3D Image Registration Using Distributed Parallel Computing

Huajun Song, and Peihua Qiu

***Abstract*—3D images have become increasingly popular in practice. They are commonly used in medical imaging applications. In such applications, it is often critical to compare two 3D images, or monitor a sequence of 3D images. To make the image comparison or image monitoring valid, the related 3D images should be geometrically aligned first, which is called image registration. However, image registration for 3D images would take much computing time, especially when a flexible method is considered, which does not impose any parametric form on the underlying geometric transformation. In this paper, we explore a fast-computing environment for 3D image registration based on the distributed parallel computing. The selected 3D image registration method is based on the Taylor's expansion and 3D local kernel smoothing. It is flexible, but involves much computation. We demonstrate that this fast-computing environment can effectively handle the computing problem while keeping the good properties of the 3D image registration method. The method discussed in the paper is therefore useful for applications involving big data.**

***Index Terms*—Big data; Function approximation; Geometric transformation; Image mapping; Kernel estimation; Spark cluster; Distributed parallel computing.**

I. Introduction

In practice, images are widely used as a medium of information extraction. To know an image object, we often take multiple images of the object and then compare different images for combining their useful information about the object. To this end, image registration (IR) is a necessary pre-processing step to geometrically align different images and make their comparison meaningful [1], [2]. For different IR applications, see papers such as [3], [4], [5], [6], [7].

In the literature, most IR methods are proposed in 2D cases. These methods are either feature-based or intensity-based [8]. Feature-based methods need to extract features from the observed images first, which is often computing intensive. Also, we need to choose threshold values, etc., when extracting features, and objective decisions for such tasks are usually difficult. For these reasons, intensity-based methods get more and more popular, which estimate the geometric transformation $\mathbf{T}$ in the IR problem from the observed image intensities of the related images directly. Some existing 2D intensity-based IR (IBIR) methods describe $\mathbf{T}$ parametrically, and some others adopt nonparametric transformations [10], [11], [12], [13], [14], [15], [16]. In real life, we need to handle 3D objects. Thanks to the rapid development of imaging techniques, we can obtain 3D images in some applications [17]. However, existing discussion about 3D IR is limited. Some people generalized certain 2D IR methods into 3D cases [18], [19]. See, for instance, the 3D IR procedures included in 3DSlicer (http://slicer.org/). Another example is the Iterative Closest Point (ICP) method [20], [21], which is for matching two 3D point cloud data that are substantially different from the observed image intensity data of 3D images.

In this paper, we focus on 3D image registration. In such cases, if a flexible nonparametric IR method (e.g., the one in [19]) is used, then computation involved is extensive because of the big data volume of 3D images and the flexibility of the 3D IR method. This computing issue is especially relevant when we try to monitor a sequence of 3D images (e.g., in applications of fMRI). In the literature, there are three main methods for handling the computing issue, all of which are based on parallel computing now [22]. First, the multicore CPU parallel computing method is developed based on OpenMP, which is one of the single-instruction-multiple-data (SIMD) models. This method splits a problem into independent sub-problems that are solved by threads in parallel. Each thread is mapped to a CPU core for execution. Second, the graphics processing unit (GPU) parallel computing method is developed in the framework of compute unified device architecture (CUDA). By this method, a problem is also splitted into independent sub-problems. These sub-problems are solved in blocks in parallel. Differing from the CPU parallel computing method, the GPU parallel computing method can further split each sub-problem and solved it by the synergic threads within a block. Third, the heterogeneous parallel computing method combines the strengths of GPUs and multicore CPUs to fully exploit the computer performance. In this model, GPU and CPU can be managed through task-parallelism or data-parallelism.

To address the computing problem, we propose to use a high performance computing (HPC) system to speed up the 3D IR algorithm in this paper. This system consists of clusters or dedicated hardware devices, such as the GPU and Field-Programmable Gate Array (FPGA). In the literature [23], [24], some researchers accelerate their algorithms using GPU. However, GPU and FPGA memory are difficult to store big 3D image data and a single system has a limited speed and a complex programming structure. To resolve the problem of big data processing, Hadoop and Spark were proposed in the literature [25], [26]. Hadoop is an open cloud computing framework that can deploy and dispatch each node in the cluster [25]. It runs a MapReduce job [27] for processing data in-parallel on clusters and generating big data-sets. However, MapReduce could not effectively process interactive and it-

Huajun Song is with the College of Information and Control Engineering, China University of Petroleum (East China).

Peihua Qiu is with the Department of Biostatistics, University of Florida.

erative computation. As an alternative, Spark was developed by Berkeley AMP laboratory [26]. The RDD (Resilient Distributed Datasets) model in Spark can speed up data processing [28], so that Spark runs faster than Hadoop in interactive and iterative computation. It has become the most popular computation platform with the potential to handle big data applications in image processing, data mining, remote data processing, and visual object recognition. Spark has already be used for processing SAR images [29]. In this paper, we propose a high speed distributed parallel computing algorithm for 3D image registration based on the Spark framework, which is critically important for effective 3D image registration due to the big data volume involved. As far as we know, high speed computation has not been discussed in the literature yet for 3D image registration. Hope this paper can attract more attention from researchers to tackle this important computing problem.

To describe the proposed distributed parallel computing algorithm, we first briefly described the local smoothing 3D IR method that was originally suggested in [19] in Section II. Then, the proposed Spark HPC framework for high speed computing is presented in Section III. To study the numerical performance of the proposed methods, we present some numerical results in Section IV. Finally, we conclude with several remarks in Section V.

II. 3D Nonparametric Intensity-Based Image Registration

In this section, we briefly introduce the nonparametric IR method that was originally discussed in [19]. This method is based on local smoothing estimation. It does not impose any parametric form on the geometric transformation describing the location difference between two images to register. Thus, it is flexible; but, the price to pay is its extensive computing.

