Two and three dimensional image registration based on B-spline composition and level sets

Chiu Ling Chan[1], Cosmin Anitescu[1], Yongjie Zhang[2] and Timon Rabczuk[1,3,4,5]

[1] Institute of Structural Mechanics, Bauhaus Universität Weimar, Germany

[2] Department of Mechanical Engineering, Carnegie Mellon University, USA

[3] Faculty of Civil Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam.

[4] Division of Computational Mechanics, Ton Duc Thang University, Ho Chi Minh City, Vietnam.

[5] School of Civil, Environmental and Architectural Engineering, Korea University, South Korea

Preprint of DOI: 10.4208/cicp.OA-2016-0042

Abstract

A method for non-rigid image registration that is suitable for large deformations is presented. Conventional registration methods embed the image in a B-spline object, and the image is evolved by deforming the B-spline object. In this work, we represent the image using B-spline and deform the image using a composition approach. We also derive a computationally efficient algorithm for calculating the B-spline coefficients and gradients of the image by adopting ideas from signal processing using image filters. We demonstrate the application of our method on several different types of 2D and 3D images and compare it with existing methods.

Keywords— Image registration, B-spline, level set, free-form deformation

Program summary

Program title: CDMFFD image registration.
Nature of problem: Find a transformation (register) between a source (moving) image and a target (fixed) image.
Software licence: BSD 3-clause licence
CiCP scientific software URL:
Programming language(s): Matlab and C.
Computer platform: x86-64.
Operating system: Linux, Windows and Mac OS X.
Compilers: Refer to the following links for supported compilers that can interface with MATLAB:

- http://www.mathworks.com/support/compilers
- http://www.mathworks.com/support/sysreq/previous_releases.html

RAM: 8 GB recommended (more RAM may be needed for large 3D images).
Running time: Image size and computer resource dependent.

1 Introduction

Image registration is the process of finding the optimal spatial transformation that maps one image to another. Image registration can be used for determining the correspondence between images taken at different times, different perspectives, or from different modalities. It has wide applications in medical imaging, remote sensing, computer vision and many other emerging fields.

In general, the registration process acquires information from the fixed and moving images for determination of the transformation map. Features and the intensity of the image are the common information used for registration. The feature-based registration requires detection of salient and distinctive objects (closed boundary regions, line intersections, edges, corners, etc.) followed by establishing the mapping of corresponding features in the images. A shortcoming of feature-based methods is that they require the detection of features (which may need to be done manually), and the accuracy of registration depends on the accuracy of feature detection. The intensity-based method employs the pixel intensity levels as the information for determining the spatial correspondence between the images. The complexity of detecting the feature can thus be avoided. This method is more flexible than the feature-based registration because all the information available from the raw data is taken into consideration throughout the registration process.

Besides the type of information adopted, image registration can also be classified into rigid and non-rigid registration. Rigid registration assumes images can be aligned with one another through rotation or translation while localized stretching of an image is necessary for non-rigid registration. Non-rigid registration is useful for instances of patient motion and atlas registration applications. There are various methods for non-rigid registration [6,11,12,14,18,21]. In particular, Thirion [18] approached the registration problem as a diffusion model. This was proven to be an approximation of a second order gradient descent based on the sum of squared intensity differences (SSD) [14]. Vemuri [21] presented a deformation method by considering an evolving model for the image. Cachier [6] introduced the iconic feature based (IFB) algorithm, where intensity, similarity, and geometric distance were taken into consideration. In [11, 12], the image was deformed by an L^2-gradient flow which is derived through minimizing a predefined energy function.

Free form deformation (FFD) is commonly used to deform the image in non-rigid registration [11–13, 15–17, 19]. In FFD, the image is transformed by deforming the B-spline object in which it is embedded, and the B-spline control points are used to describe the motion of the image. A gradient based FFD for non-rigid local deformation was discussed in [16] and a further improvement by using a B-spline fitting approach was proposed in [19]. The advantage of B-spline based FFD methods is that the multiresolution scheme can be applied using B-spline subdivision. Instead of the conventional subdivision, hierarchical B-splines were used in FFD to reduce the number of control points. In the standard FFD method, the image is embedded in a given grid which is defined by B-splines. The gradient is computed directly from the pixel-based image on the location of the control point of this grid. The gradient computed is used to update the B-spline grid and the image is updated by interpolating the intensity value along the deformed grid. In this paper, we propose an alternative FFD method, where we compute the gradient from a smooth B-spline level set representation of the image. In addition, instead of interpolation, we update the image with composition, which is be discussed in more detail in section 4.

A challenge in non-rigid registration is to obtain a smooth deformation field that is invertible. Such deformation is known as diffeomorphic registration. Two common approaches to achieve

diffeomorphic registration include the elastic [9,10] and fluid models [4,5,8]. The elastic deformation is based on the Green St. Venant strain tensor. It is however, only valid for small deformations. The fluid model is based on the rate of deformation tensor, which can handle large deformations. Vercauteren [22] presented a diffeomorphic registration method by applying ideas from the Lie Group theory. Another approach [15] is to introduce constraints on the control points such that their displacement is sufficiently small which results in a diffeomorphic B-spline transformation.

Despite the availability of well-developed registration methods, in this paper we present an alternative image deformation model compared to FFD in which we represent the image using a B-spline model. Different from the standard FFD, where an optimization is performed on the locations of the control points, the displacements are explicitly computed at each red registration step, in a manner similar to the optical flow or Thirion's Demons method. The gradients of the intensity value can be evaluated directly in terms of the derivatives of the basis functions. The explicit displacement field is projected on the space of B-spline transformations using composed updates which provides a smoothing effect to the deformation mapping. We present the use of image filtering techniques for the efficient computation of gradients and coefficients. Since filtering is commonly used in image processing, there are efficient algorithms available which help minimize the computational cost. Since both the image data and the deformations are represented in terms of B-splines, the proposed approach is in some sense "isoparametric" and has the potential for tighter integration with Computer-Aided Design and analysis software.

The paper is organized as follows: Section 2 gives a brief introduction to the registration framework. The B-spline representation is presented in Section 3. In Section 4, we discuss our method for non-rigid diffeomorphic registration, while the multiresolution registration is discussed in Section 5. The proposed method is illustrated by numerical examples in Section 6. We have also made the Matlab implementation of the proposed method available online at **https://github.com/stellaccl/cdmffd-image-registration**. We refer to the included README file for a description of the installation procedures.

2 Registration framework

Given a fixed image I_1 and a moving image I_0, image registration is an optimization problem that aims to find the spatial transformation $\mathbf{T}$ such that

$$I_0 \circ \mathbf{T} \approx I_1. \tag{1}$$

A similarity criterion is used to measure the differences between the two input images. There are several similarity measures - e.g. direct correlation, mean square error and sum of square differences (SSD). In this paper, we will adopt the SSD which is more computationally straightforward. The SSD is defined as

$$SSD(I_1, I_0) = \sum_{p \epsilon \Omega} |I_1(p) - I_0(p)|^2, \tag{2}$$

where Ω is the region domain of the image, and the sum is taken over all pixels (or voxels) p. A simple optimization of (2) will lead to unstable and non-smooth solutions. This problem can be avoided by adding the regularization term $\mathrm{Reg}(\mathbf{T})$ to the global energy function as:

$$E(\mathbf{T}) = SSD(I_1, I_0) + \sigma_T \mathrm{Reg}(\mathbf{T}) \tag{3}$$

where σ_T is a parameter to control the amount of regularization needed. The combination of SSD and regularization provides a well-posed optimization framework. However, it is quite challenging

to optimize (3) when $\mathbf{T}$ depends on a large number of parameters. Cachier [6] proposed a solution to this problem by adding a hidden variable and by defining the global energy function in terms of two transformations ($\mathbf{C}$ and $\mathbf{T}$) as follows:

$$E(\mathbf{C},\mathbf{T}) = SSD(I_1, I_0 \circ \mathbf{C}) + \sigma_x \text{dist}(\mathbf{T},\mathbf{C})^2 + \sigma_T \text{Reg}(\mathbf{T}). \tag{4}$$

Here $\text{dist}(\mathbf{C},\mathbf{T}) = ||\mathbf{T}-\mathbf{C}||$ and σ_x accounts for the spatial uncertainty between the transformation $\mathbf{C}$ and $\mathbf{T}$. The optimization is initialized by setting $\mathbf{T}_0$ equal to the identity transformation (Id) at iteration 0 and performing the following steps at each iteration $n \geq 1$:

- For a given $\mathbf{T}_{n-1}$, find $\mathbf{C}_n$ that minimizes

$$E_1(\mathbf{C}_n;\mathbf{T}_{n-1}) = SSD(I_1, I_0 \circ \mathbf{C}_n) + \sigma_x \text{dist}(\mathbf{T}_{n-1},\mathbf{C}_n)^2.$$

- For the $\mathbf{C}_n$ obtained from the previous step, find $\mathbf{T}_n$ that minimizes

$$E_2(\mathbf{T}_n;\mathbf{C}_n) = \sigma_x \text{dist}(\mathbf{T}_n,\mathbf{C}_n)^2 + \sigma_T \text{Reg}(\mathbf{T}_n).$$

  If $\text{Reg}(\mathbf{T}_n) = \| \nabla \mathbf{T}_n \|^2$ this minimization can be efficiently computed using a convolution of Gaussian kernel K_G with $\mathbf{C}_n$, i.e. $\mathbf{T}_n \leftarrow K_G * \mathbf{C}_n$. The symbol $*$ denotes the convolution operator.

