Recursive Decomposition Network for Deformable Image Registration

Bo Hu, Shenglong Zhou, Zhiwei Xiong, *Member, IEEE*, and Feng Wu, *Fellow, IEEE*

***Abstract*—Deformation decomposition serves as a good solution for deformable image registration when the deformation is large. Current deformation decomposition methods can be categorized into cascade-based methods and pyramid-based methods. However, cascade-based methods suffer from heavy computational burdens and long inference time due to their structures of repeated subnetworks, while the effectiveness of pyramid-based methods is constrained by their limited numbers of resolution levels. In this paper, to address both the insufficient and inefficient decomposition problems in current deformation decomposition methods, we propose a recursive decomposition network (RDN) to offer a novel solution for deformable image registration. Stage-wise recursion can efficiently decompose a large deformation into different pyramid estimation stages without using repeated subnetworks like in cascade-based methods. Level-wise recursion can sufficiently decompose the deformation inside each resolution level instead of only one-time estimation like in pyramid-based methods. Extensive experiments and ablation studies on two representative datasets validate the effectiveness and efficiency of our proposed RDN.**

***Index Terms*—Convolutional neural network, deformable image registration, deformation decomposition.**

I. Introduction

DEFORMABLE image registration is a crucial task in medical image analysis, such as multi-modality image fusion [1] and image segmentation [2], [3]. Deformable image registration aims to learn a nonlinear dense deformation field to align them. Thanks to deformable image registration, clinical images captured from different patients, with different devices and at different times can be compared and analyzed. Traditional methods [4]–[10] regard the registration procedure as an optimization problem for each image pair. The optimization is iterative and therefore extremely slow in practice. Recently, methods based on convolutional neural networks (CNNs) [11]–[19] have attracted research attention. With CNNs, image registration can be regarded as mappings from inputs to outputs [20], making them much faster than traditional methods. By leveraging similarity metrics, unsupervised methods are free from expensive manual annotations and are dominant in registration methods.

Large deformation is a key problem for deformable image registration. In clinic, in addition to individual differences, many anatomical structures change dramatically due to diseases. For example, the brain of a patient with Alzheimer's disease has significant atrophy compared to a normal brain [21]. Despite that the encoder-decoder architecture of UNet [22] endows the networks with strong learning abilities, the direct estimation scheme, as shown in Fig. 1(a), proves to be a heavy burden for networks when handling large deformations [15], and leads to more serious estimation errors than those introduced when handling small deformations.

Deformation decomposition therefore serves as a good solution for the large deformation problem. By decomposing a large deformation into several small deformations, the large deformation estimation problem is simplified to progressively estimate several small deformations, which is much easier for CNNs. Current deformation decomposition methods can be categorized into cascade-based methods and pyramid-based methods. Cascade-based methods [15], [20], [23] regard a UNet as a subnetwork, and they stack multiple identical subnetworks together to form a large network to estimate a deformation field in a recursive manner. As shown in Fig. 1(b), a large deformation is decomposed to equal components sufficiently, and each subnetwork only needs to deal with a small deformation. Each cascade is connected by a continuous warping operation [24] applied on the moving image. However, due to their structures of repeated subnetworks, cascade-based methods suffer from heavy computational burden and long inference time. Pyramid-based methods [17], [25], [26] employ the multi-level features to estimate a deformation field in a coarse-to-fine manner. Different from cascade-based methods, each deformation component of pyramid-based methods is unequal. As shown in Fig. 1(c), low resolution levels possess large receptive fields to deal with large deformation components, while high resolution levels possess small receptive fields to deal with small deformation components. With this coarse-to-fine strategy, a large deformation can be decomposed into multi-level components and each level can be regarded as a refinement of the previous level, so that promising registration performance can be achieved. However, pyramid decomposition is insufficient because the number of decomposition times is constrained by the limited number of pyramid resolution levels. Moreover, pyramid-based methods only consider the decomposition across

Manuscript received 23 December 2021; revised 25 March 2022, 6 May 2022, and 28 June 2022; accepted 30 June 2022. Date of publication 11 July 2022; date of current version 5 October 2022. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFA0700800, in part by the National Natural Science Foundation of China under Grant 62021001, and in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2019-025. *(Bo Hu and Shenglong Zhou are co-first authors.) (Corresponding author: Zhiwei Xiong.)*

The authors are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230026, China and also with the Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei 230088, China (e-mail: hubosist@mail.ustc.edu.cn; slzhou96@mail.ustc.edu.cn; zwxiong@ustc.edu.cn; fengwu@ustc.edu.cn).

Digital Object Identifier 10.1109/JBHI.2022.3189696

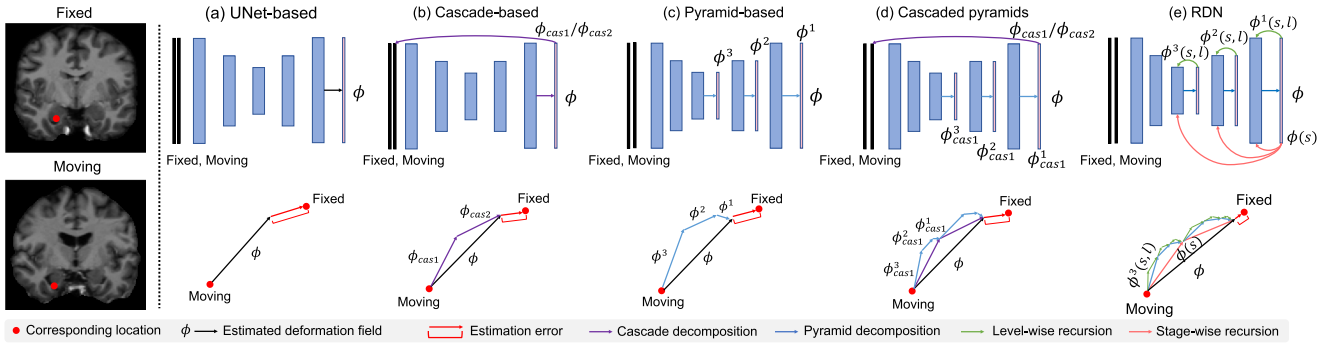

Fig. 1. Illustration of different deformable image registration methods. Left: the fixed image and the moving image. Right: The first row illustrates the network structures of the different methods, and the second row illustrates the corresponding deformation decomposition schemes. (a) UNet-based: direct estimation by a UNet; (b) Cascade-based: equal decomposition via repeating subnetworks. ϕ_{casn} is the deformation component in the n-th cascade; (c) Pyramid-based: coarse-to-fine (unequal) decomposition by a multi-level CNN feature pyramid. ϕ^l is the deformation component at the l-th resolution level; (d) Cascaded pyramid: both unequal and equal decomposition in a cascaded pyramid. ϕ^l_{casn} is the deformation component in the n-th cascade at the l-th resolution level; (e) RDN: both unequal and equal decomposition with level-wise recursion and stage-wise recursion based on a multi-level CNN feature pyramid. $\phi^l(s,r)$ is the deformation component in the s-th stage-wise recursion at the l-th resolution level in the r-th level-wise recursion, and $\phi(s)$ is the deformation component in the s-th stage-wise recursion.

levels and neglect the decomposition inside each level, which makes the deformation component at each level still large and hard to estimate accurately, and the resulting errors are prone to pass to the next levels. In summary, cascade-based methods can sufficiently decompose large deformations yet at the cost of heavy computational consumption and long inference time, while pyramid-based methods adopts the coarse-to-fine strategy yet less sufficient for large deformation decomposition. A simple combination of a cascade-based method and a pyramid-based method is a cascaded pyramid, as shown in Fig. 1(d). Cascaded pyramids inherit the sufficient decomposition of cascade-based methods and the coarse-to-fine strategy of each pyramid subnetwork, but also maintain the cumbersome repeated subnetworks of cascade-based methods, as well as the insufficient deformation decomposition of each pyramid subnetwork.

In this paper, to address both the insufficient and inefficient decomposition problems in current deformation decomposition methods, we propose a recursive decomposition network (RDN) to offer a novel solution for deformable image registration. As shown in Fig. 1(e), the RDN estimates deformation fields with both level-wise recursion and stage-wise recursion based on a multi-level CNN pyramid. The decomposition procedure of the RDN contains three steps. First, stage-wise recursion equally decomposes a large deformation into different pyramid estimation stages. Then, pyramid estimation unequally decomposes the deformation of each stage into different resolution levels in a coarse-to-fine manner. Finally, level-wise recursion further equally decomposes the deformation at each level and estimates it progressively. With this three-step decomposition scheme, a large deformation can be efficiently and sufficiently decomposed into different stages, different resolution levels and inside different resolution levels. And the main contributions can be summarized as follows:

1) We address the insufficient and inefficient problems in current deformation decomposition methods with our proposed RDN.
2) We propose stage-wise recursion to efficiently decompose a large deformation into different pyramid estimation stages without using repeated subnetworks like in cascade-based methods.
3) We propose level-wise recursion to sufficiently decompose the deformation inside each resolution level instead of only one-time estimation like in pyramid-based methods.
4) We provide extensive experiments and ablation studies on two representative datasets to validate the effectiveness and efficiency of our work.

II. Related Work

A. Traditional Methods

Traditional methods regard the registration procedure as an optimization problem and model it with a custom energy function. Given an image pair, they iteratively optimize the function with constraint equations. These methods include the elastic body models [4], [5], free-form deformations with B-Splines [6] and Demons [9]. Many methods have also been proposed to obtain deformation fields within the space of diffeomorphic maps, so that the invertibility properties can be guaranteed. These methods include diffeomorphic B-Splines [7], diffeomorphic Demons [8], large deformation diffeomorphic metric [10] and symmetric image normalization (SyN) [27]. Due to the complicated iterative optimization, all of these traditional methods are very time-consuming with heavy computational burdens [28], which is unfriendly to fast diagnosis scenarios.