Assume that R and M are two 3D images to register, where R is called reference image and M is called moved image. Their true image intensity functions have the relationship $M(T_1(u,v,w),T_2(u,v,w),T_3(u,v,w)) = R(u,v,w)$, for $(u,v,w)\in\Omega$, where Ω is the design space of the image R, and $\mathbf{T}(u,v,w)=(T_1(u,v,w),T_2(u,v,w),T_3(u,v,w))$ is an unknown transformation for describing the location difference between the images R and M. The major goal of IBIR methods is to estimate $\mathbf{T}(u,v,w)$ from the observed image intensities that are assumed to follow the model

$$\begin{aligned} &Z_M(u_i,v_j,w_k) = M(u_i,v_j,w_k)+\varepsilon_M(u_i,v_j,w_k),\\ &Z_R(u_i,v_j,w_k) = R(u_i,v_j,w_k)+\varepsilon_R(u_i,v_j,w_k),\\ &i,j,k=1,2,\ldots,n, \end{aligned} \tag{1}$$

where $\{(u_i,v_j,w_k)\}$ are voxel locations, and $\varepsilon_M(u_i,v_j,w_k)$ and $\varepsilon_R(u_i,v_j,w_k)$ are zero-mean pointwise noise of the images M and R, respectively. In (1), besides pointwise noise, we can also consider spatial blur and other degradations [30], [31]. At a given point $(u,v,w)\in\Omega$, define $b(u,v,w)=T_1(u,v,w)-x$, $c(u,v,w)=T_2(u,v,w)-y$ and $d(u,v,w)=T_3(u,v,w)-z$. Then, estimation of $\mathbf{T}(u,v,w)$ is equivalent to estimation of $(b(u,v,w),c(u,v,w),d(u,v,w))$. Let $O(u,v,w : h_n)$ be a spherical neighborhood of (u,v,w) with radius h_n. Then, by the Taylor's expansion of $M(\mathbf{T}(u_i,v_j,w_k))$ at (u_i,v_j,w_k) and by the local weighted least square estimation, we can estimate $b(u,v,w)$, $c(u,v,w)$ and $d(u,v,w)$ by

$$\begin{aligned} \min_{b(u,v,w),c(u,v,w),d(u,v,w)} \sum_{i,j=1}^{n} \Big[& Z_M(u_i,v_j,w_k)-Z_R(u_i,v_j,w_k)\\ &+\widehat{M}_x^{'}(u_i,v_j,w_k)b(u,v,w)+\widehat{M}_y^{'}(u_i,v_j,w_k)c(u,v,w)\\ &+\widehat{M}_z^{'}(u_i,v_j,w_k)d(u,v,w)\Big]^2 K_{h_n}, \end{aligned} \tag{2}$$

where $K_{h_n}=K((u_i-x)/h_n,(v_j-y)/h_n,(w_k-z)/h_n)$, K is a trivariate kernel function with unit rounded support, and $\widehat{M}_u^{'}(u_i,v_j,w_k)$, $\widehat{M}_v^{'}(u_i,v_j,w_k)$, and $\widehat{M}_w^{'}(u_i,v_j,w_k)$ are the local linear kernel estimates of $M_u^{'}(u_i,v_j,w_k)$, $M_v^{'}(u_i,v_j,w_k)$, and $M_w^{'}(u_i,v_j,w_k)$. In this paper, we choose $K(u,v,w)=(1-u^2)(1-v^2)(1-w^2)$. By (2), the estimates are weighted averages of observed intensities in the neighborhood $O(u,v,w:h_n)$, and the weights are smaller if the related voxels are farther away from (u,v,w) (cf., Section 2.3 in [32]). By some algebraic operations, we can check that (2) has a solution only in cases when

$$\begin{aligned} k_{11}k_{22}k_{33}+k_{12}k_{23}k_{13}+k_{12}k_{23}k_{13}-k_{22}k_{13}^2-\\ k_{33}k_{12}^2-k_{11}k_{23}^2\neq 0, \end{aligned} \tag{3}$$

where, for $s,t=1,2,3$,

$$\begin{aligned} k_{st} = & \sum_{i,j,k=1}^{n}\Big[\widehat{M}_x'(u_i,v_j,w_k)\Big]^{I(s=1)}\Big[\widehat{M}_x'(u_i,v_j,w_k)\Big]^{I(t=1)}\\ \times & \Big[\widehat{M}_y'(u_i,v_j,w_k)\Big]^{I(s=1)}\Big[\widehat{M}_y'(u_i,v_j,w_k)\Big]^{I(t=1)}\\ \times & \Big[\widehat{M}_z'(u_i,v_j,w_k)\Big]^{I(s=1)}\Big[\widehat{M}_z'(u_i,v_j,w_k)\Big]^{I(t=1)}K_{h_n}. \end{aligned}$$

It can be checked that (3) is true when the image intensity function of M is a degenerate function around (u,v,w). In such cases, voxel (u,v,w) is called a degenerate voxel. Otherwise, it is called a non-degenerate voxel. See a related discussion in [15] in 2D setup. So, $\mathbf{T}(u,v,w)$ is well defined and can be properly estimated by (2) only around non-degenerate voxels. Based on that result, a 3D IBIR algorithm was suggested in [19], which is described below.

Step 1. By an edge detection method, properly detect all edge voxels in Z_R (cf., Chapter 6 in [32]).

Step 2. For a voxel (u,v,w) in R, let $O(u,v,w;r_1)$ be its rounded neighborhood with radius r_1. In cases when the number of detected edge voxels in $O(u,v,w;r_1)$ is smaller than $[nr_1]$ and the left-hand-side of formula (3) with R in place of M is not smaller than a given number μ_n, then we claim that (u,v,w) is a non-degenerate voxel of R; otherwise, it is a degenerate voxel.