Thirion [18] presented an efficient method to compute a force that is equivalent to the computation of an update to $\mathbf{T}_n$, as discussed in the first step above. The main idea is to deform the image by using a diffusion model and defining a force acting on each pixel (so-called the "demon" force analogy to Maxwell's demon). The force is modified to include a normalization that is necessary to avoid instabilities due to small gradient values. Thirion used updates of the form:

$$\mathbf{V}_{n+1} = \frac{(I_1 - I_0 \circ \mathbf{T}_n)\nabla I_1}{\|\nabla I_1\|^2 + (I_1 - I_0 \circ \mathbf{T})^2}, \quad \mathbf{T}_0 = \text{Id}, \quad \mathbf{T}_n = \text{Id} + \mathbf{V}_n. \tag{5}$$

Here $\mathbf{V}_n$ is a vector field representing the displacement of each pixel at step n. Vercauteren [22] derived a more general displacement field as:

$$\mathbf{V}_{n+1} = \frac{(I_1 - I_0 \circ \mathbf{T}_n)\mathbf{J}^p}{\| \mathbf{J}^p \|^2 + \sigma_x^{-2}(I_1 - I_0 \circ \mathbf{T}_n)}, \tag{6}$$

where $\mathbf{J}^p$ can also be defined as either the gradient of the moving image or the average of the gradient of the fixed and moving image $\frac{\nabla I_1 + \nabla(I_0 \circ \mathbf{T})}{2}$. σ_x is a parameter used to control the step size of the displacement field. It can be shown [22] that the maximum displacement is bounded by:

$$\| \mathbf{V}_{n+1} \| \leqslant \frac{\sigma_x}{2}.$$

Vemuri [21] proposed to deform the image using the level set method. The idea is to let the image evolve in the direction normal to its contour lines. The vector field is updated using the following governing equation:

$$\mathbf{V}_{n+1} = (I_1 - I_0 \circ (\text{Id} + \mathbf{V}_n)) \frac{\nabla(G * (I_0 \circ (\text{Id} + \mathbf{V}_n)))}{||\nabla(G * (I_0 \circ (\text{Id} + \mathbf{V}_n)))|| + \alpha}, \tag{7}$$

where G represents the Gaussian Kernel and α is a small positive constant that acts as a stabilizer. Though Thirion [18], Vercauteren [22] and Vemuri [21] used different approaches, the vector field updates are computed similarly in all these methods. In this paper, we will adopt an approach similar to that in [22] and use the parameter σ_x to control the step size.

There are two ways to update the transformation $\mathbf{T}$ from a given displacements field $\mathbf{V}$ [22]:

1. Additive update
$$\mathbf{C}_n = \mathbf{T}_{n-1} + \mathbf{V}_n = \mathbf{V}_1 + \mathbf{V}_2 + \cdots + \mathbf{V}_n.$$

2. Composite update
$$\mathbf{C}_n = \mathbf{T}_{n-1} \circ (\mathrm{Id} + \mathbf{V}_n) = \mathbf{V}_1 \circ \mathbf{V}_2 \circ \cdots \circ \mathbf{V}_n\,.$$

Additive updates are easy to implement. However, they are less efficient for large deformations and have no geometric meaning. For example if $\mathbf{V}_n$ represents a rotation by angle α, then a rotation by angle 2α is naturally represented by $\mathbf{V}_n \circ \mathbf{V}_n = \mathbf{V}_n^2$, which is not the same as $\mathbf{V}_n + \mathbf{V}_n$, in particular for large α. According to Ashburner [3], the additive method is only suitable for small deformations. The inverse transformation of the additive method obtained through subtraction is just an approximation. Therefore, it is hard to enforce a one-to-one mapping that is invertible and can correctly represent large deformations. Composite updates provide a natural way of working with diffeomorphic transformations (since the composition of two diffeomorphisms is also diffeomorphism). However, composite updates are more difficult to implement. They normally require the interpolation of the image intensity and the gradients "in-between" the pixels, which can introduce errors in the registration algorithm. In this paper, we perform the update of transformation by composing the B-spline transformation from the previous step with a linear update $(\mathrm{Id} + \mathbf{V})$, resulting in another B-spline transformation of the same polynomial degree. More details of this procedure will be discussed in section 4.

3 B-spline representation

Suppose the image is contained in a domain $\Omega = \otimes_{d=1}^{n_d}[0, L_d] \in \mathbb{R}^{n_d}$ which is partitioned by a set of pixels or voxels Ω^e_{voxel}, $e \in \{1, 2, \ldots, m_{voxel}\}$. n_d represents the physical dimension, in the following discussion we will assume $n_d = 3$. The gray scale function $g : \Omega \to \mathbb{R}$ is defined as

$$g(X) = \begin{cases} \omega_1 & : X \in \Omega^1_{voxel} \\ \omega_2 & : X \in \Omega^2_{voxel} \\ \vdots & \\ \omega_{m_{voxel}} & : X \in \Omega^{m_{voxel}}_{voxel} \end{cases}$$

with $X = (x, y, z)$ being the coordinates of the voxel centers and $\omega_\imath$ $(\imath = 1, 2, \ldots, m_{voxel})$ is the gray scale value in the voxels $\{\Omega^e_{voxel}\}_{e=1}^{m_{voxel}}$. This discrete representation is non-smooth and would significantly affect the calculation of gradient. A smooth representation of the image is, therefore, necessary for more accurate gradient evaluations. We propose to compute the smooth approximation of the image based on a discrete cubic B-spline kernel h_a. The elements in h_a are computed using the cardinal basis function $\beta^3(x)$ defined as follows:

$$\beta^3(x) = \begin{cases} b_0(x) = \frac{(2+x)^3}{6} & : -2 \leqslant x < -1 \\ b_1(x) = \frac{2}{3} - x^2 - \frac{x^3}{2} & : -1 \leqslant x < 0 \\ b_2(x) = \frac{2}{3} - x^2 + \frac{x^3}{2} & : 0 \leqslant x < 1 \\ b_3(x) = \frac{(2-x)^3}{6} & : 1 \leqslant x < 2 \\ 0 & : \text{otherwise} \end{cases} \tag{8}$$

Given the gray values $g(x, y, z)$, we would like to find the coefficients $c_{i,j,k}$ such that the cubic

B-spline function $f(u, v, w)$ interpolates the intensity of the image:

$$f(x, y, z) = \sum_{i=l}^{l+3} \sum_{j=m}^{m+3} \sum_{k=n}^{n+3} c_{i,j,k}\, \beta^3(x-i)\beta^3(y-j)\beta^3(z-k) \tag{9}$$

with

$$l = \lceil x - 2 \rceil, \quad m = \lceil y - 2 \rceil, \quad n = \lceil z - 2 \rceil.$$

Solving for the coefficients $c_{i,j,k}$ can be performed using a matrix framework, e.g. by an L^2 projection or least squares fitting. A more efficient approach is to use digital filtering techniques. Using the fact that the B-splines are translation invariant, g can be evaluated at the voxel center (x, y, z) by using convolution as follows:

$$g(x, y, z) = (h_a * c_{i,j,k}) \tag{10}$$

where h_a is the discrete B-spline kernel. The convolution operation, in this case, can be viewed as sliding a kernel h_a over the coefficients array $c_{i,j,k}$, this process is known as "filtering with convolution". In the rest of the paper, we refer to it as filtering. $c_{i,j,k}$ can be solved by an inverse filtering:

$$c_{i,j,k} = (h_a)^{-1} * g(x, y, z). \tag{11}$$

Unser [20] presented an efficient computation method for $c_{i,j,k}$ by using casual and anti-casual recursive filter. The resulting B-spline model is an exact interpolation of the data. In our study, however, we prefer a smoother approximation to avoid oscillation caused by noise or Gibbs phenomenon. We implement the approach in [24] and compute the coefficients as follows:

$$\begin{aligned} c_{i,j,k} &= \frac{\int_\Omega \hat{\beta}^3_{i,j,k}(x, y, z) g(x, y, z)\, dx\, dy\, dz}{\int_\Omega \hat{\beta}^3_{i,j,k}(x, y, z)\, dx\, dy\, dz} \\ &= \frac{1}{A} \int_\Omega \hat{\beta}^3_{i,j,k}(x, y, z) g(x, y, z)\, dx\, dy\, dz \end{aligned} \tag{12}$$