B. UNet-Based Methods

With the development of CNNs, the procedures of deformable image registration can be regarded as mappings from inputs to a deformation field. UNet [22], which was first proposed to segment biomedical images first, has seen extensive usage in medical image analysis since it came out. VoxelMorph [28] proposed to estimate a deformation field in an unsupervised end-to-end way. The diffeomorphic version of VoxelMorph [16] preserves the diffeomorphic property by a probabilistic model. PC-Reg [29] utilized the Perception-Correspondence Decoupling and Reverse Teaching with few-shot learning to register images with labels. Kim *et al.* [30] proposed a cycle consistent registration scheme named CycleMorph to force a deformed

image to return to the original image, therefore guaranteeing invertibility. Recently, Tran *et al.* [31] proposed a light-weight deformable registration model (LDR) and an adversarial learning with distilling knowledge algorithm (ALDK) to improve its performance. However, the improvement of the performance of the model LDR depends on strong teacher models. All of the above methods are based on UNet structures and estimate deformation fields with direct schemes. However, this scheme cannot effectively deal with large deformations due to the heavy estimation burden. Unsupervised registration methods which are based on similarity metrics are sensitive to local optimas during optimization [32], especially for large deformations. A large deformation has to be eliminated by a UNet-based method at only one time, which leads the U-Net prone to be stuck in local optimas during the optimization. This non-ideal optimization limits UNet-based methods to deal with large deformations effectively.

C. Deformation Decomposition Methods

A straightforward solution to deal with a large deformation is to decompose it into several small deformations. There are two types of methods for deformation decomposition, cascade-based methods and pyramid-based methods. For cascade-based methods, VTN [23] was proposed as a framework that combines affine transformation and deformable registration together, and its performance was validated for up to three cascades. Recursive cascaded network [15] takes VoxelMorph and VTN as its subnetworks, and network cascading effects in different schemes were analyzed. Yu *et al.* [33] proposed a cyclic teacher-student training strategy to boost the performance of a cascade-based network named Cyclic MPN in the field of cardiac motion estimation, which is close to deformable image registration. However, it retains the drawback of cascade-based methods, and its training strategy is much more complicated than an end-to-end manner. Basically, more cascades make better performance, but the repeated subnetworks in cascade-based methods lead to heavy computational consumption and long inference time. Moreover, existing cascade-based methods only adopt UNets as their subnetworks, and ignore the decomposition inside each subnetwork.

For pyramid-based methods, Dual-PRNet [17] offered a coarse-to-fine approach based on dual stream pyramids. The dual-stream scheme of Dual-PRNet separates the feature learning process from the deformation estimation procedure and prevents interference between them. However, Dual-PRNet only adopts one convolution to estimate the intermediate deformation field at each resolution level, which is insufficiently strong and ignores diffeomorphic properties. LapIRN [25] takes a Laplacian image pyramid as inputs to estimate and refine deformation fields. However, it relies on lots of convolution layers to generate deformation fields from the Laplacian image pyramid, which consumes a large number of GPU memories. Shu *et al.* [26] proposed a pyramid-based network named ULAENet to continuously enhance the spatial transformation. This method focuses on improving its similarity performance and lacks an analysis for diffeomorphic properties. Kang *et al.* [34] further improved Dual-PRNet to DualPR-Net++ with 3D correlation layers and convolutional enhancement. However, the correlation computation also suffers from long inference time. Recently, Lv *et al.* [35] proposed a unified framework for robust brain MRI registration in both progressive and coarse-to-fine manners simultaneously. This method proposed the deformation field integration (DFI) module and the non-rigid feature fusion (NFF) module to decompose a large deformation to several small ones. However, the DFI and the NFF module may influence the continuity of feature and deformation learning. Pyramid-based methods adopt a coarse-to-fine strategy to decompose a large deformation progressively yet ineffectively as the limited number of resolution levels constrains the number of decomposition times. And they also ignore the sufficient deformation decomposition inside each resolution level. In summary, cascade-based methods suffer from the inefficient decomposition problem while pyramid-based methods suffer from the insufficient problem.

III. Method

A. Three-Step Deformation Decomposition

We decompose (abbreviated as decomp.) a deformation in three steps, stage-wise recursion (step 1), pyramid estimation (step 2), and level-wise recursion (step 3). Step 2 is similar as current pyramid-based methods, while step 1 and step 3 proposed by us correspond to stage-wise recursion and level-wise recursion, respectively. We downsample the deformation ϕ to $1/2$ resolution to decrease computational consumption as in previous methods [16], [25].

The objective of stage-wise recursion is the overall deformation ϕ between the fixed image and the moving image. With stage-wise recursion, the deformation ϕ between a pair of input images is decomposed into S equal components $\phi(s)$, where $s \in \{1, 2, \ldots, S\}$, and each component is estimated by a complete coarse-to-fine (pyramid) stage. This step can be formulated as:

$$Step\ 1:\ \phi \xrightarrow{decomp.} \phi(1), \phi(2), \ldots, \phi(S). \tag{1}$$

The objective of pyramid estimation is the deformation $\phi(s)$ of each stage s. With pyramid estimation, each stage deformation $\phi(s)$ is decomposed to L resolution levels unequally. This is similar to current pyramid methods. Each level l is responsible for estimating its level deformation component $\phi^l(s)$, where $l \in \{1, 2, \ldots, L\}$. This step is formulated as:

$$Step\ 2:\ \phi(s) \xrightarrow{decomp.} \phi^1(s), \phi^2(s), \ldots, \phi^L(s). \tag{2}$$

The objective of level-wise recursion is the deformation $\phi^l(s)$ inside a resolution level of a stage. With level-wise recursion, the deformation $\phi^l(s)$ inside a resolution level is decomposed into several equal components $\phi^l(s, r)$, where $r \in \{1, 2, \ldots, R\}$, and each component is estimated by an estimator. This step can be formulated as:

$$Step\ 3:\ \phi^l(s) \xrightarrow{decomp.} \phi^l(s, 1), \phi^l(s, 2), \ldots, \phi^l(s, R). \tag{3}$$

As a consequence, after these three sufficient decomposition steps, each estimator merely needs to estimate a very small deformation $\phi^l(s, r)$ with a light burden.

B. Overall Structure of the RDN

The overall RDN structure is illustrated in Fig. 2, and we summarize all the symbols in our method to ease the reading in

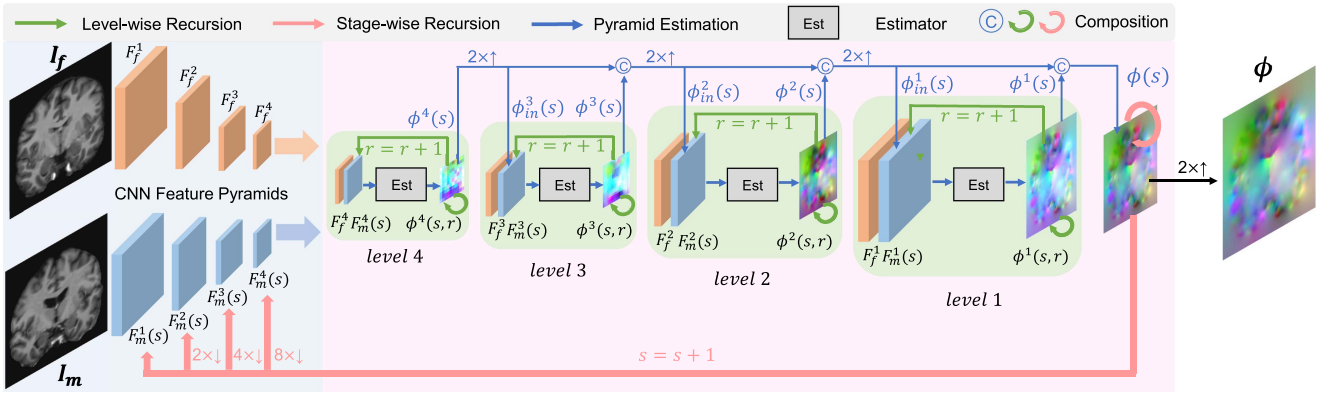

Fig. 2. Overall structure of the RDN. Given a pair of images $\{I_f, I_m\}$, the RDN first extracts the fixed features $\{F_f^1, F_f^2, F_f^3, F_f^4\}$ and the moving features $\{F_m^1, F_m^2, F_m^3, F_m^4\}$ through two groups of four weight-sharing convolutions. Then, the RDN estimates deformation fields with stage-wise recursion and level-wise recursion. For each stage-wise recurion $s = \{1, 2, \ldots, S\}$, the RDN utilizes the level-wise recursion estimation for R times inside each resolution level $l = \{1, 2, 3, 4\}$ and obtain $\{\phi^l(s,1), \phi^l(s,2), \ldots, \phi^l(s,R)\}$ based on the corresponding features $\{F_f^l, F_m^l(s)\}$. The composition of $\{\phi^l(s,1), \phi^l(s,2), \ldots, \phi^l(s,R)\}$ is $\phi^l(s)$, which is the output of the level l. We composite it with previous outputs and upsample the result to obtain the initial deformation field $\phi_{in}^{l-1}(s)$ for the next level $l-1$. After a complete pyramid estimation step, we obtain $\phi(s)$ and take it as the output for the stage-wise recursion s. Finally, $\{\phi(1), \phi(2), \ldots, \phi(S)\}$ for all stages are composed together with upsampling to obtain the final output ϕ. For simplicity, we illustrate the structure of RDN with 2D images (3D images actually).

TABLE I
EXPLANATIONS OF SYMBOLS

I_f / I_m	Fixed / Moving image
$\mathbf{F}_f$	Multi-level fixed features set
F_f^l	Fixed features at the l-th resolution level
$\mathbf{F}_m$	Multi-level moving features set
F_m^l	Moving features at the l-th resolution level
$\mathbf{F}_m(s)$	Multi-level moving features set at the s-th stage recursion
$F_m^l(s)$	Moving features at the l-th pyramid level in the s-th stage-wise recursion
$F_m^l(s,r)$	Moving features in the r-th level-wise recursion at the l-th pyramid level in the s-th stage-wise recursion
ϕ	Final output deformation field
$\mathbf{\Phi}(s)$	Multi-level deformation field set in the s-th stage recursion
$\phi(s)$	Deformation field in the s-th stage recursion
$\phi^l(s)$	Deformation field at the l-th resolution level in the s-th stage recursion
$\phi^l(s,r)$	Deformation field in the r-th level-wise recursion at the l-th resolution level in the s-th stage-wise recursion

Table I. The RDN takes a fixed image I_f and a moving image I_m as inputs, then extracts their multi-level features respectively through the four weight-sharing convolutional layers of strides of 2 and kernel sizes of 3 with leaky ReLUs. All estimations in RDN are based on these two CNN feature pyramids. The extracted fixed features are $\mathbf{F}_f$ and the moving features are $\mathbf{F}_m$. $\mathbf{F}_f$ and $\mathbf{F}_m$ are the sets for F_f^l and F_m^l respectively, where l denotes the l-th level of the pyramid and $l \in \{1, 2, \ldots, L\}$. L is the number of resolution levels, and the finest level is 1 while the coarsest level is L. More resolution levels provide the larger receptive field for CNNs, but result in longer inference time as well. We set $L = 4$ by balancing the performance and inference time, and experiments about different number of levels are provided in Section V-C-3). In the l-th level, the feature resolution is $1/2^l$ of the input images. The RDN aims to estimate a deformation field ϕ to align I_m with I_f. By leveraging extracted features $\mathbf{F}_f$ and $\mathbf{F}_m$, the RDN estimates ϕ with S stage-wise recursions, L resolution levels for each stage and R level-wise recursions inside each level. Here we introduce the workflow of the RDN from the start.