Step 3. Let (u,v,w) be a non-degenerate voxel of R, or a voxel whose neighborhood $O(u,v,w;r_1)$ contains $[nr_1]$ or more detected edge voxels. Then, we estimate $\mathbf{T}(u,v,w)$ as follows. For any voxel $(u',v',w')\in O(u,v,w;r_1)$ in M,

consider $O(x', y', z'; r_2)$, where r_2 might be different from r_1. The mean squared difference (MSD) is then defined by

$$MSD((u', v', w'); (u, v, w)) = \frac{1}{N} \sum_{(u'+x, v'+y, w'+z) \in O(u', v', w'; r_2)} [Z_M(u' + x, v' + y, w' + z) - Z_R(u + x, v + y, w + z)]^2, \quad (4)$$

where N denotes the number of voxels in $O(u', v', w'; r_2)$. Then, $\mathbf{T}(u, v, w)$ is estimated by the minimizer of

$$\min_{(u', v', w') \in O(u, v, w; r_1)} MSD((u', v', w'); (u, v, w)).$$

Step 4. Let (u, v, w) be a degenerate voxel of R. Then, $\mathbf{T}(u, v, w)$ is estimated by the following algorithm. First, find a non-degenerate voxel, or a voxel (u', v', w') whose neighborhood $O(u', v', w'; r_1)$ contains $[nr_1]$ or more detected edge voxels, that is closest to (u, v, w). That voxel is denoted as $(u^{(1)}, v^{(1)}, w^{(1)})$. Let $\widehat{\mathbf{T}}^*(u, v, w) = (u, v, w) + (\widehat{b}(u^{(1)}, v^{(1)}, w^{(1)})), \widehat{c}(u^{(1)}, v^{(1)}, w^{(1)}), \widehat{d}(u^{(1)}, v^{(1)}, w^{(1)}))$. Define $\widehat{\mathbf{T}}(u, v, w) = \widehat{\mathbf{T}}^*(u, v, w)$ if $MSD(\widehat{\mathbf{T}}^*(u, v, w); (u, v, w)) \leq MSD((u, v, w); (u, v, w))$; Otherwise, define $\widehat{\mathbf{T}}(u, v, w) = (u, v, w)$.

III. Distributed Parallel Computing Framework

A. Building of Spark Computing Platform

As discussed in Section I, the 3D IBIR procedure described in Section II is quite computing extensive, partly because (i) 3D images often have millions of voxels, (ii) there are a large number of 3D images to handle in many applications (e.g., monitoring of a sequence of fMRI images), and (iii) the considered 3D IBIR procedure is flexible by allowing the underlying geometric transformation $\mathbf{T}(u, v, w)$ to be nonparametric. In this part, we describe the structure of our proposed distributed parallel computing framework using Spark and RDD. As mentioned earlier, Spark is an in-memory cluster computing platform that is designed specifically for iterative calculation, and an RDD is a read-only collection of records partitioned across a set of computers. Because Spark with RDD has been demonstrated to be much faster than Hadoop in handling iterative and interactive jobs [26], it is appropriate to use here for the computing of the 3D IBIR procedure described in Section II.

Our proposed Spark platform is designed using the master-slave model and implemented on a computer cluster. The designed computer cluster consists of one master node, also called driver program, and a set of slave nodes, also called data nodes, both of which are physical machines. Fig. 1 demonstrates the platform with four data nodes. The driver program runs the main function of the application and creates the Spark Context. Spark Context is the entrance of the Spark platform, which is responsible for connecting the Spark cluster, creating RDD, initialization of the computing algorithms, and so forth. In the Spark platform, the driver program serves as the master, and the data nodes serve as workers. In each data node, there could be multiple CPU cores to execute different tasks, and cache is the high performance memory for storing data. The two most critical parts in Hadoop are the

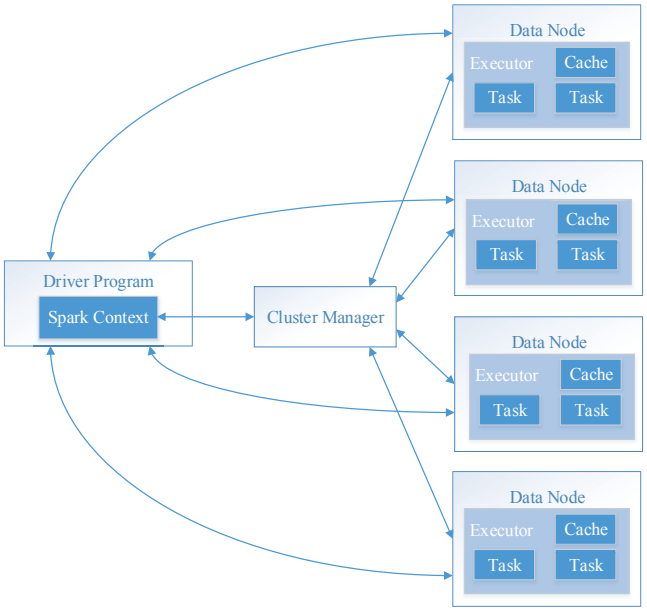


Fig. 1. Demonstration of the proposed Spark distributed parallel computing framework with one master node and four slave nodes.

MapReduce and the HDFS (Hadoop Distributed File System). Spark improves the MapReduce by using RDD, while keeping HDFS, to speed up the overall computation. For all numerical experiments presented in Section IV, we use a Spark platform with the hardware configuration specified in Table I and software configuration specified in Table II.

TABLE I
HARDWARE CONFIGURATION OF THE SPARK PLATFORM USED IN ALL NUMERICAL EXAMPLES.

Role	quantity	Memory (GB)	CPU
Driver Program	1	6	Intel Core(TM) CPU i3-3220 @ 3.3 GHz 4
Data Node	4	4	Intel Core(TM) CPU i3-3220 @ 3.3 GHz 4

TABLE II
SOFTWARE CONFIGURATION OF THE SPARK PLATFORM USED IN ALL NUMERICAL EXAMPLES.

Software	Version
Operating system	CentOS 7.0
Spark	Spark-1.6.1
Hadoop	Hadoop-2.6.0
Scala	Scala-2.10.6
Java	Jdk1.7.0_21

B. Computing Complexity of the 3D IBIR Procedure

In the proposed 3D local registration algorithm, the amount of computation is mainly contributed by (4). This formula computes the squared difference for each voxel in the spherical neighborhood with radius r_2 of a given voxel (u, v, w). Also, we need to search for the minimum of all computed MSDs by the formula immediately below (4). Let the total number of voxels in the reference image be S, and the numbers of voxels in the spherical neighborhoods with radius of r_1 and r_2 be P and N, respectively. Then, the computational complexity of the 3D IBIR procedure involves $2N \times P \times S$ additions, $(N + 1) \times P \times S$ multiplications, and $P \times S$ comparisons.