where $\hat{\beta}^3_{i,j,k}(x, y, z) = \beta^3(x-i)\beta^3(y-j)\beta^3(z-k)$, and $A = \int_\Omega \hat{\beta}^3_{i,j,k}(x, y, z)\, dx\, dy\, dz$. Since $g(x, y, z)$ is a piecewise constant function and $\hat{\beta}^3_{i,j,k}(x, y, z)$ is a piecewise polynomial, we obtain

$$\begin{aligned} &\int_\Omega \hat{\beta}^3_{i,j,k}(x, y, z) g(x, y, z) dx\, dy\, dz \\ &= \int_\Omega \sum_{i=l}^{l+3} \sum_{j=m}^{m+3} \sum_{k=n}^{n+3} \beta^3(x-i)\beta^3(y-j)\beta^3(z-k) g(x, y, z)\, dx\, dy\, dz \\ &= h_a * M \end{aligned} \tag{13}$$

where h_a is a kernel given by:

$$h_a = \sum_{\hat{i}=0}^{3} \sum_{\hat{j}=0}^{3} \sum_{\hat{k}=0}^{3} a_{\hat{i}} a_{\hat{j}} a_{\hat{k}}\,,$$

and $a_{\hat{i}} = \int_{p_{\hat{i}}}^{p_{\hat{i}+1}} b_{\hat{i}}(x)\, dx$, $p_{\hat{i}} = \hat{i} - 2$. Moreover, matrix M is composed of gray scale values $\omega_\imath$ of the corresponding integration domain. Thus, we can compute the coefficients efficiently by:

$$c_{i,j,k} = \frac{1}{A}(h_a * M). \tag{14}$$

Note that h_a is a $4 \times 4 \times 4$ matrix whose entries only depend on the cardinal B-spline β^3 and can be precomputed. In addition, h_a is a separable kernel which has a computational advantage, since applying it to a 3D image can be done by 1D filtering in each direction. Quantitatively, if $h_a \in \mathbb{R}^{p \times q \times r}$is non-separable, then filtering requires $\mathcal{O}(p \cdot q \cdot r)$ multiplication and additions for each voxel. For a separable kernel, the computation cost decreases to $\mathcal{O}(p + q + r)$. Efficient algorithms exist for performing these operations, including those that exploit parallelism on multi-core systems or GPUs.

With the coefficients $c_{i,j,k}$ obtained using the approach in [24], the following properties hold:

1. Conservation of average gray scale intensity

$$\frac{1}{A}\int_{\Omega} f(X)dX = \frac{1}{A}\int_{\Omega} g(X)dX = \frac{1}{m_{voxel}}\sum_{e=1}^{m_{voxel}} c_e. \tag{15}$$

2. Local Boundedness

$$\min_{X \in \Omega^e_{supp}} (g(X)) \leq f(X) \leq \max_{X \in \Omega^e_{supp}} (g(X)) \quad \forall X \in \Omega^e_{voxel}. \tag{16}$$

where $\Omega^e_{supp} = \bigcup_{e \in \mathcal{I}_e} \Omega^e_{voxel}$, and $\mathcal{I}_e = \{e : \hat{\beta}^3(X)\big|_{\Omega^e_{voxel}} \neq 0\}$

3. Approximation to a Gaussian Kernel
Substituting (12) into (9) we obtain

$$f(X) = \int_{\Omega} \Big[\sum_{i,j,k} \frac{\hat{\beta}^3_{i,j,k}(X)\hat{\beta}^3_{i,j,k}(Y)}{A}\Big] g(Y)dY. \tag{17}$$

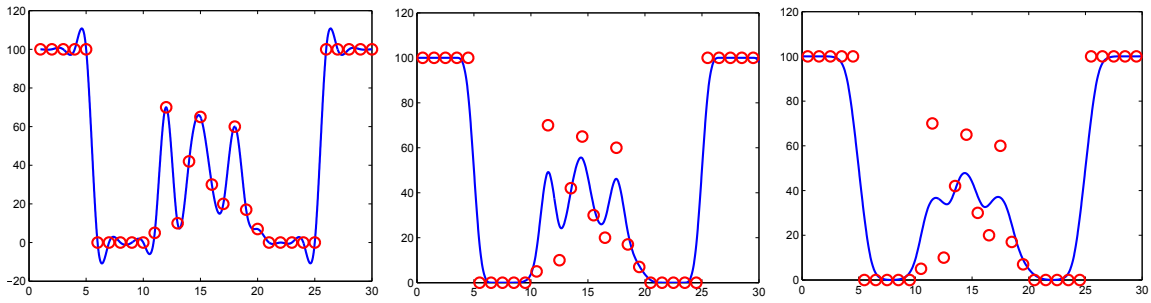
Thus $f(X)$ is the integral transform of intensity value $g(X)$ with the kernel

$$k(X,Y) = \sum_{i,j,k} \frac{\hat{\beta}^3_{i,j,k}(X)\hat{\beta}^3_{i,j,k}(Y)}{A}. \tag{18}$$

This kernel is an approximation to the Gaussian filter.

An alternative way to compute the interpolation coefficients, commonly used in FFD, is to set the coefficient $c_{i,j,k}$ to the values of the input data [19]. This method is very easy to implement, but it does not take into account the overlap between the basis functions. The resulting spline is less smooth for noisy input than the convolution approach.

We illustrate with a 1D example the difference between computing coefficient using the exact interpolation method in [20], the method used in [19] and the method using (14). In Figure 1, the red circles represent the data values. Figure 1a and Figure 1b show interpolation using the exact interpolation method in [20] and interpolation used in FFD respectively. For both methods, oscillations occur at the places where there is a sharp change of data value. Figure 1c shows interpolation using the convolution strategy discussed in [24], whereby a smoother approximation is obtained. Such smooth interpolation is desirable in the context of image registration, because it provides a filtering effect that smoothes irregular intensity caused by noise. Thus, it gives a representation of the image that is suitable for computing the gradient of the intensity. Another advantage of representing the image using a B-spline model is that it is second-order continuously differentiable. We will discuss in the following section fast computation of gradients from the coefficients $c_{i,j,k}$ by filtering.



(a) (b) (c)

Figure 1: (a) Interpolation using the exact interpolation method, (b) interpolation in [19], and (c) interpolation using (14).

4 Non-rigid registration

4.1 The governing equation

In this study, we deform the image using the following displacement $\mathbf{V}(X)$ at each point X:

$$\mathbf{V}(X) = \frac{(I_1(X) - f(X))\nabla f(X)}{\| \nabla f(X) \|^2 + \gamma \cdot (I_1(X) - f(X))^2}. \tag{19}$$

An important difference in this equation compared to (5), (6) and (7) is that we use the B-spline representation to evaluate the gradient values and that $f(X)$ is a B-spline representation of $I_0 \circ \mathbf{T}$. This can be done by finding the partial derivative of the basis using a filtering method as discussed in the previous section. This vector update contains the registration parameter γ corresponding to $\frac{1}{\sigma_x^2(x)}$ in (6) which allows more control over the step size. To compute the gradient $\nabla f(X) = \begin{bmatrix} \frac{\partial f(X)}{\partial x} \\ \frac{\partial f(X)}{\partial y} \\ \frac{\partial f(X)}{\partial z} \end{bmatrix}$, the partial derivative of $f(X)$ with respect to x is given using (9) by:

$$\frac{\partial f(X)}{\partial x} = \frac{\partial f(x,y,z)}{\partial x} = \sum_{i=l}^{l+3} \sum_{j=m}^{m+3} \sum_{k=n}^{n+3} c_{i,j,k} \frac{\partial \beta^3(x-i)}{\partial x} \beta^3(y-j)\beta^3(z-k). \tag{20}$$

Using the fact that we need to evaluate the gradients only at the voxel centers, it suffices to precompute the values of the cardinal B-spline $\hat{\beta}^3$ and its gradients at the knot midpoints -1.5, -0.5, 0.5 and 1.5. In particular, we define the discrete kernel h_x by:

$$h_x(\hat{i},\hat{j},\hat{k}) = \frac{\partial \beta^3(x_{\hat{i}})}{\partial x} \times \beta^3(x_{\hat{j}}) \times \beta^3(x_{\hat{k}}) \quad , \quad \hat{i},\hat{j},\hat{k} = 0,1,2,3\,, \tag{21}$$

where $x_0 = -1.5$, $x_1 = -0.5$, $x_2 = 0.5$, $x_3 = 1.5$. Now (20) can be written in the form of the convolution

$$\frac{\partial f(X)}{\partial x} = (h_x * c)(X). \tag{22}$$

The corresponding kernel h_y and h_z of variable y and z can be computed using a similar procedure. By using these kernels, $\nabla f(X)$ can be computed efficiently by the filtering technique:

$$\nabla f(X) = \begin{bmatrix} (h_x * c)(X) \\ (h_y * c)(X) \\ (h_z * c)(X) \end{bmatrix}. \tag{23}$$