For the first stage-wise recursion 1, the input is the fixed features $\mathbf{F}_f$ and the moving features $\mathbf{F}_m(1)$, where $\mathbf{F}_m(1) = \mathbf{F}_m$. At the coarsest level L, we feed the concatenation of fixed features and moving features $\{F_f^L, F_m^L(1)\}$ as inputs. Then the estimators estimate a series of deformation fields $\{\phi^L(1,1), \phi^L(1,2), \ldots, \phi^L(1,R)\}$ through the proposed level-wise recursion (details are in Method C). We composite [23] these R deformation fields to obtain the deformation field $\phi^L(1)$, which is the output of level L. Then, we upsample $\phi^L(1)$ by a factor of 2 as $\phi_{2\times\uparrow}^L(1)$, and regard $\phi_{2\times\uparrow}^L(1)$ as the initial deformation field $\phi_{in}^{L-1}(1)$ for the next level $L-1$.

For the first stage-wise recursion 1 at level $L-1$, we first warp the moving features $F_m^{L-1}(1)$ with the initial deformation field $\phi_{in}^{L-1}(1)$ as the warped moving features $F_m^{L-1}(1)$ and feed the concatenation of $\{F_f^{L-1}, F_m^{L-1}(1)\}$ as inputs. Then, we can get the deformation field $\phi^{L-1}(1)$ as the output of the level $L-1$ through level-wise recursion following a similar procedure. Before moving to the next level, different from level L, we composite $\phi_{in}^{L-1}(1)$ with $\phi^{L-1}(1)$ and upsample the result by a factor of 2 to obtain $\phi_{in}^{L-2}(1)$, which is the initial deformation field for the next level $L-2$. The estimation procedures at the following levels $l \in \{L-2, L-3, \ldots, 1\}$ for the stage-wise recursion 1 are similar to level $L-1$, and we can get the output of the stage-wise recursion 1 as $\phi(1)$.

We give the estimation procedure of the RDN for the first stage-wise recursion 1 above, and the following stage-wise recursions $s \in \{2, 3, \ldots, S\}$ are similar to the first one. Through stage-wise recursion (details are in Method D), we obtain the outputs $\{\phi(1), \phi(2), \ldots, \phi(S)\}$. Then, we composite these S deformation fields together and upsample the result by a factor of 2 to obtain ϕ as the output of the overall RDN.

C. Level-Wise Recursion

We illustrate level-wise recursion in Fig. 3. The RDN recursively estimates a deformation field at each resolution level with

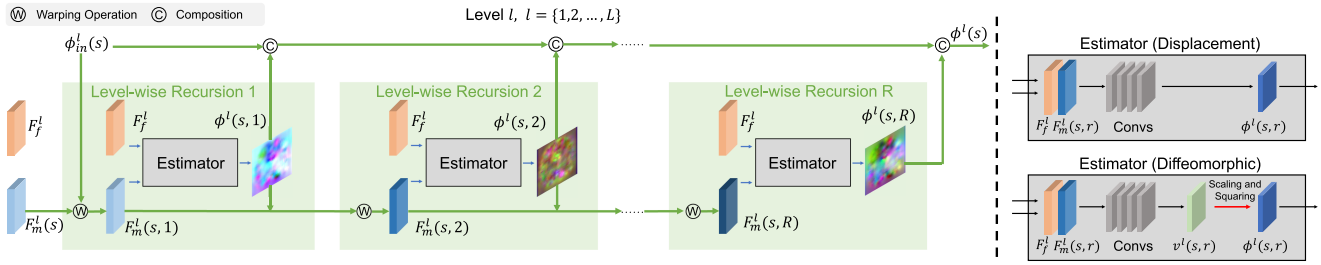

Fig. 3. Illustration of level-wise recursion (left) and the structure of the estimator (right). At each resolution level $l = \{1, 2, \ldots, L\}$ of each stage, we estimate the deformation field $\phi^l(s)$ with R level-wise recursions through R corresponding estimators. Each estimator is structurally the same. Regarding an estimator, the upper version corresponds to the displacement version and the lower version corresponds to the diffeomorphic version.

a group of estimators, and each estimator consists of four convolutions and a Leaky ReLU. We take the stage-wise recursion s and level l as an example here. We first feed the concatenated features $\{F_f^l, F_m^l(s)\}$ at the level l as the input, where $F_m^l(s)$ denotes the moving features for stage s. Then, we use the initial deformation field $\phi_{in}^l(s)$ passed from the previous level to warp the moving features $F_m^l(s)$, and we denote the result as $F_m^l(s,1)$. For the coarest level L, there is no initial deformation field and $F_m^L(s,1)$ is equal to $F_m^L(s)$. After that, we feed the concatenated features $\{F_f^l,\ F_m^l(s,1)\}$ to the first estimator to estimate the deformation field $\phi^l(s,1)$. The procedure can be formulated as:

$$\begin{aligned} \phi^l(s,1) &= E(F_f^l,\ F_m^l(s,1)), \\ F_m^l(s,2) &= F_m^l(s,1) \circ \phi^l(s,1), \end{aligned} \tag{4}$$

where we use E to denote the estimator estimation, and use $\circ$ to denote the warp operation.

For the second level-wise recursion 2, we concatenate $F_m^l(s,2)$ with the same fixed features F_f^l and feed them to the second estimator, which estimates the second deformation field $\phi^l(s,2)$. Then, we employ $\phi^l(s,2)$ to warp $F_m^l(s,2)$, and obtain $F_m^l(s,3)$. The procedures of the third level-wise recursion and beyond are similar, which can be formulated as:

$$\begin{aligned} \phi^l(s,2) &= E(F_f^l,\ F_m^l(s,2)), \\ F_m^l(s,3) &= F_m^l(s,2) \circ \phi^l(s,2), \\ &\ldots \\ \phi^l(s,R) &= E(F_f^l,\ F_m^l(s,R)). \end{aligned} \tag{5}$$

Finally, we composite $\phi^l(s,1), \phi^l(s,2), \ldots, \phi^l(s,R)$ together, and obtain the deformation field for level l:

$$\phi^l(s) = comp(\phi^l(s,1), \phi^l(s,2), \ldots, \phi^l(s,R)), \tag{6}$$

where $comp()$ denotes the composition operation.

With this scheme, the deformation component of each resolution level is decomposed and estimated with R recursions, instead of being estimated only once and directly passing to the next level as performed in existing pyramid-based methods. Therefore level-wise recursion overcomes the drawback of insufficient decomposition inside each resolution level in existing pyramid-based methods.

D. Stage-Wise Recursion

In addtion to the level-wise recursion at each resolution level of a pyramid estimation, the RDN recursively estimates deformation fields for S stages of pyramid estimation. We illustrate stage-wise recursion in Fig. 4.

For the first stage-wise recursion 1, we take the multi-level fixed features $\mathbf{F}_f = \{F_f^1, F_f^2, \ldots, F_f^L\}$ along with the multi-level moving features $\mathbf{F}_m(1) = \{F_m^1(1), F_m^2(1), \ldots, F_m^L(1)\}$ as inputs. By a complete stage of the pyramid estimation, we obtain the deformation field $\phi(1)$ as the output. Then, we downsample $\phi(1)$ by a factor of $\{2^1, \ldots, 2^{L-1}\}$ to obtain $\mathbf{\Phi}(1) = \{\phi(1), \phi(1)_{2\times\downarrow}, \ldots, \phi(1)_{2^{L-1}\times\downarrow}\}$. After that, we employ $\mathbf{\Phi}(1)$ to warp the features $\mathbf{F}_m(1)$ at corresponding levels and obtain the new multi-level moving features $\mathbf{F}_m(2) = \{F_m^1(2), F_m^2(2), \ldots, F_m^L(2)\}$ for the second stage as follows,

$$\begin{aligned} \phi(1) &= P(\mathbf{F}_f, \mathbf{F}_m(1)), \\ \mathbf{\Phi}(1) &= \{\phi(1), \phi(1)_{2\times\downarrow}, \ldots, \phi(1)_{2^{L-1}\times\downarrow}\}, \\ \mathbf{F}_m(2) &= \mathbf{F}_m(1) \circ \mathbf{\Phi}(1), \end{aligned} \tag{7}$$

where we use P to denote the pyramid estimation and use $2^l\times\downarrow$ to denote the downsample operation of 2^l factor.

For the second stage-wise recursion 2, we utilize these warped multi-level moving features $\mathbf{F}_m(2)$ together with the same multi-level fixed features F_f to estimate the deformation field $\phi(2)$. The process of the second stage-wise estimation and beyond is similar to that of the first stage-wise recursion, which can be formulated as:

$$\begin{aligned} \phi(2) &= P(\mathbf{F}_f,\ \mathbf{F}_m(2)), \\ \mathbf{\Phi}(2) &= \{\phi(2), \phi(2)_{2\times\downarrow}, \ldots, \phi(2)_{2^{L-1}\times\downarrow}\}, \\ \mathbf{F}_m(3) &= \mathbf{F}_m(2) \circ \mathbf{\Phi}(2). \\ &\ldots \\ \phi(S) &= P(\mathbf{F}_f,\ \mathbf{F}_m(S)). \end{aligned} \tag{8}$$

At the end of the final stage-wise recursion S, we composite the deformation fields of all stage-wise recursions $\{\phi(1), \phi(2), \ldots, \phi(S)\}$ together, and then upsample the result by a factor of 2 (denoted as $2\times\uparrow$) to obtain the output of the RDN as following,

$$\phi = comp(\phi(1), \phi(2), \ldots, \phi(S))_{2\times\uparrow}. \tag{9}$$

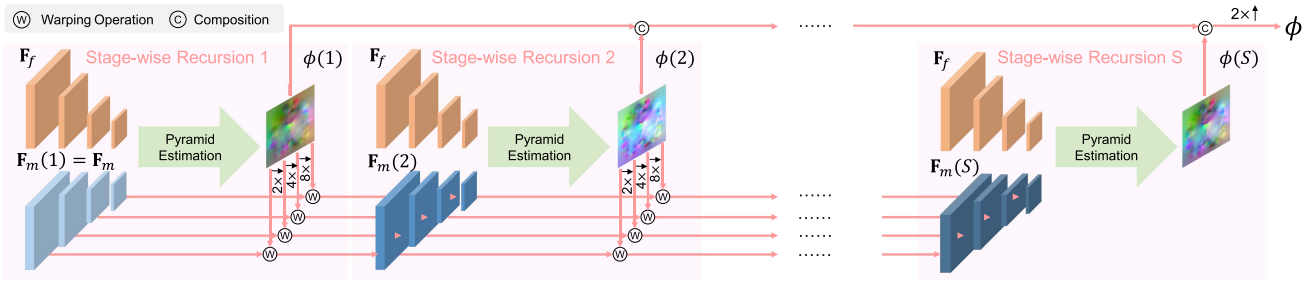

Fig. 4. Illustration of stage-wise recursion. For each stage-wise recursion s, we warp the fixed features $\mathbf{F}_f = \{F_f^1, F_f^2, \ldots, F_f^L\}$ and the moving features $\mathbf{F}_m(s) = \{F_m^1(s), F_m^2(s), \ldots, F_m^L(s)\}$ by the deformation field $\phi(s-1)$ from the previous stage-wise recursion $s-1$. Before warping operation, we downsample $\phi(s-1)$ by the factors of $1, 2, 4, 8$ to match the resolutions of corresponding features at different levels.