Assume that the computer's clock cycle is T. Then, the computer dominant frequency is $f = 1/T$. Let the computation cycles of addition or comparison be aT, and the computation cycles of multiplication be bT. Then, the computing time t of a Spark system with m CPU cores should be

$$t = \frac{SP((2a + b)N + b + a)}{mf},$$

where the parameters a, b and f are related to the performance of the computer and their values can be found from the CPU manual.

Now, assume that the reference image has $S = W \times H \times D$ voxels, where W, H, and D are the numbers of rows in the three dimentions, respectively. Then, the total computing time can be expressed as

$$t = \frac{4WHDr_1^3(4(2a + b)r_2^3 + b + a)}{mf}, \tag{5}$$

in which we have approximated the values of $P = \frac{4}{3}\pi r_1^3$ and $N = \frac{4}{3}\pi r_2^3$ by $P = 4r_1^3$ and $N = 4r_2^3$, respectively.

IV. Numerical Study

A. Evaluation of the 3D IBIR Procedure

In this part, we present some numerical results regarding the performance of the local smoothing 3D IR procedure described in Section II, which is denoted as LS3DIR. When implementing this procedure, we use the proposed Spark platform with 8 CPU cores and other hardware and software configurations specified in Tables I and II. To geometrically match up two related images, the reference image is divided into slides along the z-axis and different CPU cores are assigned to handle image registration for different slides. When registering one slide of the reference image with the moved image, the whole moved image is available. So, the division of the original image registration task into separate registrations between individual slides of the reference image and the moved image will not change the registration results, and it will only affect the computing time. In the numerical study, besides LS3DIR, we also consider certain methods in the software package *3DSlicer* for comparison purposes. These methods include the free-form deformation method based on B-splines, denoted as B-Spline, the IR method using the affine invariant geometric transformation, denoted as Affine, and the method using the rigid-body geometric transformation, denoted as Rigid. These methods represent different state-of-the-art 3D IBIR methods in the literature. The version of *3DSlicer* used here is the latest release 4.5.0, available at http://www.slicer.org/. For comparison purposes, we also include results when no image registration is performed. This case is denoted as No-Registration.

Three popular performance measures are used here, which include the root residual mean squares (RRMS), the correlation coefficient (CC), and the entropy of image difference (EID) [33]. Image registration is regarded better if RRMS and EID are smaller, or CC is larger.

For the methods B-Spline, Affine, and Rigid in the software package *3DSlicer*, we use their default settings in the software. The method LS3DIR has three parameters h_n, r_1 and r_2. Based on many numerical experiments, we found that the results are reasonably good if we choose $h_n/n \in [0.1, 0.2]$, $r_1 = 2r_2$, and $r_2/n \in [0.1, 0.2]$. In the first example, we choose $h_n = 10$, $r_1 = 20$, and $r_2 = 10$. The 3D reference image is downloaded from http://www.slicer.org/slicerWiki/images/5/59/RegLibC011.nrrd. It is a 3D brain image with $128 \times 128 \times 56$ voxels. Its 10 slices at $z = 10, 15, 20, 25, 30, 35, 40, 45, 50, 55$ are shown in the first row of Fig. 2. The moved image is obtained as follows: for $y = 32, 33, \ldots, 75$, we move the voxel at (u, v, w) to $(x + 5\sin(4\pi y/180), y, z)$. By this transformation, voxels located between 32th and 75th rows along the y-axis are moved along the x-axis by the amount of $5\sin(4\pi y/180)$, for all slices along the z-axis. After the local distortion transformation, we rotate the resulting reference image by 2, 1.5, 2.5 degrees along the x-, y-, and z-axes, respectively, and then move the rotated image by -3, 2 and -2 voxels along the three axes. The 10 slices at $z = 10, 15, 20, 25, 30, 35, 40, 45, 50, 55$ of the moved image are shown in the second row of Fig. 2. By comparing these slices with the slices of the reference image, we can see that the geometric transformation changed the shape of the reference image quite dramatically.

We then apply the five IBIR methods to this dataset. The 10 slices of the restored image of the proposed method LS3DIR are shown in the last row of Fig. 2. It can be seen that the restored image by LS3DIR looks almost the same as the original reference image. The performance measures of the related 5 methods are presented in Table III. It can be seen that the methods Affine and Rigid are slightly better than the method No-Registration in this example, because part of the true geometric transformation is rigid-body (i.e., the rotation and translation part) although the remaining part is not affine-invariant (i.e., the local distortion part). The method B-Spline is effective in this example, and the proposed method LS3DIR is much more effective than the other four competing methods. The 20th and 40th slices along the z-axis of the residual images of the five IR methods are shown in Fig. 3. It can be seen that the ones of the proposed method LS3DIR have the least pattern, compared to the ones of the other four methods.

B. Evaluation of the Proposed Spark Platform

In this part, we evaluate the computing complexity of the proposed Spark platform, and the acuracy of the for-