In this study, we compute the composition of transformation using B-splines. Let $\mathbf{T}_{n-1}$ be the previous transformation and $\mathbf{V}_n$ be the updated vector field, we compose $\mathbf{V}_n$ with $\mathbf{T}_{n-1}$ as:

$$\begin{aligned}\mathbf{T}_n(X) &= \mathbf{T}_{n-1} \circ (\mathrm{Id} + \mathbf{V}_n)(X) \\ &= \sum_{i=l}^{l+3} \sum_{j=m}^{m+3} \sum_{k=n}^{n+3} \mathbf{C}_{i,j,k}^{(n-1)} \hat{\beta}^3_{i,j,k}((\mathrm{Id} + \mathbf{V}_n)(X)) \\ &= \sum_{i=l}^{l+3} \sum_{j=m}^{m+3} \sum_{k=n}^{n+3} \mathbf{C}_{i,j,k}^{(n)} \hat{\beta}^3_{i,j,k}(X)\end{aligned} \tag{24}$$

where $\mathrm{Id}(X) = X$ is the identity transformation. The coefficients $\mathbf{C}_{i,j,k}^{(n)} = [c_{i,j,k}^{x,(n)}, c_{i,j,k}^{y,(n)}, c_{i,j,k}^{z,(n)}]$ are obtained by convolution of the x, y and z components of the previous transformation $\mathbf{T}_{n-1}^x(X)$, $\mathbf{T}_{n-1}^y(X)$ and $\mathbf{T}_{n-1}^z(X)$ composed with the displacement $(\mathrm{Id}+\mathbf{V}_n)(X)$. However, note that here the B-spline $\hat{\beta}^3$ is evaluated at points that are not equally spaced. This requires the evaluation of cubic polynomials at arbitrary points. However, the computation of $\mathbf{C}_{i,j,k}^{(n)}$ can be performed efficiently using filtering with the B-spline kernel h_a as follows:

$$\begin{aligned}c_{i,j,k}^{x,(n)} &= \tfrac{1}{A}(h_a * \mathbf{T}_n^x(X)), \\ c_{i,j,k}^{y,(n)} &= \tfrac{1}{A}(h_a * \mathbf{T}_n^y(X)), \\ c_{i,j,k}^{z,(n)} &= \tfrac{1}{A}(h_a * \mathbf{T}_n^z(X)).\end{aligned} \tag{25}$$

We note that the repeated convolution using (25) only affects the transformation mapping and can be used to provide a smoothing effect. The source image is converted to a B-spline level-set only once via convolution, as described in more detail in Section 4.4, so there is no degradation of the image quality during the registration.

4.2 Diffeomorphisms

It is important for many applications that the deformation field is smooth and injective. Finding such deformations is known as diffeomorphic registration. Referring to Rueckert [15], it can be achieved by constraining the displacement to be a fraction of the distance between the voxel centers. Let $\Delta c_{i,j,k} = c_{i,j,k}^{(n)} - c_{i,j,k}^{(n-1)}$ be the displacement of control point $c_{i,j,k}^{(n-1)}$. Moreover the deformation is locally injective if $\max|\Delta c_{i,j,k}^x| < \frac{1}{K}$, $\max|\Delta c_{i,j,k}^y| < \frac{1}{K}$ and $\max|\Delta c_{i,j,k}^z| < \frac{1}{K}$, where $K \approx 2.48$ when using uniform cubic B-spline functions [7]. Using the fact [24] that the displacement of the control points is bounded by the displacement of the voxels, and equations (6) and (19) we write:

$$\max\|\Delta \mathbf{C}_{i,j,k}\| \leqslant \|\mathbf{V}(X)\| \leqslant \frac{\sigma_x}{2} = \frac{1}{2\sqrt{\gamma}}.$$

Thus, in order to obtain diffeomorphic registration, we can set γ such that

$$\frac{1}{2\sqrt{\gamma}} \leqslant \frac{1}{k}.$$

Since $K \approx 2.48$, we have $\gamma \geqslant 1.5376$. Therefore diffeomorphic transformation can be obtained by choosing such γ in (19). Since the composition of diffeomorphic transformations is diffeomorphic, this property is preserved by the overall registration.

4.3 Regularization

The transformations can be regularized by relying on physical models (i.e. elastic model, fluid model, etc.). Note that the B-spline convolution strategy that we used for computing the coefficient for B-spline model of the image (discussed in section 3) and also vector transformation (discussed in section 4.1) already have a regularization effect. However, if more regularization is necessary, we can apply a Gaussian filter to the vector field. The convolution of a vector field V with Gaussian kernel G to give a smoothed displacement is performed as:

$$\mathbf{V}(x,y,z) * G = \sum_{i=-N}^{N} \sum_{j=-N}^{N} \sum_{k=-N}^{N} \mathbf{V}(x,y,z) G(x-i, y-j, z-k), \quad N = 6 \times \sigma_G, \tag{26}$$

where x,y and z are the point position of vector V, and σ_G is the standard deviation of Gaussian filter. Applying the Gaussian smoothing to the displacement field at each step could be considered as a rough simulation of a fluid model [14].

4.4 Moving Image Update

Interpolation is applied to update the image with the corresponding transformation. A natural way to perform this update, since both the image and the transformation are represented in terms of B-splines, is to use composition. In particular,

$$f(X) = I_0 \circ \mathbf{T}_n(X) = \sum_{i,j,k} c_{i,j,k} \hat{\beta}^3(\mathbf{T}_n(X)). \tag{27}$$

The coefficients $c_{i,j,k}$ are the coefficients of the B-spline level set function of the source image. This composition can be easily converted to a level set function using (9) for use in the next step. To summarize, we use the Algorithm 1 to find the optimal spatial mapping $\mathbf{T}$. This algorithm is implemented in regImage2D.m and regImage3D.m

4.5 Parallel computing and Matlab Implementation

In image registration, we need to find a solution (the spatial transformation) from a given data set (the images). Solving this problem for large data sets requires a lot of memory. Also, it involves computationally intensive tasks. Thus, parallel computing which take advantage of multi-core processors is suitable for image registration. MPI (Message Passing Interface) and OpenMP (Open Multi-Processing) are the two often used communications protocols for parallel computing. MPI is used in distributed memory systems, where computational tasks are distributed to different nodes. The running code on each node does not have access to the memory of the other nodes, therefore, they communicate via a messaging interface. A very simple MPI implementation is provided in Matlab by the "parfor" command which can be used to parallelise a for loop, where each iteration is independent of the others. When this command is issued, Matlab starts worker-threads that perform the computations in each loop and return them to the main thread. We used "parfor" in the following subroutines:

- computing the coefficients $c_{i,j,k}$ in (14), which is implemented in img2coef2D.m and img2coef3D.m. In these functions, we filtered the data using discrete kernel in each spatial dimension by using the function bspline1dfilt.m.

Algorithm 1: Algorithm for image registration

Input : source image I_0, target image I_1

Output: $\begin{bmatrix} c^x_{i,j,k} \\ c^y_{i,j,k} \\ c^z_{i,j,k} \end{bmatrix}$ such that $I_0 \circ \sum_{i,j,k} \begin{bmatrix} c^x_{i,j,k} \\ c^y_{i,j,k} \\ c^z_{i,j,k} \end{bmatrix} \hat{\beta}_{i,j,k}(X) \approx I_1(X)$

1: Compute $c_{i,j,k}$ using (14) such that $f(x, y, z) \approx I_0(x, y, k)$.

2: Let $\mathbf{T}_0 = \mathrm{Id}(X)$, compute $\begin{bmatrix} c^x_{i,j,k} \\ c^y_{i,j,k} \\ c^z_{i,j,k} \end{bmatrix}$ by using (25).

3: For each iteration:

(a) Compute $\nabla f(X)$ using (23).

(b) Compute $\mathbf{V}_n(X)$ using (19).

(c) Regularize if necessary $\mathbf{V}_n = \mathbf{V}_n * G$.

(d) Evaluate $\mathbf{T}_n = \mathbf{T}_{n-1} \circ (\mathrm{Id} + \mathbf{V}_n)$ using (24).

(e) Compute $\begin{bmatrix} c^x_{i,j,k} \\ c^y_{i,j,k} \\ c^z_{i,j,k} \end{bmatrix}$ for the B-Spline representation of $\mathbf{T}_n$ using (25).

(f) compute $f(X) = I_0 \circ \mathbf{T}_n$ using (27).

(g) Compute $c_{i,j,k}$ of that $f(X)$ using (14).