In existing pyramid-based methods, the number of decomposition is constrained by the limited number of resolution levels. With stage-wise recursion, the number of stages can be arbitrary as long as the resources are allowed theoretically. Therefore stage-wise recursion breaks the limitation of insufficient decomposition in existing pyramid-based methods.

E. Diffeomorphism

We provide two versions of our method, the displacement version RDN and the diffeomorphic version RDN-*diff*. For the displacement version RDN, we parameterize our deformation model with a displacement field under the Euclidean framework, which does not constrain the diffeomorphic properties explicitly. For the diffeomorphic version RDN-*diff*, we parameterize our deformation model using stationary velocity field method [36] under the Log-Euclidean framework, aiming to guarantee the topology preservation and invertibility. Following [16], [25], we define the diffeomorphic deformation field ϕ through the ordinary differential equation about time t:

$$\frac{d\phi^{(t)}}{dt} = v(\phi^{(t)}), \quad s.t. \ \phi^{(0)} = Id, \tag{10}$$

where Id denotes the identity transformation. And we use the scaling and squaring [37] to integrate the stationary velocity field v over $t = [0, 1]$ to obtain the deformation field ϕ.

F. Loss Functions

The loss function $\mathcal{L}$ of the RDN consists of two terms, a dissimilarity penalty term $\mathcal{L}_{sim}$ and a regularization term $\mathcal{L}_{reg}$:

$$\mathcal{L} = \mathcal{L}_{sim} + \lambda \mathcal{L}_{reg}, \tag{11}$$

where λ is a hyperparameter. We empirically choose the same $\lambda = 1$ for all experiments involving the RDN for a fair comparison, and the experiments about different values of λ are provided in Section V-C-5). For the dissimilarity penalty term $\mathcal{L}_{sim}$, we employ the negative local cross-correlation loss [28] to make the appearances of the fixed image and the warped moving image close, which can be formulated as:

$$\mathcal{L}_{sim} = -\frac{Cov(I_m \circ \phi, I_f)}{\sqrt{Cov(I_f, I_f) \cdot Cov(I_m \circ \phi, I_f)}}, \tag{12}$$

where $Cov(\cdot, \cdot)$ denotes the covariance between two images.

For the regularization term $\mathcal{L}_{reg}$, we employ the differences of spatial gradients of the voxels in the deformation field ϕ. As a result, $\mathcal{L}_{reg}$ encourages the ϕ to be smooth spatially and can be formulated as:

$$\mathcal{L}_{reg} = \|\nabla \phi\|^2. \tag{13}$$

In the displacement version RDN, we enforce the regularization on the estimated deformation fields $\phi(s)$ at each stage-wise recursion directly. In the diffeomorphic version RDN-*diff*, we enforce the regularization on the velocity fields $v(s)$.

IV. Experiments

A. Datasets and Preprocessing

We conduct experiments on two public datasets OASIS and Mindboggle-101.

OASIS [38] is a brain MRI dataset that is broadly used to evaluate registration methods. We adopt the Neurite OASIS [39] version, whose preprocessing steps are following HyperMorph [40]. It includes 414 images in total and all images are pre-aligned to a template space. We split OASIS into 380, 14 and 20 images, and use them to construct 380 × 379, 14 × 13 and 20 × 19 image pairs for training, validation and testing respectively. Each image is companied with a segmentation annotation on 35 issues. The resolutions for scans of this dataset are 160 × 192 × 224.

Mindboggle-101 [41] contains 5 subsets, Extra-18, MMRR-21, NKI-RS-22, NKI-TRT-20 and OASIS-TRT-20. Since OASIS-TRT-20 is included in our OASIS dataset, we use NKI-RS-22, NKI-TRT-20 as our training sets (42 × 41 pairs), and split MMRR-21 as our validation set (8 × 7 pairs) and test set (12 × 11 pairs). All of images are pre-aligned to the MNI152 template space. We conduct standard preprocessing steps such as center-cropping and min-max normalization following [28]. The resolutions of the processed scans are 192 × 192 × 192.

B. Implementation

1) Registration Strategy: We utilize the subject-to-subject registration strategy for our experiments. Subject-to-subject registration takes two arbitrary images as inputs to register one of them to the other. In this strategy, the fixed image is not set in advance, enabling this approach better reflect the robust registration abilities of methods in comparison with that of the atlas-based strategy.

2) Training Strategies: In all experiments, we employ the Adam [42] optimizer with the first momentum of 0.9, the second

momentum of 0.99. Adam converges quickly and accurately, making it suitable for optimizing the complicated learning procedure of deformable image registration. We set the number of optimization steps as 10^5 for all methods in the training phase and set the initial learning rate as 10^{-4}. For better convergence, we halve the learning rate halves after 6×10^4 steps and halves again after 8×10^4 steps.

3) Environment: For the CNN-based methods, we utilize the PyTorch 1.1 as the basic experiment environment. We set the batch size as 4 on 4 NVIDIA Titan XP GPUs with 12 GB memories. For the traditional methods, we implement them on CPUs of Xeon E5-2650 V4 for testing the inference speed where GPU is not applicable.

C. Evaluation Metrics

1) Similarity Metric: We adopt the Dice score as our main similarity metric, as it is designed to measure the overlap degree between two regions. When we evaluate scans containing multiple anatomical structures, we compute their Dice scores separately and take their average as the final Dice score. A higher Dice score between two regions denotes a higher degree of their overlap, meaning higher similarity performance. In addtion, we evaluate the 95% maximum Hausdorff distance (abbreviated as HD95) on various anatomical areas as an additional similarity metric. HD95 can reflect the degree of the estimation errors between two corresponding anatomical area outlines [43]. Therefore, the lower HD95 value is, the better. A lower HD95 means that corresponding locations between the registered moving image and the fixed image are closer.

2) Diffeomorphic Metric: In addition to the Dice score, the diffeomorphic property is anther important measurement for evaluating the performance of registration methods. We adopt the number of negative Jacobian determinants of the deformation field (denoted as Folds) as our quantitative metric for the diffeomorphic property. Jacobian matrices encode the local stretching, shearing and rotating of the deformation field, and their determinants indicate relative volumes before and after spatially transforming. A region of negative determinants indicates that the one-to-one mapping has been lost [36]. So fewer Folds denote better diffeomorphic property.

3) Voxel-Intensity Metrics: We also evaluate the methods in mean square error (MSE) and normalization correlation coefficient (NCC) [28]. They are metrics concerning voxel intensity consistency. For better presentation, we multiply all the values of MSE with an uniform coefficient 10^{-3}, and adopt the NCC errors (NCCE) to replace NCC. The values of NCCE are obtained by 1 minus the values of NCC. Lower MSE and NCCE denote better consistency on voxels within the registered image pair.

4) Efficiency Metrics: We provide four efficiency metrics for the methods: the floating point operations per second (FLOPs), the number of parameters (Params.), the consumption of GPU memories during inference, and the consumption of time during inference. These metrics measure the efficiency of methods in different aspects including space and time complexity.

D. Baseline Methods

We categorize our baseline methods into displacement methods and diffeomorphic methods according to their model parameterization approach. We provide traditional methods, and CNN-based methods including UNet-based methods, deformation decomposition methods. For traditional methods, we implement B-Splines [6] with Elastix [44], and implement SyN [27] with the popular ANTs software package [45]. Both of them are top-performing traditional methods in registration. For CNN-based methods, we implement direct UNet methods VoxelMorph [28], the diffeomorphic version of VoxelMorph [16] (abbreviated as VM and VM-*diff*), VTN [23], and CycleMorph [30]. Furthermore, we implement the method LDR-ALDK [31] for a comparison, which is a novel UNet-based method under the guidance of knowledge distillation. In terms of current deformation decomposition methods, regarding the cascade-based methods, we provide n-cascade VM, VM-*diff*, and VTN according to [15], where n is limited by the 12 GB GPU memory of the Titan XP during training.

We also implement Cyclic MPN [33] on our datasets, which is proposed in the field of cardiac motion estimation, which is close to deformable image registration. Regarding the pyramid-based methods, we implement the representative methods including Dual-PRNet [17], LapIRN [25] along with its diffeomorphic version LapIRN-*diff* and ULAENet [26]. Furthermore, we implement the state-of-the-art methods recently following the descriptions in their papers, including Dual-PRNet++ [34] which is the extension of Dual-PRNet and PCNet [35] which is a recent high-performance pyramid-based method.

V. Results

A. Comparison With Baseline Methods

Table II gives a summary of the quantitative results. For convenient comparison, we arrange the results of the traditional methods, CNN-based UNet methods and pyramid-based methods together, and we arrange the results of the CNN-based n-cascade methods with our proposed RDN(S, $RRRR$) together. The S of RDN(S, $RRRR$) denotes the number of stage-wise recursions, while the $RRRR$ denotes that the number of level-wise recursions at the resolution level 4,3,2 and 1, respectively.