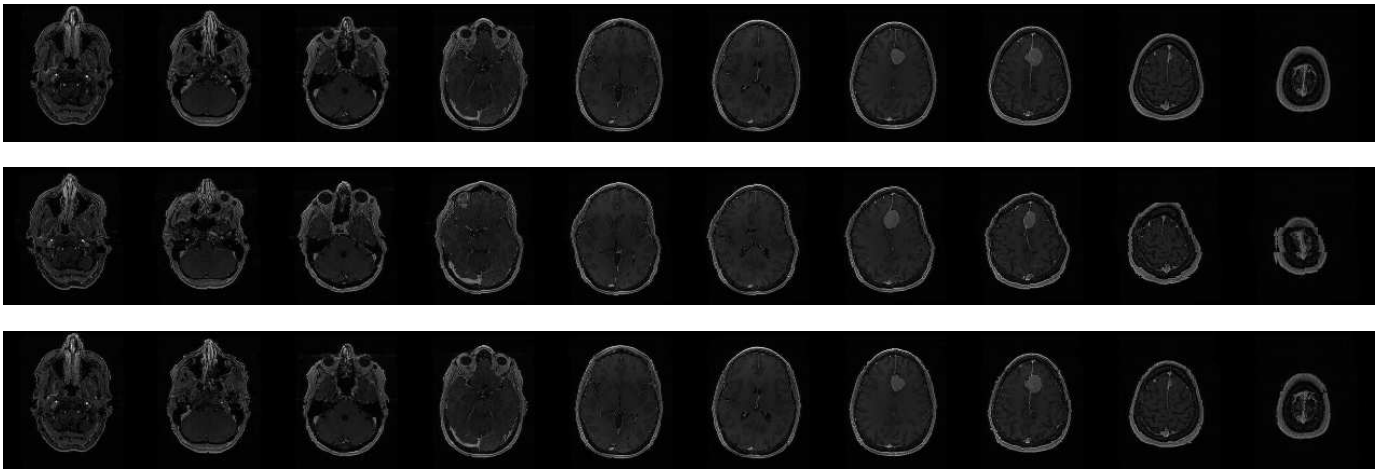
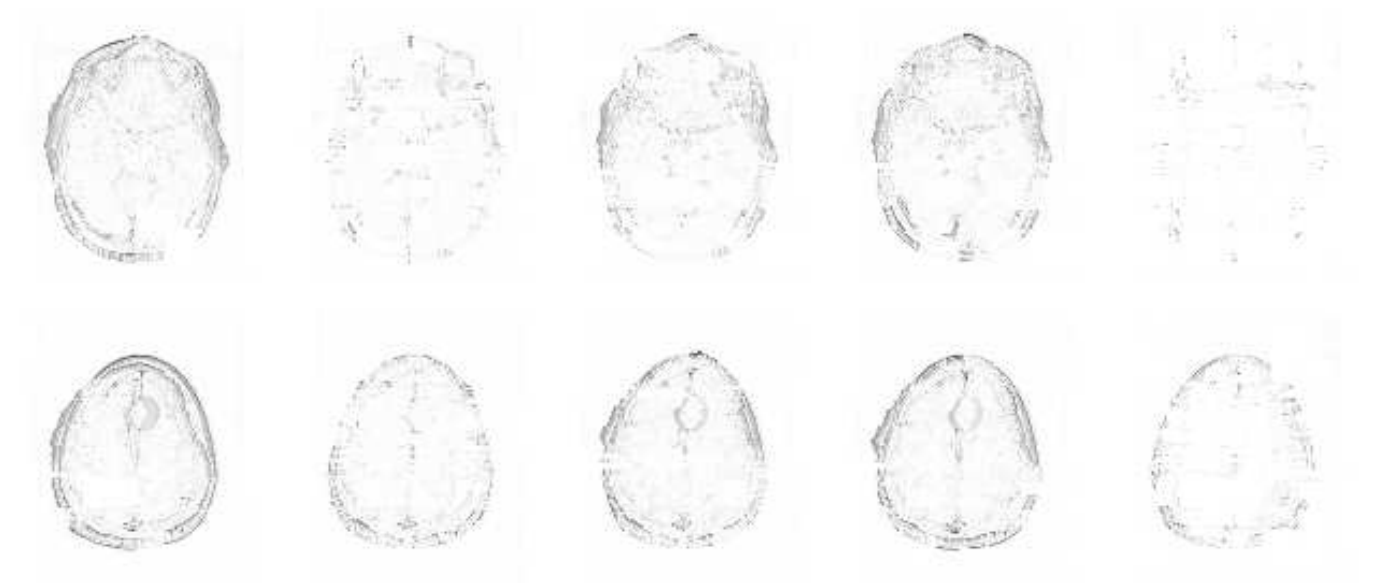


Fig. 2. Downloaded brain image example. Images in the first row are the 10 slices at $z = 10, 15, 20, 25, 30, 35, 40, 45, 50, 55$ of the reference image. The ones in the second row are the corresponding slices of the moved image, and the ones in the last row are the corresponding slices of the restored reference image by LS3DIR, defined as $M(\widehat{T}(u, v, w))$.



(a) (b) (c) (d) (e)

Fig. 3. Downloaded brian image example. (a)-(e) The 20th (top) and 40th (bottom) slices of the residual images of the methods No-Registration, B-Spline, Affine, Rigid, and LS3DIR, respectively.

TABLE III
PERFORMANCE MEASURES OF THE FIVE IBIR METHODS IN THE DOWNLOADED BRAIN IMAGE EXAMPLE.

	No-Registration	B-Spline	Affine	Rigid	LS3DIR
RRMS	18.435	9.305	14.388	15.823	**5.769**
CC	0.649	0.919	0.804	0.762	**0.966**
EID	4.241	3.338	3.729	3.954	**1.487**

mula (5). To this end, we use the following two datasets. The first one is a 3D chest image downloaded from http://www.slicer.org/slicerWiki/images/3/31/CT-chest.nrrd. It has $128 \times 128 \times 69$ voxels, and is used as a reference image here. Its 10 slices are shown in the first row of Fig. 4. The second dataset is a patient's 3D brain image obtained in a recent medical study at University of Florida (UF). It has $128 \times 128 \times 88$ voxels and its 10 slices are shown in the first row of Fig. 5. This image is different from the one considered in the previous example in Section IV-A. We then apply the same geometric transformation as the one in the previous example to these two reference images to obtain the two moved images. Their corresponding slices are shown in the second rows of Fig. 4 and Fig. 5, respectively.