- composing the transformations in (24), which is implemented in BsplineCompose2D.m and BsplineCompose3D.m. The outermost sum is parallelised by using the "parfor" loop.
- composing the B-Spline level set representation of the moving image I_0 with the current transformation in (27). This is implemented in BsplineComposeImage2D.m and BsplineComposeImage3D.m. The outermost sum is parallelised by using the "parfor" loop.

There are also some subroutines in our code which are parallelised automatically by Matlab. These included the use of imfilter in computing the gradient and also the Gaussian filter (if used). We note there is also a GPU-optimized implementation of imfilter provided by GPUArray / imfilter. Besides MPI, the computation can be further enhanced by using the OpenMP protocol. OpenMP targets shared memory systems, and uses a thread based parallelism. It is suitable for the case where the computational task for each thread is small compared to the size of the variables involved. In Matlab, the "parfor" loops will be converted to OpenMP when it is compiled to a mex file via the built-in "coder" toolbox.

5 Multiresolution Registration

The proposed method can be improved when used in combination with a coarse to fine multiresolution scheme. First, the computational load is reduced significantly when the computations are performed on a coarse version of the image. Second, large deformations can be handled fast and efficiently since the transformation field is coarse. Recall that the goal of image registration is, for given I_0 and I_1, to find the transformations $\mathbf{C}$ and $\mathbf{T}$ such that

$$\min E(\mathbf{C}, \mathbf{T})$$

where $E(\mathbf{C}, \mathbf{T})$ is defined in (4).

In the multiresolution framework, we solve a sequence of registration problems successively from the coarser level ($\ell = 1$) to the finest level ($\ell = \textit{number of levels}$). The scaling factor for the image at level (ℓ) is given by $\frac{1}{2^{number\ of\ levels-\ell}}$. By using this scaling factor, we reduce the image size in each direction by half in each level. In the first level, we can obtain I_0^1 and I_1^1 directly by scaling I_0 and I_1 with the scale factor. For $\ell > 1$, we scale the I_0 and I_1 according the scaling factor of the corresponding level, followed by composing the transformation field ($\mathbf{T}^{\ell-1}$) from level ($\ell - 1$). Then the transformation field ($\mathbf{T}^{\ell}$) is obtained by performing Algorithm 1 using I_0^ℓ and I_1^ℓ. We also update the deformation grids in a similar way so that the information is inherited and the complete transformation can be seen clearly in the final deformation grids.

The multiresolution algorithm is implemented in the functions Multiresolution2D.m and Multiresolution3D. Note that it is not always the case that more multiresolution levels result in faster and better registration. The number of levels should depend on the image size and also on the type of the image. If there are a lot of small details in the image, a very coarse level would not be able to capture the deformation very well.

In the following example, we illustrate the multi-resolution registration discussed above using the Lena image. We perform a three level multiresolution registration on it. In the first row, Figures 2a and 2b are the given fixed and moving image with size 256 $\times$ 256 pixels. The results from the coarse to fine levels are shown in the second to the fourth row. The image sizes of the first and second levels are 64 $\times$ 64 pixels, and 128 $\times$ 128 pixels respectively, and the images in the final level have the original size. Note that it is difficult to illustrate the transformation clearly with dense deformation grids, thus, the deformation grids for all levels are plotted using 40 $\times$ 40 uniformly

spaced grid lines. We can observe that the large deformation can be registered well in the coarse level, and the fine detail is progressively recovered at the finer levels. Also, the resulting deformation grids does not show only the deformation of its level, but the up-scaled transformation from the previous levels. The final deformation grids (Figures 2k and 2j) show the deformation information of the entire registration process. Please refer to main2DLena.m for the Matlab implementation of this example.

6 Numerical examples

We examine our image registration method through several examples. We use the similarity ratio, Rs defined as:

$$Rs = 1 - \frac{\|I_1 - I_0 \circ \mathbf{T}\|}{\|I_1 - I_0\|}$$

to evaluate the result. Perfect registration will result in $Rs = 1$, and a low Rs reflects a poor match between the source and target images. Note that the core aspect of registration is to produce the transformation between the moving and fixed images that is physically plausible. The transformation maps from our algorithm are illustrated in the 2D examples.

6.1 Synthetic transformation example

We first evaluate the registration algorithm by using an image deformed with a known synthetic transformation. The photographic image is distorted using a sinusoidal function as shown in Figure 3c. This synthetic transformation is used such that the same deformation pattern is repeated in the image, and we can examine the ability of our proposed method for recovering the sinusoidal transformation. We compare the deformation grids obtained by the exact transformation, the proposed method and a diffeomorphic optical flow implementation [23] using Matlab [2], respectively. Please refer to main2DLenaSynthetic.m for the Matlab code of this example. In Figure 3, both registered images from our method and the optical flow method look very similar to the fixed image. However, the deformation grids from the optical flow method are unable to recover the deformation grids well. On the other hand, there is a very good match between the computed transformation from the proposed method with the exact transformation. This is true even in the parts of the image that have uniform intensity (such as the background).

6.2 C-shaped image

Next we consider the benchmark example of "C-shape" synthetic images, where we seek to deform a circle into a C-shaped region. The diffeomorphic optical flow method with the implementation in [2] cannot correctly capture the large deformation in this synthetic image, therefore we compare our result with the additive method. In Figure 4, the first row shows the moving image (Figure 4a) and the fixed image (Figure 4b). The second row shows the results of the additive update and the results of our method using composition update are shown in the third row. The Rs of the composition method and additive method is 0.9369 and 0.9504 respectively. Though the additive method obtains a higher Rs value, the corresponding deformation grids (Figure 4d and Figure 4e) looks chaotic and is unable to provide useful information for practical use. This deficiency is due to the additive update only involving vector addition; the fact that large deformation should be based on composition is neglected. The composite method updates the displacement field by warping it

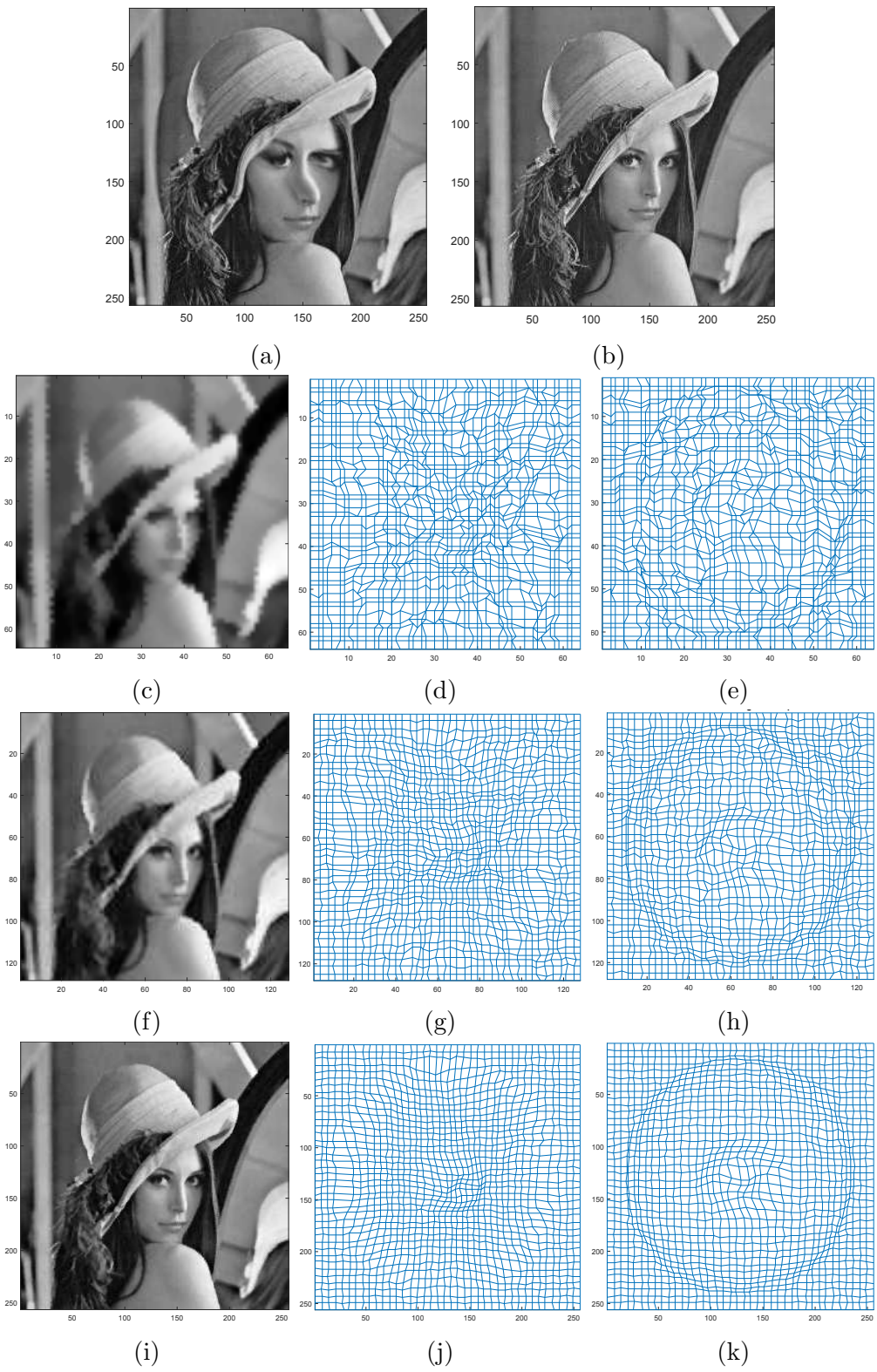


Figure 2: First row: (a) the moving image and (b) the fixed image. Second row: result from the first level; (c) deformed image, (d) forward deformation grids, and (e) inverse deformation grids. Third row: result from the second level; (f) deformed image, (g) forward deformation grids, and (h) inverse deformation grids. Fourth row: result from the third level; (i) deformed image, (j) forward deformation grids, and (k) inverse deformation grids.