1) Similarity & Diffeomorphic Properites: We first analyze the Dice scores and number of Folds. For the traditional method, RDN(4, 4444) improves upon the performance of B-Splines by large margins in terms of both Dice and Folds. For the CNN-based methods, compared with the UNet methods VM and LDR-ALDK, the RDN achieves greatly improved performance. The Dice score is increased from 0.756 of VM and 0.742 of LDR-ALDK to 0.812 of RDN(4, 4444) on OASIS. And the Folds are decreased from the 3183 of VM and 4027 of LDR-ALDK to the 23 of RDN(4, 4444) on OASIS, which proves that our RDN preserves the diffeomorphic property more effectively. Compared with pyramid-based methods, taking the results of diffeomorphic methods to analyze, a Dice score of 0.808 of our RDN-*diff*(4, 4444) is still better than a Dice score of 0.804 of PCNet on OASIS, which is the best-performed pyramid-based method. Our RDN advantage is consistent with the results on Mindboggle-101. Compared with the n-cascade methods, our RDN performs better than them as well. Cascade-based methods with large numbers of cascades such as 4-cascade VM, 3-cascade VM-*diff* and 4-cascade VM-*diff* are not able to train since their large computational consumptions (GPU memories) are beyond the 12 GB limit of Titan XP, while the RDN with recursions is able to run under the same condition.

TABLE II

COMPARISON WITH THE BASELINE METHODS ON OASIS AND MINDBOGGLE-101. TRA., UNET, PYR. AND CAS. ARE THE ABBREVIATIONS FOR TRADITIONAL METHODS, UNET-BASED METHODS, PYRAMID-BASED METHODS AND CASCADE-BASED METHODS, RESPECTIVELY. THE STANDARD DEVIATION IS IN PARENTHESES, AND BOLD FONTS MEAN THE BEST IN COMPARISON. HIGHER DICE DENOTES BETTER PERFORMANCE WHILE LOWER FOLDS, HD95, MSE AND NCCE DENOTE BETTER PERFORMANCE

		OASIS					Mindboggle-101				
Type	Name	Dice	Folds	HD95	MSE	NCCE	Dice	Folds	HD95	MSE	NCCE
						Displacement Methods					
Tra.	B-Splines	0.746 (0.032)	2681 (2251)	2.039 (0.301)	1.582 (0.143)	0.053 (0.006)	0.544 (0.021)	1532 (1388)	5.878 (0.507)	8.912 (0.912)	0.071 (0.005)
UNet	VM	0.756 (0.028)	3183 (1410)	1.732 (0.296)	1.027 (0.085)	0.033 (0.004)	0.556 (0.018)	2809 (1429)	5.023 (0.402)	3.676 (0.558)	0.038 (0.003)
UNet	VTN	0.725 (0.031)	3106 (1344)	1.913 (0.274)	1.440 (0.111)	0.046 (0.005)	0.509 (0.014)	528 (1496)	5.158 (0.388)	5.595 (0.473)	0.060 (0.004)
UNet	CycleMorph	0.772 (0.029)	4264 (2516)	1.889 (0.304)	1.006 (0.071)	0.033 (0.004)	0.567 (0.019)	2975 (1763)	4.931 (0.394)	5.186 (0.487)	0.054 (0.003)
UNet	LDR-ALDK	0.742 (0.026)	4027 (2654)	2.012 (0.309)	1.371 (0.103)	0.043 (0.005)	0.539 (0.018)	2206 (1269)	5.326 (0.438)	5.912 (0.561)	0.062 (0.004)
Pyr.	Dual-PRNet	0.734 (0.031)	3931 (1796)	1.831 (0.244)	1.459 (0.120)	0.045 (0.005)	0.521 (0.011)	1586 (2867)	4.818 (0.326)	5.488 (0.504)	0.058 (0.004)
Pyr.	LapIRN	0.802 (0.023)	2325 (1768)	1.695 (0.287)	0.863 (0.095)	0.029 (0.004)	0.616 (0.013)	1999 (1673)	4.927 (0.380)	3.986 (0.639)	0.032 (0.003)
Pyr.	Dual-PRNet++	0.804 (0.022)	1986 (1031)	1.628 (0.270)	1.107 (0.103)	0.033 (0.004)	0.618 (0.012)	1063 (688)	4.704 (0.351)	3.413 (0.486)	0.041 (0.003)
Pyr.	ULAENet	0.803 (0.023)	2276 (1597)	1.716 (0.298)	0.951 (0.084)	0.028 (0.004)	0.620 (0.014)	1648 (874)	5.003 (0.362)	3.612 (0.535)	0.036 (0.004)
Cas.	2-cascade VM	0.796 (0.023)	673 (462)	1.495 (0.236)	0.611 (0.054)	0.020 (0.002)	0.605 (0.011)	201 (288)	4.788 (0.372)	2.281 (0.664)	0.021 (0.002)
Cas.	2-cascade VTN	0.785 (0.024)	621 (446)	1.556 (0.228)	0.801 (0.065)	0.026 (0.003)	0.571 (0.011)	51 (80)	4.841 (0.366)	3.409 (0.487)	0.034 (0.003)
Cas.	Cyclic MPN	0.801 (0.022)	367 (405)	1.502 (0.267)	0.725 (0.081)	0.023 (0.003)	0.616 (0.011)	55 (39)	4.627 (0.392)	2.782 (0.611)	0.029 (0.002)
Ours	**RDN(2, 2222)**	**0.805 (0.021)**	**83 (100)**	**1.488 (0.265)**	**0.599 (0.054)**	**0.019 (0.002)**	**0.622 (0.010)**	**31 (28)**	**4.412 (0.317)**	**2.217 (0.641)**	**0.020 (0.001)**
Cas.	3-cascade VM	0.805 (0.024)	267 (152)	1.462 (0.242)	0.486 (0.045)	0.017 (0.002)	0.622 (0.011)	92 (56)	4.647 (0.362)	1.901 (0.707)	0.017 (0.001)
Cas.	3-cascade VTN	0.802 (0.024)	247 (187)	1.675 (0.243)	0.662 (0.059)	0.020 (0.003)	0.600 (0.012)	41 (33)	4.639 (0.360)	2.459 (0.548)	0.024 (0.002)
Ours	**RDN(3, 3333)**	**0.811 (0.021)**	**33 (49)**	**1.448 (0.250)**	**0.504 (0.047)**	**0.016 (0.002)**	**0.632 (0.010)**	**24 (15)**	**4.401 (0.303)**	**1.830 (0.642)**	**0.015 (0.001)**
Cas.	4-cascade VTN	0.805 (0.020)	125 (56)	1.597 (0.240)	0.615 (0.048)	0.018 (0.002)	0.612 (0.012)	**26 (21)**	4.526 (0.362)	2.237 (0.612)	0.022 (0.002)
Ours	**RDN(4, 4444)**	**0.812 (0.022)**	**23 (38)**	**1.443 (0.256)**	**0.456 (0.044)**	**0.015 (0.002)**	**0.637 (0.011)**	37 (28)	**4.388 (0.325)**	**1.755 (0.649)**	**0.015 (0.001)**
						Diffeomorphic Methods					
Tra.	SyN	0.748 (0.027)	0 (0)	2.106 (0.299)	1.599 (0.127)	0.049 (0.007)	0.549 (0.021)	0 (0)	5.126 (0.338)	6.130 (0.531)	0.058 (0.003)
UNet	VM-*diff*	0.759 (0.028)	0 (0)	1.768 (0.276)	1.094 (0.113)	0.031 (0.005)	0.552 (0.015)	0 (0)	5.011 (0.349)	2.896 (0.615)	0.027 (0.002)
Pyr.	LapIRN-*diff*	0.795 (0.021)	0 (0)	1.709 (0.258)	0.697 (0.108)	0.028 (0.005)	0.606 (0.011)	0 (0)	5.102 (0.358)	4.812 (0.576)	0.030 (0.003)
Pyr.	PCNet	0.804 (0.021)	48 (39)	1.765 (0.286)	0.930 (0.106)	0.031 (0.004)	0.627 (0.013)	48 (61)	4.878 (0.386)	3.891 (0.598)	0.039 (0.003)
Cas.	2-cascade VM-*diff*	0.796 (0.023)	54 (33)	1.685 (0.254)	0.686 (0.076)	0.022 (0.003)	0.599 (0.011)	0 (0)	4.759 (0.348)	2.339 (0.612)	0.025 (0.002)
Ours	RDN-*diff*(2, 2222)	0.800 (0.021)	0 (0)	1.479 (0.259)	0.587 (0.057)	0.019 (0.002)	0.610 (0.008)	0 (0)	4.538 (0.299)	2.601 (0.587)	0.022 (0.002)
Ours	RDN-*diff*(3, 3333)	0.805 (0.021)	0 (0)	1.446 (0.210)	0.501 (0.056)	0.019 (0.002)	0.625 (0.010)	0 (0)	4.450 (0.310)	2.121 (0.599)	0.019 (0.001)
Ours	**RDN-*diff*(4, 4444)**	**0.808 (0.021)**	**0 (0)**	**1.429 (0.225)**	**0.525 (0.050)**	**0.017 (0.002)**	**0.632 (0.011)**	**0 (0)**	**4.411 (0.363)**	**1.939 (0.627)**	**0.017 (0.001)**

2) Voxel-Intensity Properties: In addition to the Dice score and Folds, our RDN(4, 4444) achieves the lowest HD95 of 1.443 on OASIS and 4.388 on Mindboggle-101, which means the estimation errors produced by our RDN on anatomical area outlines are the lowest. Besides, in terms of MSE and NCCE, RDN also performs the best among all baseline methods, which proves the RDN guarantees the consistency of the voxel intensities between the fixed image and the warped moving image. These results can be regarded as a verification for the analysis in Fig. 1. We take a pair of images from OASIS and Mindboggle-101 as examples respectively, and show the visualization results of different methods in Fig. 5. Moreover, we give the boxplots achieved by different methods on OASIS in Fig. 6. Basically, our RDN performs the best on every anatomical structure, which demonstrates RDN has a strong generalization ability.

3) Efficiency Properties: We evaluate the efficiency properties of our RDN and representative baseline methods. For a given CNN model, the number of parameters denotes the storage space that can measure the space complexity, and its FLOPs denote the computational consumption that can measure the time complexity. The consumption of GPU memories and time are commonly used metrics to measure efficiency. Since SyN and B-Splines are traditional methods and lack of GPU implementation, we only evaluate their time here. The evaluation results on Mindboggle-101 are shown in Table III, demonstrating the high efficiency of our proposed RDN. Our RDN decreases the inference time from 2452 s and 353 s to 0.28 s compared with the traditional methods SyN and B-Splines. In comparison with the UNet-based methods VM and VTN, our RDN consumes the least FLOPs and GPU memories. Compared with the representative pyramid-based method LapIRN, our RDN only demands about 20% FLOPs, 30% GPU memories and 30% inference time of it. Furthermore, compared with cascasde-based methods, RDN only consumes less than 2% parameters of 2-cascade VTN and 50% FLOPs of both 2-cascade VM and 2-cascade VTN.