We then apply the 3D IBIR procedure LS3DIR under the proposed Spark platform to these two datasets. In the IBIR procedure, the selected values of the two parameters r_1 and r_2 are given in Table IV, together with the dimensionality of the images. These values are chosen because they can give

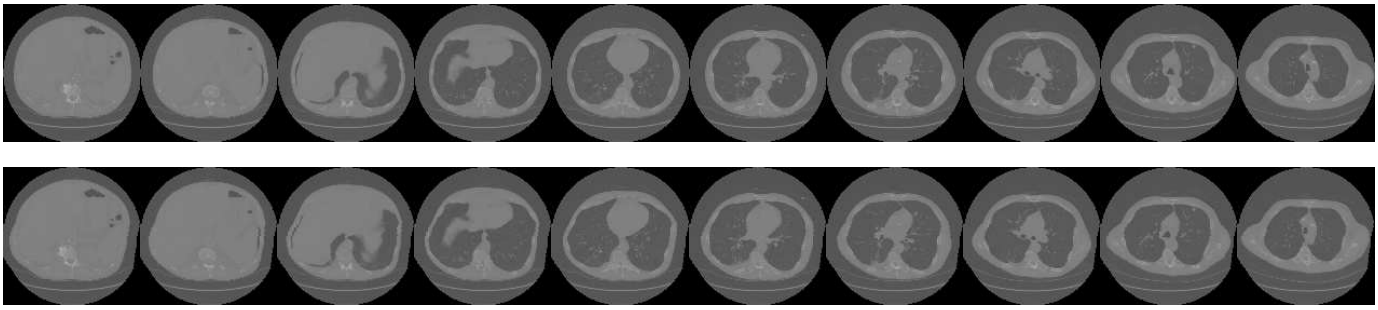


Fig. 4. Chest image example. The ones in the first row are the 10th, 15th, . . ., and 55th slices along the z-axis of the 3D reference image, and those in the second row are the corresponding slices of the moved image.

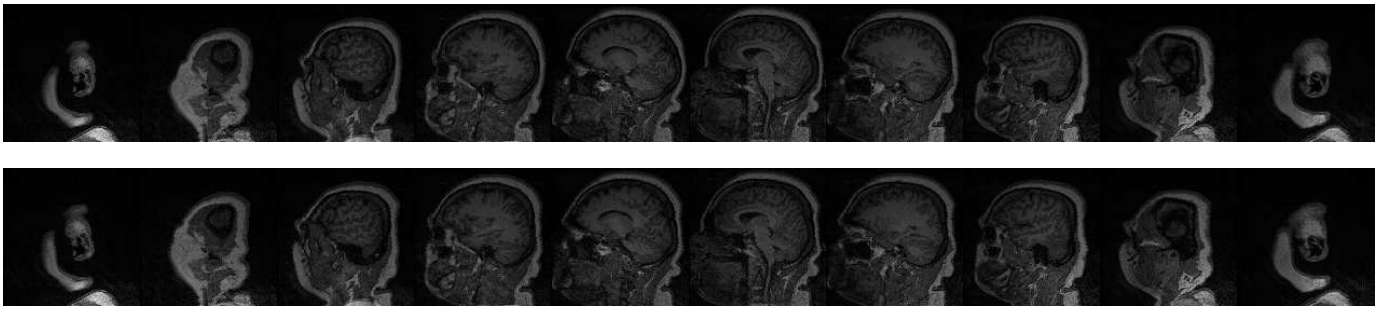


Fig. 5. UF Patient's brain image example. The ones in the first row are the 10th, 15th, . . ., and 55th slices along the z-axis of the 3D reference image, and those in the second row are the corresponding slices of the moved image.

reasonably good image registration results. The Spark platform still has 4 data nodes and its other hardware and software configurations are specified in Tables I and II. Our computers' addition clock cycle number is 5, their multiplication clock cycle number is 95. The number of CPU cores is changed among 4, 8, 12 and 16. In each case, the actual computing time is recorded and presented in Table V and Fig. 6, along with the computing time calculated by formula (5). From Table V and Fig. 6, it can be seen that (i) the more CPU cores we use, the less computing times, (ii) the actual and theoretical computing times are very close, and (iii) the actual computing times are slightly larger than the theoretical computing times. Results (i) and (ii) are intuitively reasonable. One explanation for result (iii) is that we only include the major computing operations (e.g., additions and multiplications) in the formula (5), and some less time-consuming operations and data processing (e.g., checking whether a give voxel is in the neighborbood of another voxel) are neglected.

TABLE IV
DIMENSIONS OF THE CHEST IMAGE AND THE UF PATIENT'S BRAIN IMAGE USED IN EVALUATING THE PROPOSED SPARK PLATFORM, AND THE SELECTED VALUES OF THE PARAMETERS r_1 AND r_2 IN THE IBIR PROCEDURE LS3DIR.

	W	H	D	r_1	r_2
Chest Image	128	128	69	10	6
Brain Image	128	128	88	12	8

In the above example, for comparison purposes, we also consider an alternative parallel computing platform GPU that uses NVIDIA GTX1050 TI and has 768 CUDA cores, 4G high-speed internal memory, and 1.3GHz core frequency. To finish the same image registration tasks using LS3DIR, its computing time is 2.7h for the chest image and 13.9h for the UF brain image. By comparing the computing times of the proposed Spark platform with the ones of this alternative platform (cf., Table V), we can see that the proposed platform with 16 CPU cores can already outperform the GPU with 768 CUDA cores. Therefore, the former is much more efficient in saving computing time.

TABLE V
THE ACTUAL AND THEORETICAL COMPUTING TIMES OF THE IBIR PROCEDURE LS3DIR USING THE PROPOSED SPARK PLATFORM WITH DIFFERENT NUMBERS OF CPU CORES. THE LAST COLUMN GIVES THE COMPUTING TIMES OF THE ALTERNATIVE PARALLEL COMPUTING PLATFORM GPU WITH 768 CUDA CORES.