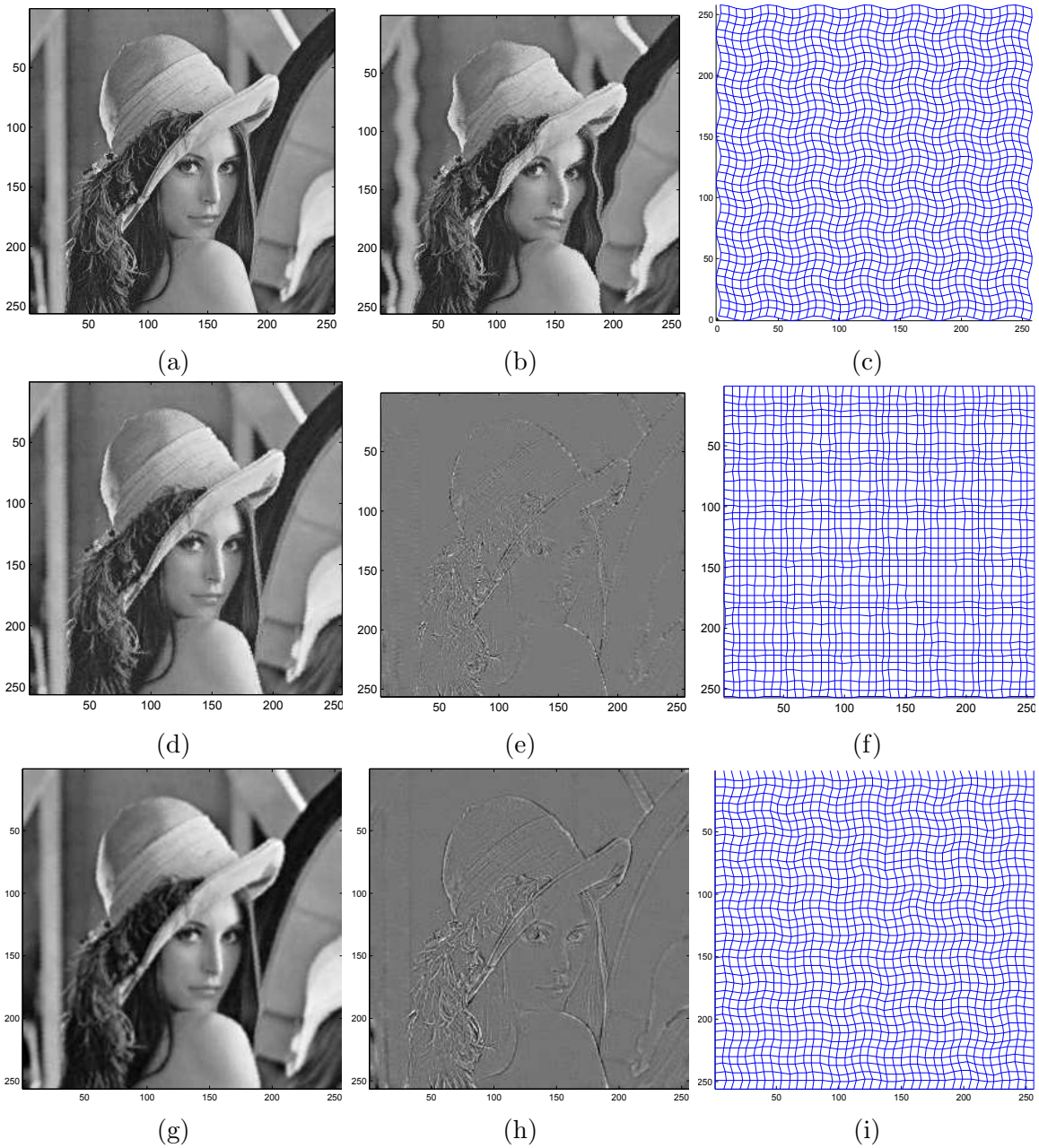


Figure 3: First row: (a) the fixed image, (b) the moving image, and (c) synthetic forward deformation grids. Second row: result from the optical flow method; (d) deformed image, (e) intensity differences, and (f) forward deformation grids. Third row: result from our proposed method; (g) deformed image, (h) intensity differences, and (i) forward deformation grids.

with the previous transformation, providing a smoother and more accurate outcome. It is shown in Figure 4g and Figure 4h that the transformation map from the composite method can depict clearly how the image is deformed. Please refer to main2DCShape.m for the Matlab code of this example.

6.3 3D registration

3D registration of synthetic images is illustrated in Figure 5. In this example, we register a sphere to a star-shape object. The moving image (sphere) is shown in Figure 5a. Figure 5b shows the fixed image (the star-shape object), which is generated by distributing cones slightly above a small sphere. The star is generated in a way that the fluid characteristic of diffeomorphic registration can be illustrated clearly. Figure 5c and Figure 5d show the registration results generated using $\gamma = 1.54$ and $\gamma = 0.1$ respectively. In diffeomorphic registration, tears or folding of parts of the image should not occur. However, an approximation can be obtained as can be seen in Figure 5c. We can observe that the cone remains attached to the sphere, but the part that links the cone to the sphere is compressed to be narrower. When γ is set to be at least 1.5376, diffeomorphisms are guaranteed, and the output is smooth. Note that the inequality $\gamma \geqslant 1.5376$ is not always sharp. Thus, when γ is set to a lower value, it may better able to capture sharp corners or other "non-smooth" feature of the image (see Figure 5d). The main Matlab file that run this is example is main3D.m.

6.4 3D registration of medical image

In the last example, we test our method on 3D medical images of size 181 x 217 x 181 voxels. We demonstrate the application of our algorithm on registering the anatomical models of brain images obtained from [1]. These models consist of 12 classes of labels, one representing the background, and each of the others represents different classes of brain tissues. We computed the Dice similarity for the overlap regions. Given two different labels, denoted by w_1 and w_2, the Dice similarity is given by:

$$D(w_1, w_2) = \frac{f(w_1 \cup w_2)}{f(w_1) + f(w_2)} \tag{28}$$

where $f(w)$ is a function that returns the volume of w. For identical labels, $D(w_1, w_2) = 1$ indicates a perfect overlap and small value of $D(w_1, w_2)$ implies a low overlapping. In these examples, we used three resolution levels for registration, the average computation time used in an iteration for each level is shown in Table 1. We compared our method again with the diffeomorphic version of the optical flow method [23]. The Dice similarity is computed for all the labels and the average of 10 model sets is summarized in a chart (Figure 6). From the chart, we can see that our proposed method produces a better match between the tissue types compared to the optical flow method. The 2D slices of a selected set of models are given in Figure 7. The first row are the slices of the fixed image (Figure 7a) and the moving image (Figure 7b). The comparison of results is given in the second row. The Matlab file that shows this example is given in main3DDiceSimilarity.m, however, to use this code, the reader should download the anatomic Brain model from [1] and put them in the current directory.