TABLE III

COMPARISON WITH BASELINE METHODS ON EFFICIENCY

Method	FLOPs (G)	Params. (M)	GPU (MB)	Time (s)
SyN	-	-	-	2452
B-Splines	-	-	-	353
VM	192	0.3	4008	0.16
VTN	170	28.3	2440	1.12
LapIRN	848	0.9	6184	0.88
2-cascade VM	384	0.6	4442	0.50
2-cascade VTN	339	56.5	2902	2.23
RDN(2, 2222)	168	1.0	2112	0.28

TABLE IV

EXPERIMENTS OF THE EXPLORATION OF CASCADED PYRAMIDS

	OASIS		Mindboggle-101		
	Dice	Folds	Dice	Folds	Time (s)
2-cascade RDN	0.802 (0.022)	264 (323)	0.616 (0.009)	132 (306)	0.63
RDN(2, 2222)	**0.805 (0.021)**	**83 (100)**	**0.622 (0.010)**	**31 (28)**	**0.28**
3-cascade RDN	0.808 (0.021)	110 (88)	0.627 (0.009)	71 (150)	1.27
RDN(3, 3333)	**0.811 (0.021)**	**33 (49)**	**0.632 (0.010)**	**24 (15)**	**0.63**
4-cascade RDN	0.810 (0.022)	86 (82)	0.633 (0.010)	69 (43)	1.70
RDN(4, 4444)	**0.812 (0.022)**	**23 (38)**	**0.637 (0.011)**	**37 (28)**	**0.95**

B. Exploration of Cascaded Pyramids

We explore the structures of cascaded pyramids as supplements to current deformation decomposition methods. The results are shown in Table IV. We denote the cascaded pyramids as n-cascade RDN, where the pyramid subnetwork is RDN(1, 1111) without recursions for a fair comparison. We can see that our RDN($n, nnnn$) performs better in terms of both similarity

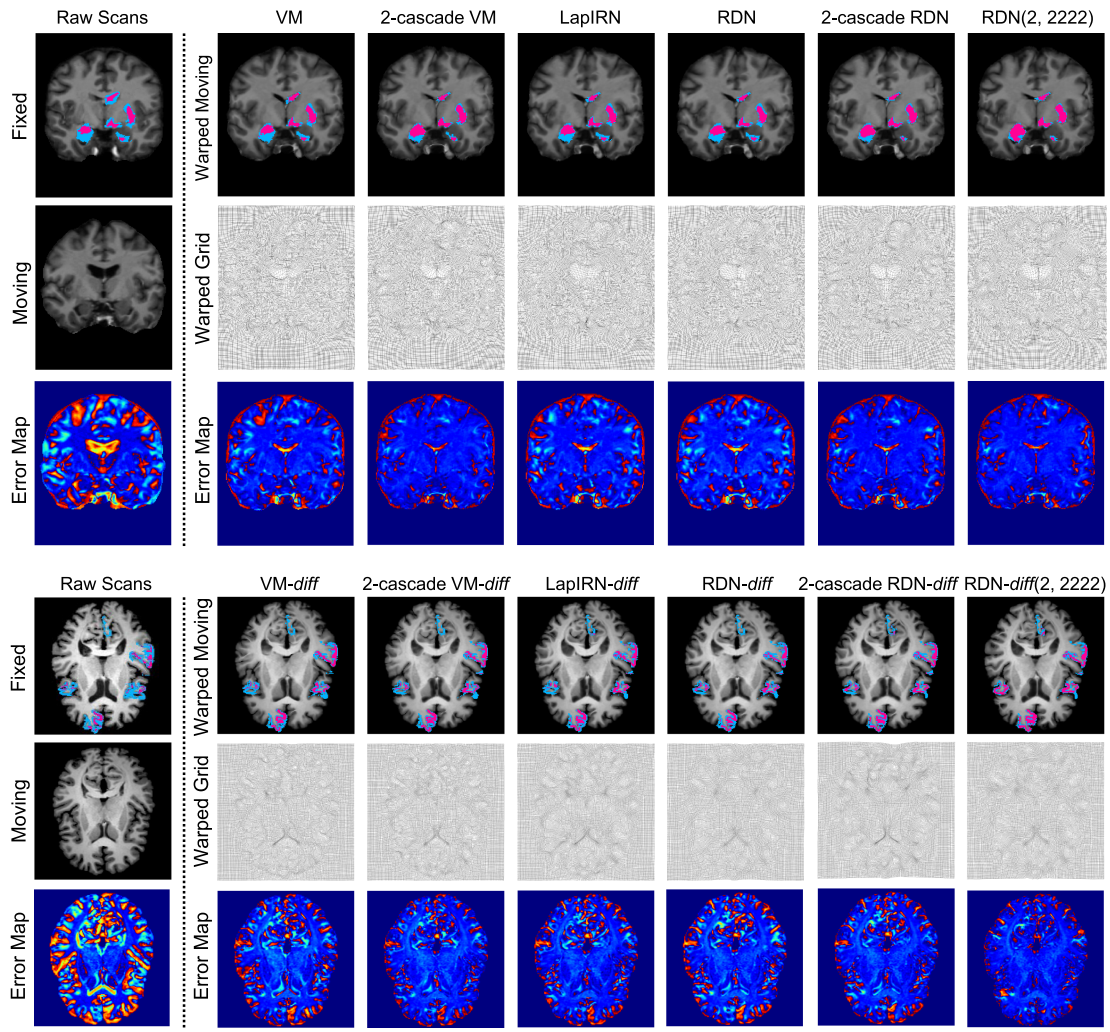

Fig. 5. Visualization of two examples on OASIS (top) and Mindboggle-101 (bottom) respectively. We provide the comparison of displacement methods for OASIS, and comparison of diffeomorphic methods for Mindboggle-101. The first row shows the target fixed image and warped moving images with five representative anatomical segments. The overlap between the fixed image and the (warped) moving image is red, and the rest part is blue. The second row shows the raw moving image on the left, and shows warped grids to visualize diffeomorphic property on the right. The third row shows error maps, which are obtained by taking each (warped) moving image and substracted the fixed image. A redder color represents a larger registration error while a bluer color represents a smaller error. 'RDN' here denotes the RDN(1, 1111) without any recursion.

and diffeomorphic properties, and it consumes less time than the n-cascade RDN. The reason for this is that the structures of cascaded pyramids also retain the drawbacks of pyramid-based and cascaded-based methods. Cascaded pyramids maintain cumbersome repeated subnetworks, which brings in much computational consumption. Moreover, the insufficient decomposition for each resolution level of a pyramid subnetwork is still not solved by cascaded pyramids, meaning that the deformation of each resolution level is still hard to estimate accurately. The results demonstrate that the RDN is a more efficient structure with better performance than cascaded pyramids.

C. Ablation Studies

We design ablation studies to explore the RDN further. We take the displacement version RDN as an example and the regularities are consistent with those of RDN-*diff*.

1) Three Decomposition Steps: We explore the impacts of the three decomposition steps on the RDN. Our three-step decomposition includes stage-wise recursion (SR), pyramid estimation (PE), and level-wise recursion (LR) as mentioned in Section III-A. Since RDN is based on a multi-level pyramid structure, we explore the impacts of stage-wise recursion and level-wise recursion based on the pyramid estimation. We take RDN(2, 2222) as an example to verify the impact of each decomposition step. Both level-wise and stage-wise recursion contributes to the performance of RDN, and using them together can provide much better performance than using just one type of it. Using the level-wise and stage-wise recursion together can combine their respective advantages.The results are shown in Table V.

2) Different Numbers of Recursions: We explore the impacts of different numbers of recursions on the RDN. The results of the number of recursions are shown in Table VI. We observe that the more stage-wise recursions we use, the better performance

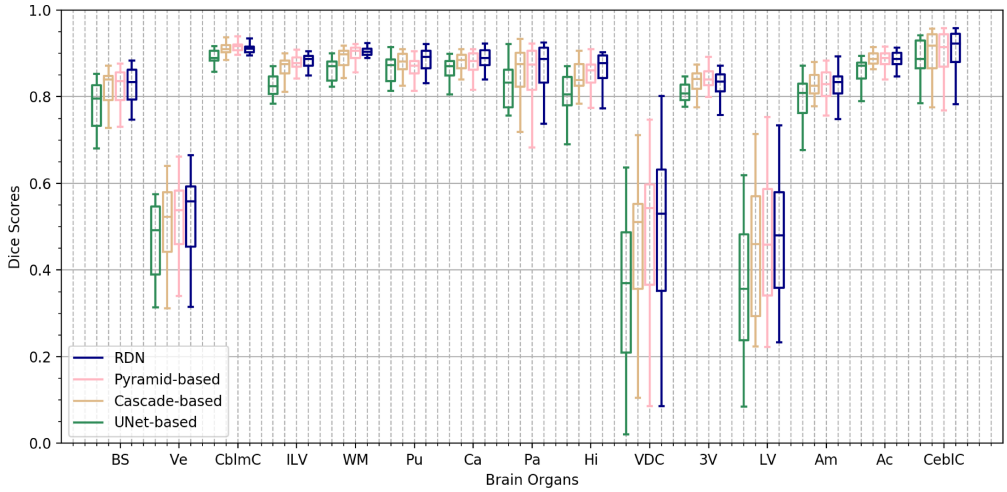

Fig. 6. Boxplots which depict the average Dice scores of each anatomical structure for test pairs on OASIS for different methods. We use RDN (2, 2222) as our method, LapIRN as the pyramid-based method, 2-cascade VoxelMorph as the cascade-based method and VoxelMorph as the direct UNet method. The horizontal axis denotes main anatomical structures: brain stem (BS), vessel (Ve), cerebellum cortex (CblmC), lateral ventricle (LV), cerebellum white matter (WM), putamen (Pu), caudate (Ca), pallidum (Pa), hippocampus (Hi), Ventral DC (VDC), 3 rd ventricle (3 V), amygdala (Am), Ac (Accumbens), and cerebral cortex (CeblC).