	CPU cores	Actual time	Theoretical time	GPU
Chest Image	4	9h	8.6h	2.7h
	8	4.6h	4.3h	
	12	3.2h	2.9h	
	16	2.4h	2.2h	
UF Brain Image	4	45.8h	45.12h	13.9h
	8	23.3h	22.56h	
	12	16.2h	15.04h	
	16	12.2h	11.28h	

It should be pointed out that the computing time of an image registration task using the IBIR procedure LS3DIR and the proposed Spark platform depends mainly on the size of

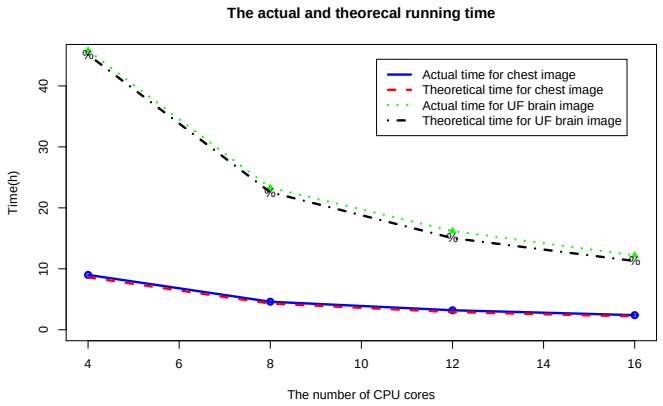


Fig. 6. The actual and theoretical computing times of the IBIR procedure LS3DIR under the proposed Spark platform with different numbers of CPU cores.

the original reference image and the parameters r_1 and r_2 in LS3DIR. See formula (5) and the related discussion in Section III-B. Next, we use the UF brain image as an example and consider six cases with different values of image size and (r_1, r_2) that are listed in Table VI. In each case, the first D rows of the original reference and moved images along the z-axis are used in the study. Then, the proposed method is applied and its actual and theoretical computing times are presented in Table VII. In the table, besides cases with 4, 8, 12 and 16 CPU cores, we also consider the case with only 1 CPU core. This case is equivalent to the case when LS3DIR is executed traditionally without using the proposed Spark platform. From the table, we can see that the actual and theoretical computing times are generally close to each other in this example, as in Table V, and the proposed Spark platform can save computing times dramatically, as compared to the computing times when only 1 CPU core is used.

TABLE VI
SIX CASES WITH DIFFERENT VALUES OF THE UF BRAIN IMAGE SIZE AND THE PARAMETERS r_1 AND r_2 IN THE IBIR PROCEDURE LS3DIR.

Case	W	H	D	r_1	r_2
(i)	128	128	4	20	10
(ii)	128	128	8	20	10
(iii)	128	128	12	16	8
(iv)	128	128	14	16	8
(v)	128	128	69	10	6
(vi)	128	128	88	10	6

TABLE VII
THE ACTUAL AND THEORETICAL COMPUTING TIMES OF THE IBIR PROCEDURE LS3DIR UNDER THE PROPOSED SPARK PLATFORM WITH DIFFERENT NUMBERS OF CPU CORES.

Case	CPU cores	Actual time	Theoretical time
(i)	1	2.48h	2.23h
	4	0.73h	0.56h
	8	0.35h	0.28h
	12	0.23h	0.19h
	16	0.18h	0.14h
(ii)	1	18.47h	17.80h
	4	4.9h	4.45h
	8	2.5h	2.23h
	12	1.7h	1.48h
	16	1.3h	1.11h
(iii)	1	22.03h	21.87h
	4	6h	5.47h
	8	3h	2.73h
	12	2h	1.83h
	16	1.5h	1.37h
(iv)	1	29.79h	29.77h
	4	8h	7.44h
	8	4h	3.72h
	12	2.7h	2.48h
	16	2h	1.86h
(v)	1	35.21h	34.57h
	4	9h	8.64h
	8	4.6h	4.32h
	12	3.2h	2.88h
	16	2.4h	2.16h
(vi)	1	46.23h	44.09h
	4	12.5h	11.02h
	8	6.4h	5.51h
	12	4.3h	3.67h
	16	3.2h	2.76h

V. CONCLUDING REMARKS

We have described our proposed Spark distributed parallel computing platform for 3D image registration. Numerical study shows that this platform can reduce the computing time dramatically while keeping the same performance in image registration. Thus, it is appropriate to use in 3D image registration, where the data volume is usually big and it is important to keep the computing time within a reasonable time frame. In the current version of the Spark platform, individual CPU cores are assigned to handle image registration for different slices along the z-axis of a reference image. We have not studied whether there is a better job assignment scheme yet. Also, there are data nodes and CPU cores involved in the proposed Spark platform, where different data nodes represent different computers, and then each computer may have multiple CPU cores. So far, we have not studied the best combination of data nodes and CPU cores yet in terms of computing efficiency. These issues should be further studied in future research.

ACKNOWLEDGMENTS

The authors thank the editor, the associate editor and two referees for many constructive comments and suggestions, which improved the quality of the paper greatly. This research was supported in part by the Fundamental Research Funds of the Central Universities in China (18CX02109A) and by a grant of the US National Science Foundation (DMS1405698).

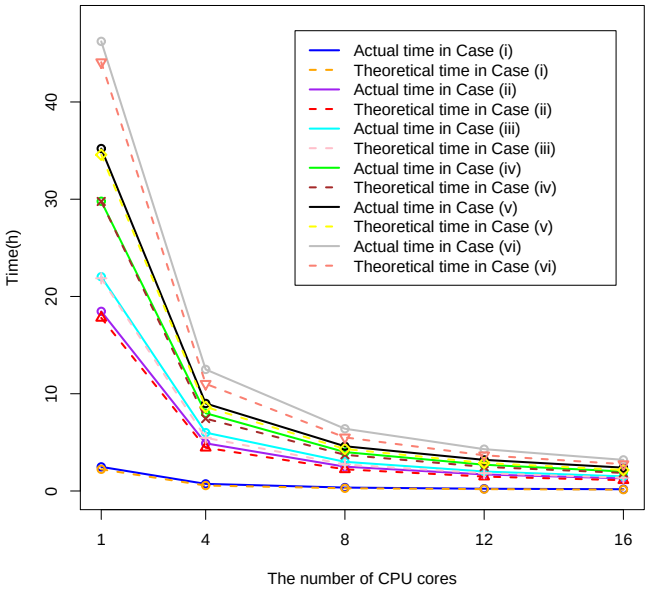


Fig. 7. The actual and theoretical computing times of the IBIR procedure LS3DIR using the proposed Spark platform in six cases listed in Table VI.

REFERENCES

[1] Zitova, B., and Flusser, J. (2003), *Image registration methods: a survey,* Image and Vision Computing, 21, 977–1000.