6.5 Conclusion

We have proposed a non-rigid image registration algorithm which is a hybrid of the optical flow method and FFD. The key aspect of the proposed method is that we use a consistent framework for

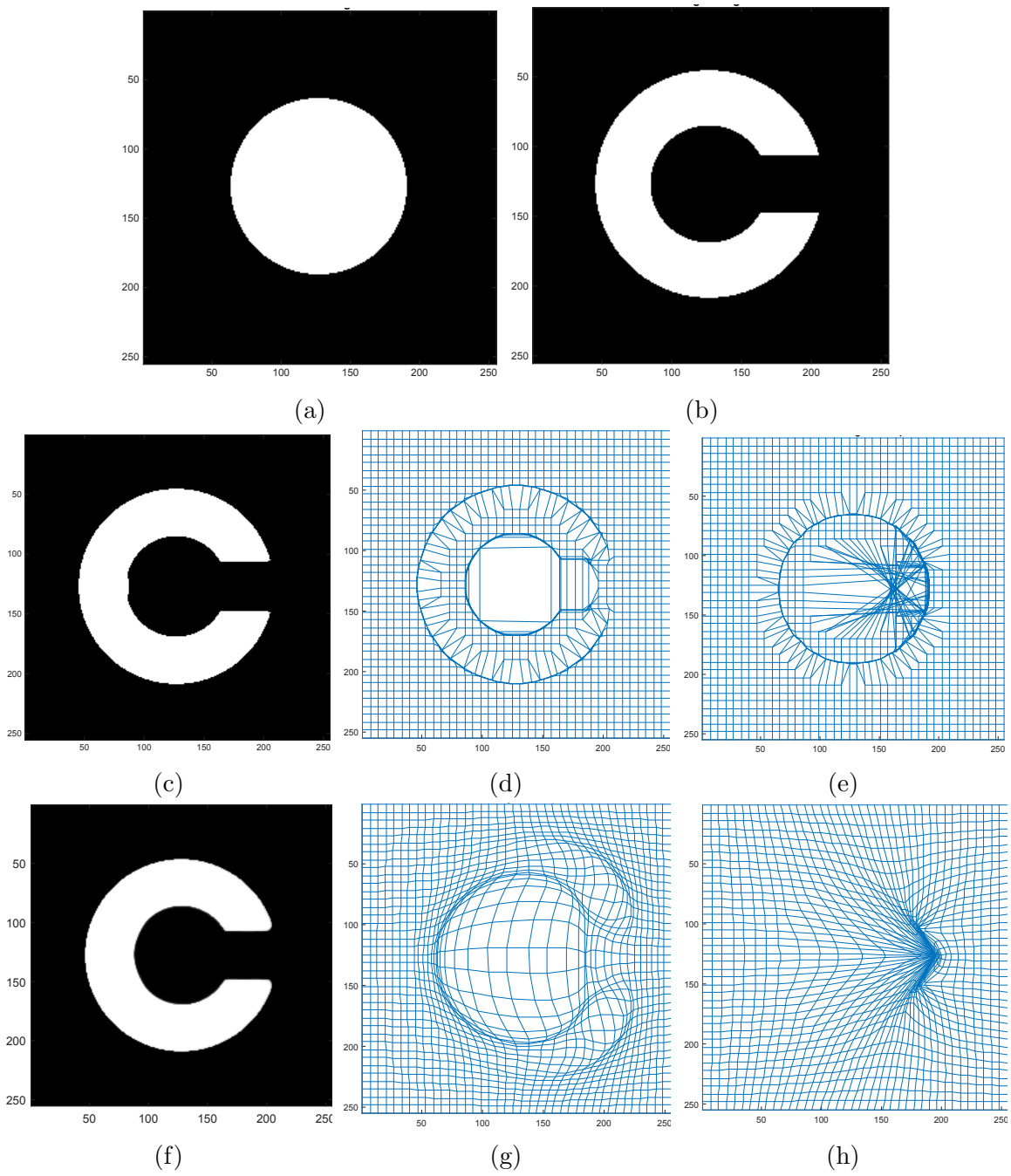


Figure 4: Comparison of our method (using the composition update) with the additive method. First row: (a) the moving image and (b) the fixed image. Second row: results from the additive method; (c) deformed image, (d) forward transformation grids, and (e) inverse transformation grids. Third row: results from the composition method; (f) deformed image, (g) forward transformation grids, and (h) inverse transformation grids.

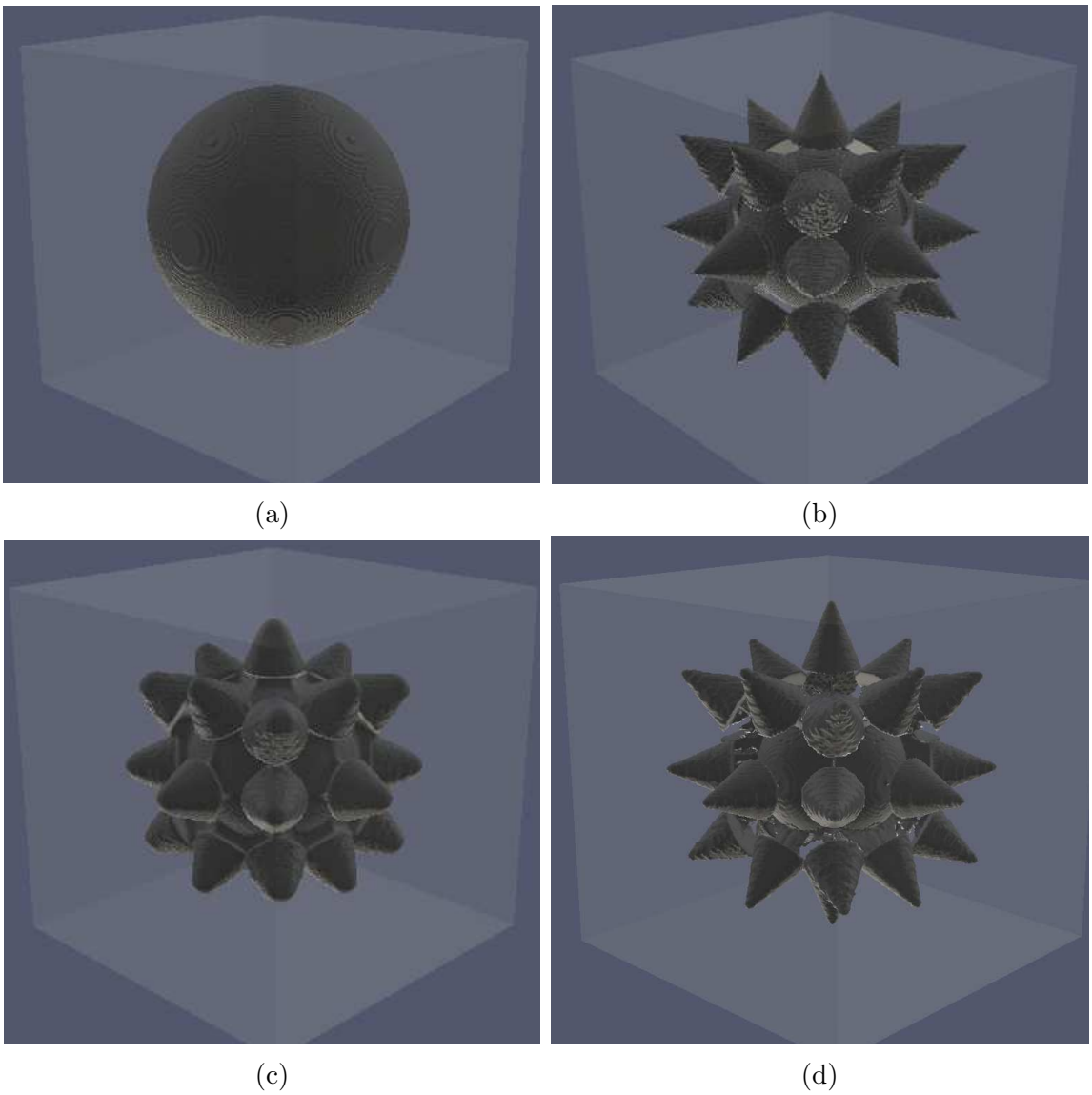


Figure 5: Registration of synthetic 3D images. (a) The moving image, (b) the fixed image, (c) deformed image computed using $\gamma = 1.54$, and (d) deformed image computed using $\gamma = 0.1$.