TABLE V
EXPERIMENTS ON DIFFERENT DECOMPOSITION STEPS

RDN			OASIS		Mindboggle-101	
PE	LR	SR	Dice	Fold	Dice	Folds
✓			0.789 (0.021)	897 (390)	0.597 (0.010)	997 (318)
✓	✓		0.798 (0.021)	211 (152)	0.604 (0.010)	307 (142)
✓		✓	0.799 (0.022)	**76 (100)**	0.614 (0.009)	**20 (16)**
✓	✓	✓	**0.805 (0.021)**	83 (100)	**0.622 (0.010)**	31 (28)

TABLE VI
EXPERIMENTS ON DIFFERENT NUMBERS OF RECURSIONS

	OASIS		Mindboggle-101		
	Dice	Folds	Dice	Folds	Time (s)
RDN(1, 1111)	0.789 (0.021)	897 (390)	0.597 (0.010)	997 (318)	0.10
RDN(2, 1111)	0.799 (0.022)	76 (100)	0.614 (0.009)	20 (16)	0.16
RDN(3, 1111)	0.802 (0.021)	37 (45)	0.622 (0.010)	12 (10)	0.22
RDN(4, 1111)	**0.804 (0.022)**	**17 (25)**	**0.628 (0.010)**	**9 (7)**	0.27
RDN(1, 2222)	0.798 (0.022)	211 (152)	0.604 (0.010)	307 (142)	0.19
RDN(2, 2222)	0.805 (0.021)	83 (100)	0.622 (0.010)	31 (28)	0.28
RDN(3, 2222)	**0.808 (0.020)**	53 (79)	0.629 (0.010)	25 (37)	0.36
RDN(4, 2222)	**0.808 (0.020)**	**51 (67)**	**0.634 (0.010)**	**17 (16)**	0.51
RDN(1, 3333)	0.799 (0.021)	169 (86)	0.608 (0.010)	225 (167)	0.27
RDN(2, 3333)	0.809 (0.020)	53 (79)	0.625 (0.011)	36 (45)	0.39
RDN(3, 3333)	**0.811 (0.021)**	33 (49)	0.632 (0.010)	24 (15)	0.53
RDN(4, 3333)	**0.811 (0.020)**	**27 (44)**	**0.636 (0.010)**	**21 (15)**	0.76
RDN(1, 4444)	0.801 (0.021)	95 (105)	0.609 (0.011)	56 (28)	0.36
RDN(2, 4444)	0.808 (0.021)	51 (68)	0.628 (0.010)	46 (50)	0.56
RDN(3, 4444)	0.811 (0.021)	23 (40)	0.633 (0.010)	45 (46)	0.81
RDN(4, 4444)	**0.812 (0.022)**	**23 (38)**	**0.637 (0.011)**	**37 (28)**	0.95

RDN achieves, and the more inference time RDN consumes. The growth of time is basically linear with the number of recursions. The regularities are consistent with level-wise recursion. Second, taking the comparison between RDN(2, 2222) with RDN(1, 4444) or RDN(4, 1111) as the example, for the same total number 4 of two types of recursions, the comparison results demonstrate that utilizing level-wise recursion and stage-wise recursion jointly yields greater performance than only using one type of recursion.

TABLE VII
EXPERIMENTS ON DIFFERENT NUMBERS OF RESOLUTION LEVELS

	OASIS			Mindboggle-101		
L	Dice	Folds	Time (s)	Dice	Folds	Time (s)
$L = 2$	0.798 (0.023)	51 (94)	0.18	0.613 (0.010)	16 (8)	0.18
$L = 3$	0.801 (0.021)	62 (86)	0.24	0.617 (0.010)	22 (30)	0.25
$L = 4$	0.805 (0.021)	83 (100)	0.27	0.622 (0.010)	31 (28)	0.28
$L = 5$	0.805 (0.021)	142 (136)	0.28	0.621 (0.010)	94 (125)	0.29

3) Different Numbers of Resolution Levels: In pyramid estimation, the deformation component in each stage is decomposed to several resolution levels l, where $l = 1, 2, \ldots, L$. Here we explore the impacts of the different number of resolution levels L in pyramid estimation, and the results are shown in Table VII. More resolution levels provide a larger receptive field for deformation estimation. However, more resolution levels do not always produce better results. First, too large receptive fields are not meaningful since the actual deformation range is limited. Second, too many resolution levels damage the consistency of deformation estimation and lead to poor diffeomorphic properties. We therefore set $L = 4$ as default.

4) Level-Wise Recursion at Different Levels: We explore the impacts of level-wise recursion on different levels and the results are shown in Table VIII. We can see that recursion on finer levels matters more. Beyond that, the more levels at which we apply recursions, the better performance we obtain. Although recursion at coarse levels yields few gains, it facilitates the learning process for the subsequent levels and improves the performance of the following recursive levels.

5) Hyperparameter of the Regularization Term: We experiment with different values of λ by taking RDN(2, 2222) as an example, and the results on the validation set are shown in Table IX. A higher value for λ can guarantee the diffeomorphic properties better, but too high values will harm the similarity

TABLE VIII
EXPERIMENTS ON LEVEL-WISE RECURSION AT DIFFERENT LEVELS

	OASIS		Mindboggle-101	
	Dice	Folds	Dice	Folds
RDN(1, 1111)	0.789 (0.021)	897 (390)	0.597 (0.010)	997 (318)
RDN(1, 2111)	0.789 (0.021)	879 (384)	0.597 (0.009)	965 (360)
RDN(1, 1211)	0.790 (0.021)	761 (364)	0.598 (0.010)	642 (192)
RDN(1, 1121)	0.791 (0.021)	593 (270)	0.599 (0.010)	596 (180)
RDN(1, 1112)	0.795 (0.022)	277 (203)	0.602 (0.010)	485 (289)
RDN(1, 1122)	0.797 (0.022)	260 (191)	0.603 (0.009)	399 (186)
RDN(1, 1222)	0.797 (0.021)	254 (189)	0.604 (0.009)	285 (192)
RDN(1, 2222)	**0.798 (0.022)**	**211 (173)**	**0.604 (0.010)**	**207 (142)**

TABLE IX
EXPERIMENTS ON DIFFERENT VALUES OF THE HYPERPARAMETER λ

	OASIS		Mindboggle-101	
λ	Dice	Folds	Dice	Folds
λ =0.1	0.800 (0.022)	5232 (1557)	0.649 (0.010)	5372 (715)
λ =0.5	0.810 (0.022)	223 (149)	0.644 (0.010)	399 (220)
λ =1 (ours)	0.807 (0.021)	46 (58)	0.630 (0.010)	22 (16)
λ =2	0.794 (0.021)	7 (10)	0.603 (0.009)	2 (5)
λ =10	0.730 (0.030)	0 (0)	0.528 (0.011)	0 (0)

TABLE X
EXPERIMENTS ON WEIGHT-SHARING RECURSION. THE SUBSCRIPT s MEANS WEIGHTS ARE SHARED AMONG STAGES OR LEVELS OR BOTH

	OASIS		Mindboggle-101		
	Dice	Folds	Dice	Folds	Params
RDN(2, $2_s2_s2_s2_s$)	0.804 (0.021)	10 (21)	0.619 (0.010)	4 (5)	0.56M
RDN(2, $3_s3_s3_s3_s$)	0.806 (0.020)	5 (9)	0.620 (0.009)	10 (11)	0.56M
RDN (2, $4_s4_s4_s4_s$)	**0.806 (0.020)**	**4 (7)**	**0.621 (0.010)**	**3 (4)**	0.56M
RDN(2_s, 2222)	0.802 (0.020)	16 (37)	0.619 (0.010)	22 (30)	0.56M
RDN(3_s, 2222)	0.803 (0.021)	11 (18)	0.628 (0.010)	11 (23)	0.56M
RDN (4_s, 2222)	**0.805 (0.021)**	**7 (17)**	**0.631 (0.010)**	**5 (7)**	0.56M
RDN(2_s, $2_s2_s2_s2_s$)	0.803 (0.021)	7 (16)	0.612 (0.010)	1 (1)	0.34M
RDN(2_s, $3_s3_s3_s3_s$)	0.803 (0.021)	4 (16)	0.613 (0.010)	0 (1)	0.34M
RDN(2_s, $4_s4_s4_s4_s$)	**0.804 (0.020)**	**3 (10)**	**0.615 (0.010)**	**0 (0)**	0.34M
RDN(2_s, $2_s2_s2_s2_s$)	0.803 (0.021)	4 (8)	0.612 (0.010)	1 (1)	0.34M
RDN(3_s, $2_s2_s2_s2_s$)	0.803 (0.020)	4 (5)	0.621 (0.010)	0 (1)	0.34M
RDN(4_s, $2_s2_s2_s2_s$)	**0.804 (0.021)**	**3 (9)**	**0.627 (0.010)**	**0 (0)**	0.34M

properties. Our setting of $\lambda = 1$ balances similarity and diffeomorphic properties effectively.

6) Weight-Sharing Recursion: We also explore a weight-sharing design for level-wise recursion and stage-wise recursion of RDN respectively, as shown in Table X. The results show that the performance improvement tendency provided by the recursions is similar to that of the RDN without sharing weights. Although the overall performance drops slightly, the number of parameters in the CNN models is greatly reduced, which is helpful to storage-constraint scenarios.

7) Different Training Strategies: We adopt Adam [42] as the optimizer following [25], [28]. Here we explore the effectiveness of different optimizers in terms of training RDN(2, 2222) on OASIS as an example. We adopt SGD [46] with/without a momentum term 0.9 and RMSProps [47] with $\beta = 0.999$ as our comparative optimizers. The results are shown in Table XI. We can see that the training procedure adopting SGD without momentum can hardly converge, and the training procedure adopting SGD with momentum or RMSProps perform worse than Adam. Meanwhile, adopting Adam as the optimizer converges faster than adopting SGD or RMSProps. We also provide ablation studies on our learning rate strategy. We conduct the experiment with using a higher learning rate 10^{-3}, and conduct the experiment without the halving of the learning rate. The results demonstrate our setting can obtain the highest performance.