[2] Modersitzki, J. (2009), *Fair: Flexible Algorithms for Image Registration*, SIAM: Philadelphia.

[3] Shang, L., Lv, C.J., and Yi, Z. (2006), *Rigid medical image registration using PCA neural network,* Neurocomputing, 69, 1717–1722.

[4] Li, H., Manjunath, B.S., and Mitra, S.K. (1995), *A contour-based approach to multisensor image registration,* IEEE Transactions on Image Processing, 4, 320–334.

[5] Liu, L., Jiang, T., Yang, J., and Zhu, C. (2006), *Fingerprint registration by maximization of mutual information,* IEEE Transactions on Image Processing, 15, 1100–1110.

[6] Dufaux, F., and Konrad, J. (2000), *Efficient, robust, and fast global motion estimation for video coding,* IEEE Transactions on Image Processing, 9, 497–501.

[7] Irani, M., and Peleg, S. (1993), *Motion analysis for image enhancement: resolution, occlusion and transparency,* Journal of Visual Communication and Image Representation, 4, 324–335.

[8] Qiu, P., and Xing, C. (2013a), *Feature based image registration using non-degenerate pixels,* Signal Processing, 93, 706–720.

[9] Denton, E.R., Sonoda, L.I., Rueckert, D., Rankin, S.C., Hayes, C., Leach, M.O., Hill, D.L., and Hawkes, D.J. (1999), *Comparison and evaluation of rigid, affine, and nonrigid registration of breast MR images,* Journal of Computer Assisted Tomography, 23, 800–805.

[10] Avants, B.B., Epstein, C.L., Grossman, M., and Gee, J.C. (2008), *Symmetric diffeomorphic image registration with cross-correlation: evaluating automated labeling of elderly and neurodegenerative brain,* Medical Image Analysis, 12, 26–41.

[11] Pan, W., and Qin, K., and Chen, Y. (2009), *An adaptable-multilayer fractional Fourier transform approach for image registration,* IEEE Transactions on Pattern Analysis and Machine Intelligence, 31, 400–412.

[12] Qiu, P., and Xing, C. (2013b), *On nonparametric image registration,* Technometrics, 55, 174–188.

[13] Rajwade, A., Banerjee, A. and Rangarajan, A. (2009), *Probability density estimation using isocontours and isosurfaces: application to information-theoretic image registration,* IEEE Transactions on Pattern Analysis and Machine Intelligence, 31, 475–491.

[14] Tustison, N.J., Avants, B.B., and Gee, J.C. (2009), *Directly manipulated free-form deformation image registration,* IEEE Transactions on Image Processing, 18, 624–635.

[15] Qiu, P., and Xing, C. (2010), *Intensity Based Nonparametric Image Registration,* Proceedings of the 2010 ACM SIGMM International Conference on Multimedia Information Retrieval, 221–225, Philadelphia, PA.

[16] Xing, C., and Qiu, P. (2011), *Intensity Based Image Registration By Nonparametric Local Smoothing,* IEEE Transactions on Pattern Analysis and Machine Intelligence, 33, 2081–2092.

[17] Mukherjee, P.S., and Qiu, P. (2011), *3-D Image Denoising By Local Smoothing And Nonparametric Regression,* Technometrics, 53, 196–208.

[18] Song, H., and Qiu, P. (2017a), *A Parametric Intensity-Based 3D Image Registration Method for Magnetic Resonance,* Imaging, Signal, Image and Video Processing, 11, 455–462.

[19] Song, H., and Qiu, P. (2017b), *Intensity-Based 3D Local Image Registration,* Pattern Recognition Letters, 94, 15–21.

[20] Besl, P., and McKay, N. (1992), *A method for registration of 3-D shapes*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14, 239–256.

[21] Yang, J., Li, H., and Jia, Y. (2013), *Go-ICP: Solving 3D Registration Efficiently and Globally Optimally*, Proceedings of the 2013 IEEE International Conference on Computer Vision, 1457–1464.

[22] Chen K, Hui Y, Kumara S. *Parallel computing and network analytics for fast Industrial Internet-of-Things (IIoT) machine information processing and condition monitoring*. Journal of Manufacturing Systems, 2018, 46:282–293.

[23] Zhu, H. (2013), *Parallel unsupervised Synthetic Aperture Radar image change detection on a graphics processing unit*, International Journal of High Performance Computing Applications, 27, 109–122.

[24] Frey, O., Werner, C.L., and Wegmuller, U. (2014), *GPU-based parallelized time-domain back-projection processing for Agile SAR platforms*, IEEE Proceedings of the Geoscience and Remote Sensing Symposium, 1132–1135.

[25] Pengyao, W., Jianqin, W., and Ying, C. (2013), *Rapid processing of remote sensing images based on cloud computing*, Future Generation Computer Systems, 29, 1963–1968.

[26] Zaharia, Matei, et al. (2010), *Spark: cluster computing with working sets*, Proceedings of the 2nd USENIX conference on Hot topics in cloud computing, 10–10.

[27] Dean, J., and Ghemawat, S. (2008), *MapReduce: Simplified data processing on large clusters*, Communications of the ACM, 51, 107–113.

[28] Zaharia, Matei, et al. (2012), *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing*, Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, 2–2.

[29] Zhu, H., Guo, Y., Niu, M., Yang, G., and Jiao, L. (2015), *Distributed SAR image change detection based on Spark*, IEEE Proceedings of the International Geoscience and Remote Sensing Symposium, 4149–4152.

[30] Qiu, P. (2008), *A nonparametric procedure for blind image deblurring,* Computational Statistics and Data Analysis, 52, 4828–4841.

[31] Hall, P., and Qiu, P. (2007), *Blind deconvolution and deblurring in image analysis,* Statistica Sinica, 17, 1483–1509.

[32] Qiu, P. (2005), *Image Processing and Jump Regression Analysis*, New York: John Wiley & Sons.

[33] Qiu, P., and Nguyen, T. (2008), *On image registration in magnetic resonance imaging*, IEEE Proceedings of the 2008 International Conference on BioMedical Engineering and Informatics, 753–757.