDIA. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide: Version 2.0*. NVIDIA Corporation, 2008.

[11] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever. Mutual-information-based registration of medical images: a survey. *IEEE Trans. Med. Imag.*, 22(8):986–1004, 2003.

[12] P. Rogelj, S. Kovačič, and J. Gee. Point similarity measures for non-rigid registration of multi-modal data. *Comput. Vis. Imag. Under.*, 92:112–140, 2003.

[13] T. Rohlfing, R. Brandt, R. Menzel, D. Russakoff, and C. Maurer, Jr. Quo vadis, atlas-based segmentation? In *The Handbook of Medical Image Analysis*, pages 435–486. Kluwer Academic / Plenum Publishers, New York, 2005.

[14] M. Sdika. A fast nonrigid image registration with constraints on the Jacobian using large scale constrained optimization. *IEEE Trans. Med. Imag.*, 27(2):271–281, 2008.

[15] R. Shams and N. Barnes. Speeding up mutual information computation using NVIDIA CUDA hardware. In *Proc. Digital Image Computing: Techniques and Applications (DICTA)*, pages 555–560, Australia, Dec. 2007.

[16] R. Stefanescu, X. Pennec, and N. Ayache. Grid powered nonlinear image registration with locally adaptive regularization. *Med. Image Anal.*, 8:325–342, 2004.

[17] E. Suárez, J. Santana, E. Rovaris, C. Westin, and J. Ruiz-Alzola. Fast entropy-based nonrigid registration. In *EUROCAST 2003*, LNCS 2809, pages 607–615. Spinger, Heidelberg, 2003.

[18] P. Viola and W. Wells. Alignment by maximization of mutual information. *Int. J. Comput. Vision*, 24(2):137–154, 1997.

[19] S. Warfield, K. Zou, and W. Wells. Simultaneous truth and performance level estimation (STAPLE): An algorithm for the validation of image segmentation. *IEEE Trans. Med. Imag.*, 23(7):903–921, 2004.

[20] J. Wu and A. Chung. Markov dependence tree-based segmentation of deep brain structures. In *MICCAI 2008, Part II*, LNCS 5242, pages 1092–1100. Springer-Verlag, 2008.

[21] M. Wu, C. Rosano, P. Lopez-Garcia, C. Carter, and H. Aizenstein. Optimum template selection for atlas-based segmentation. *NeuroImage*, 34:1612–1618, 2007.

[22] J. Zhou and J. Rajapakse. Segmentation of subcortical brain structrues using fuzzy templates. *NeuroImage*, 28:915–924, 2005.