TABLE XI
EXPERIMENTS WITH DIFFERENT TRAINING STRATEGIES

	OASIS		
	Dice	Folds	Convergence Steps
SGD w/o momentum	0.497 (0.056)	0 (0)	Not Convergence
SGD w/ momentum	0.803 (0.021)	91 (99)	98000
RMSProps	0.802 (0.021)	86 (72)	93500
Adam (lr = 10^{-3})	0.776 (0.022)	231 (358)	100000
Adam (w/o lr halving)	0.801 (0.021)	102 (113)	65000
Adam (ours)	**0.805 (0.021)**	**83 (100)**	**86500**

VI. CONCLUSION

In this paper, we propose the RDN to offer a novel solution for deformable image registration which addresses the insufficient and inefficient problems in current deformation decomposition methods. Specifically, we propose stage-wise recursion and level-wise recursion based on a multi-level CNN pyramid to decompose a large deformation into different stages, into different resolution levels and inside different resolution levels. With our proposed RDN, downstream clinical analyses such as the measurement of tumor sizes and the segmentation of anatomical areas can be conducted efficiently. Since our RDN is based on a multi-level CNN pyramid, the deformation fields are connected by the trilinear interpolation between adjacent pyramid resolution levels and adjacent stage-wise recursions. This non-adaptive trilinear interpolation in RDN might cause interpolation artifacts. In the future, we plan to propose an adaptive interpolation module to overcome the interpolation artifacts problem, and employ the deformation field at the fine resolution level to guide the learning of the deformation field at the coarse resolution level.

REFERENCES

[1] J. Du, W. Li, K. Lu, and B. Xiao, "An overview of multi-modal medical image fusion," *Neurocomputing*, vol. 215, pp. 3–20, 2016.
[2] Z. Xu and M. Niethammer, "DeepAtlas: Joint semi-supervised learning of image registration and segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2019, pp. 420–429.
[3] M. Li, S. Zhou, C. Chen, Y. Zhang, D. Liu, and Z. Xiong, "Retinal vessel segmentation with pixel-wise adaptive filters," in *Proc. IEEE 19th Int. Symp. Biomed. Imag.*, 2022, pp. 1–5.
[4] R. Bajcsy and S. Kovačič, "Multiresolution elastic matching," *Comput. Vis. Graph. Image Process.*, vol. 46, no. 1, pp. 1–21, 1989.
[5] D. Shen and C. Davatzikos, "HAMMER: Hierarchical attribute matching mechanism for elastic registration," *IEEE Trans. Med. Imag.*, vol. 21, no. 11, pp. 1421–1439, 2002, doi: 10.1109/TMI.2002.803111.
[6] D. Rueckert, L. I. Sonoda, C. Hayes, D. L. Hill, M. O. Leach, and D. J. Hawkes, "Nonrigid registration using free-form deformations: Application to breast MR images," *IEEE Trans. Med. Imag.*, vol. 18, no. 8, pp. 712–721, Aug. 1999.
[7] D. Rueckert, P. Aljabar, R. A. Heckemann, J. V. Hajnal, and A. Hammers, "Diffeomorphic registration using b-splines," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2006, pp. 702–709.
[8] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache, "Diffeomorphic demons: Efficient non-parametric image registration," *NeuroImage*, vol. 45, no. 1, pp. S61–S72, 2009.
[9] J.-P. Thirion, "Image matching as a diffusion process: An analogy with Maxwell's demons," *Med. Image Anal.*, vol. 2, no. 3, pp. 243–260, 1998.

[10] J. Glaunes, A. Qiu, M. I. Miller, and L. Younes, "Large deformation diffeomorphic metric curve mapping," *Int. J. Comput. Vis.*, vol. 80, no. 3, pp. 317–336, 2008.
[11] S. Zhou et al., "Fast and accurate electron microscopy image registration with 3D convolution," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervention*, 2019, pp. 478–486.
[12] B. Hu, S. Zhou, Z. Xiong, and F. Wu, "Self-recursive contextual network for unsupervised 3D medical image registration," in *Proc. Int. Workshop Mach. Learn. Med. Imag.*, 2020, pp. 60–69.
[13] M.-M. Rohé, M. Datar, T. Heimann, M. Sermesant, and X. Pennec, "SVF-Net: Learning deformable image registration using shape matching," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2017, pp. 266–274.
[14] X. Yang, R. Kwitt, M. Styner, and M. Niethammer, "Quicksilver: Fast predictive image registration-A deep learning approach," *NeuroImage*, vol. 158, pp. 378–396, 2017.
[15] S. Zhao, Y. Dong, E. I. Chang, and Y. Xu, "Recursive cascaded networks for unsupervised medical image registration," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 10600–10610.
[16] A. V. Dalca, G. Balakrishnan, J. Guttag, and M. R. Sabuncu, "Unsupervised learning for fast probabilistic diffeomorphic registration," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2018, pp. 729–738.
[17] X. Hu, M. Kang, W. Huang, M. R. Scott, R. Wiest, and M. Reyes, "Dual-stream pyramid registration network," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2019, pp. 382–390.
[18] Z. Shen, X. Han, Z. Xu, and M. Niethammer, "Networks for joint affine and non-parametric image registration," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4224–4233.
[19] B. Hu, S. Zhou, Z. Xiong, and F. Wu, "Cross-resolution distillation for efficient 3D medical image registration," *IEEE Trans. Circuits Syst. Video Technol.*, to be published, doi: 10.1109/TCSVT.2022.3178178.
[20] B. D. de Vos, F. F. Berendsen, M. A. Viergever, H. Sokooti, M. Staring, and I. Išgum, "A deep learning framework for unsupervised affine and deformable image registration," *Med. Image Anal.*, vol. 52, pp. 128–143, 2019.
[21] X. Cao, J. Fan, P. Dong, S. Ahmad, P.-T. Yap, and D. Shen, "Image registration using machine and deep learning," in *Handbook of Medical Image Computing and Computer Assisted Intervention*. Amsterdam, The Netherlands: Elsevier, 2020, pp. 319–342.
[22] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2015, pp. 234–241.
[23] S. Zhao, T. Lau, J. Luo, I. Eric, C. Chang, and Y. Xu, "Unsupervised 3D end-to-end medical image registration with volume tweening network," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 5, pp. 1394–1404, May 2020.
[24] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial transformer networks," *Adv. Neural Inf. Process. Syst.*, vol. 28, 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/33ceb07bf4eeb3da587e268d663aba1a-Paper.pdf
[25] T. C. Mok and A. C. Chung, "Large deformation diffeomorphic image registration with Laplacian pyramid networks," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2020, pp. 211–221.
[26] Y. Shu, H. Wang, B. Xiao, X. Bi, and W. Li, "Medical image registration based on uncoupled learning and accumulative enhancement," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2021, pp. 3–13.
[27] B. B. Avants, C. L. Epstein, M. Grossman, and J. C. Gee, "Symmetric diffeomorphic image registration with cross-correlation: Evaluating automated labeling of elderly and neurodegenerative brain," *Med. Image Anal.*, vol. 12, no. 1, pp. 26–41, 2008.
[28] G. Balakrishnan, A. Zhao, M. R. Sabuncu, J. Guttag, and A. V. Dalca, "An unsupervised learning model for deformable medical image registration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 9252–9260.
[29] Y. He et al., "Few-shot learning for deformable medical image registration with perception-correspondence decoupling and reverse teaching," *IEEE J. Biomed. Health Informat.*, vol. 26, no. 3, pp. 1177–1187, Mar. 2022.
[30] B. Kim, D. H. Kim, S. H. Park, J. Kim, J.-G. Lee, and J. C. Ye, "CycleMorph: Cycle consistent unsupervised deformable image registration," *Med. Image Anal.*, vol. 71, 2021, Art. no. 102036.
[31] M. Q. Tran, T. Do, H. Tran, E. Tjiputra, Q. D. Tran, and A. Nguyen, "Light-weight deformable registration using adversarial learning with distilling knowledge," *IEEE Trans. Med. Imag.*, vol. 41, no. 6, pp. 1443–1453, Jun. 2022.
[32] T. Rohlfing, "Image similarity and tissue overlaps as surrogates for image registration accuracy: Widely used but unreliable," *IEEE Trans. Med. Imag.*, vol. 31, no. 2, pp. 153–163, Feb. 2012.
[33] H. Yu, X. Chen, H. Shi, T. Chen, T. S. Huang, and S. Sun, "Motion pyramid networks for accurate and efficient cardiac motion estimation," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2020, pp. 436–446.
[34] M. Kang, X. Hu, W. Huang, M. R. Scott, and M. Reyes, "Dual-stream pyramid registration network," *Med. Image Anal.*, vol. 78, 2022, Art. no. 102379.
[35] J. Lv et al., "Joint progressive and coarse-to-fine registration of brain MRI via deformation field integration and non-rigid feature fusion," *IEEE Trans. Med. Imag.*, to be published, doi: 10.1109/TMI.2022.3170879.
[36] J. Ashburner, "A fast diffeomorphic image registration algorithm," *Neuroimage*, vol. 38, no. 1, pp. 95–113, 2007.
[37] V. Arsigny, O. Commowick, X. Pennec, and N. Ayache, "A log-euclidean framework for statistics on diffeomorphisms," in *Proc. Int. Conf. Med. Image Comput. Comput.- Assist. Intervention*, 2006, pp. 924–931.
[38] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open access series of imaging studies (oasis): Crosssectional mri data in young, middle aged, nondemented, and demented older adults," *J. Cogn. Neurosci.*, vol. 19, no. 9, pp. 1498–1507, 2007.
[39] Dalca, "Oasis." [Online]. Available: https://github.com/adalca/medicaldatasets/blob/master/neurite-oasis.md
[40] A. Hoopes, M. Hoffmann, B. Fischl, J. Guttag, and A. V. Dalca, "Hypermorph: Amortized hyperparameter learning for image registration," in *Proc. Int. Conf. Inf. Process. Med. Imag.*, Springer, 2021, pp. 3–17.
[41] A. Klein and J. Tourville, "101 labeled brain images and a consistent human cortical labeling protocol," *Frontiers Neurosci.*, vol. 6, p. 171, 2012.
[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[43] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklindge, "Comparing images using the hausdorff distance," *IEEE Trans. Pattern Anal. Mach Intell.*, vol. 15, no. 9, pp. 850–863, 1993.
[44] S. Klein, M. Staring, K. Murphy, M. A. Viergever, and J. P. Pluim, "Elastix: A toolbox for intensity-based medical image registration," *IEEE Trans. Med. Imag.*, vol. 29, no. 1, pp. 196–205, 2009.
[45] B. B. Avants, N. Tustison, and G. Song, "Advanced normalization tools (ants)," *Insight J.*, vol. 2, no. 365, pp. 1–35, 2009.
[46] H. Robbins and S. Monro, "A stochastic approximation method," *Ann. Math. Statist.*, pp. 400–407, 1951.
[47] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited On*, vol. 14, no. 8, p. 2, 2012.