Table 1: Comparison of average computation time per iteration

method	proposed method	optical flow method
level 1	1.9631sec	1.3420 sec
level 2	3.6069 sec	9.2602 sec
level 3	20.3873 sec	85.8869 sec

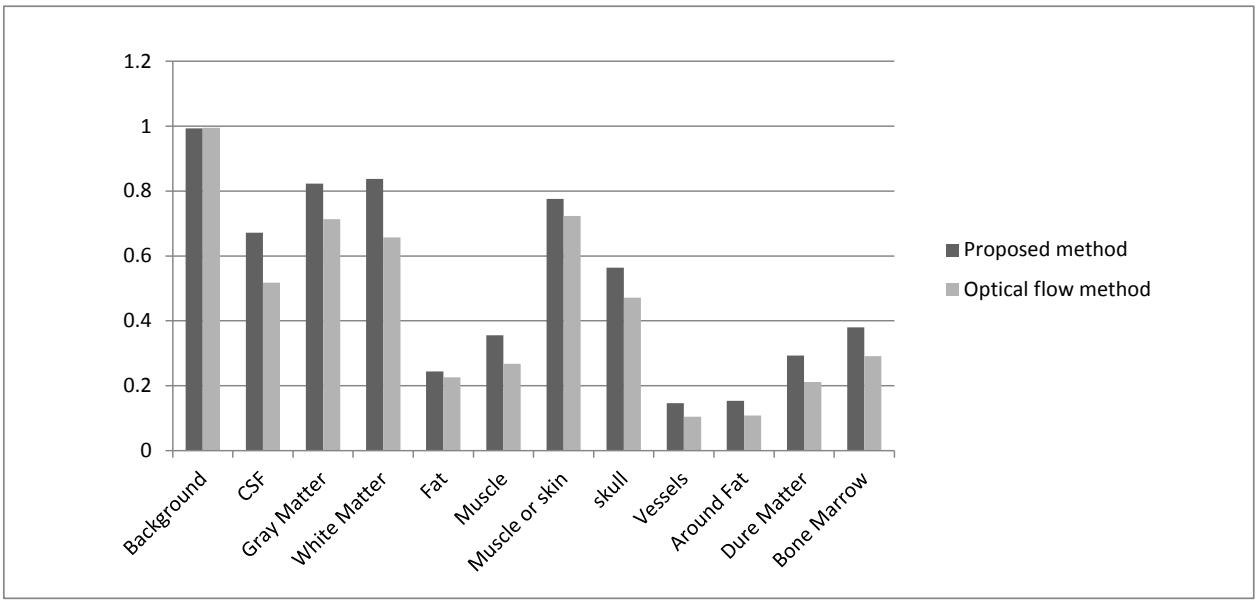


Figure 6: Comparison of the average of segmentation result from 10 model sets between the proposed method and the optical flow method.

the image representation and deformation field. The advantage is that deformation can be mapped to the moving image using composition method instead of interpolation method which is employed in the standard FFD. Also, by applying the concept of convolution, we proposed to use the filtering technique for the efficient computation of gradients and coefficients. The use of filter allows us to utilize parallel computing, hence, reduce the computation time significantly. We showed that our proposed method can produce a smooth transformation map between the images (even for large deformation). We also showed possible medical applications of our method, whereby we applied it for registration of brain images. Compared to the optical flow method, the output from our proposed method shows a better match between the registered image and the target image.

Acknowledgement

We gratefully acknowledge the financial support of the ITN-INSIST (grant number 289361) and the German Academic Exchange Program (DAAD). Y. Zhang was supported in part by NSF CAREER Award OCI-1149591.

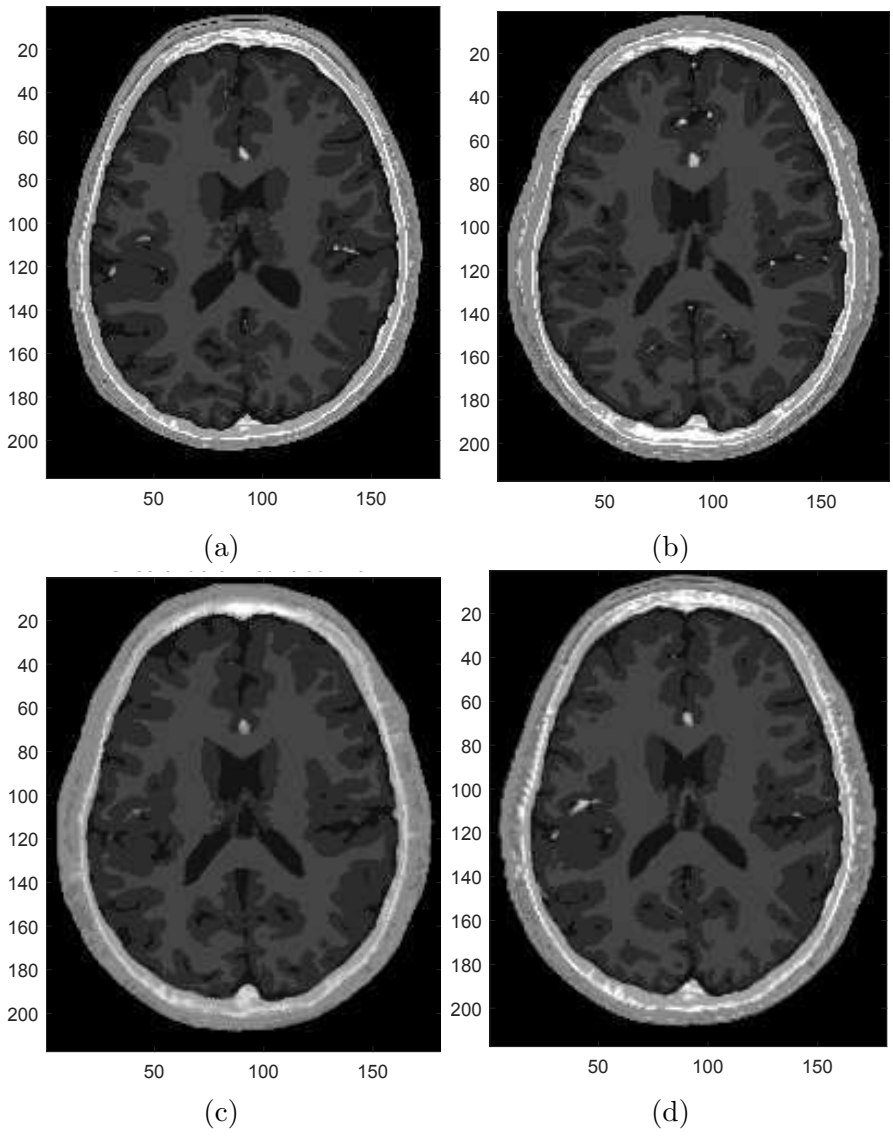


Figure 7: 2D slice of brain image. First row: (a) the moving image and (b) the fixed image. Second row: (c) result produced using our proposed method, (d) result form the optical flow method.

References

[1] Brainweb: Simulated brain database. Available at http://brainweb.bic.mni.mcgill.ca/brainweb/.

[2] Diffeomorphic log demons image registration. Available at http://www.mathworks.com/matlabcentral/fileexchange/39194-diffeomorphic-log-demons-image-registration.

[3] J. Ashburner. A fast diffeomorphic image registration algorithm. *NeuroImage*, 38(1):95 – 113, 2007.

[4] M. F. Beg, M. I. Miller, A. Trouvé, and L. Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International Journal of Computer Vision*, 61(2):139–157, 2005.

[5] M. Bro-Nielsen and C. Gramkow. Fast fluid registration of medical images. In KarlHeinz Höhne and Ron Kikinis, editors, *Visualization in Biomedical Computing*, volume 1131 of *Lecture Notes in Computer Science*, pages 265–276. Springer Berlin Heidelberg, 1996.

[6] P. Cachier, E. Bardinet, D. Dormont, X. Pennec, and N. Ayache. Iconic feature based nonrigid registration: the PASHA algorithm. *Computer Vision and Image Understanding*, 89(2-3):272 – 298, 2003. Nonrigid Image Registration.

[7] Y. Choi and S. Lee. Injectivity conditions of 2D and 3D uniform cubic B-spline functions. *Graphical Models*, 62(6):411 – 427, 2000.

[8] G. E. Christensen, R. D. Rabbitt, and M. I. Miller. Deformable templates using large deformation kinematics. *IEEE Transactions on Image Processing*, 5(10):1435–1447, Oct 1996.

[9] C. Davatzikos. Spatial transformation and registration of brain images using elastically deformable models. *Computer Vision and Image Understanding*, 66(2):207 – 222, 1997.

[10] J.C. Gee. On matching brain volumes. *Pattern Recognition*, 32(1):99 – 111, 1999.

[11] Y. Jia, Y. Zhang, and T. Rabczuk. A novel dynamic multilevel technique for image registration. *Computers and Mathematics with Applications*, 69(9):909–925, May 2015.

[12] J. Leng, G. Xu, and Y. Zhang. Medical image interpolation based on multi-resolution registration. *Computers & Mathematics with Applications*, 66(1):1 – 18, 2013.

[13] A. Pawar, Y. Zhang, Y. Jia, X. Wei, T. Rabczuk, C. L. Chan, and C. Anitescu. Adaptive FEM-based nonrigid image registration using truncated hierarchical b-splines. *A Special Issue of FEF 2015 in Computers and Mathematics with Applications*, 2016.

[14] X. Pennec, P. Cachier, and N. Ayache. Understanding the "demon's algorithm": 3D non-rigid registration by gradient descent. In Chris Taylor and Alain Colchester, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI'99*, volume 1679 of *Lecture Notes in Computer Science*, pages 597–605. Springer Berlin Heidelberg, 1999.

[15] D. Rueckert, P. Aljabar, R. A. Heckemann, J. V. Hajnal, and A. Hammers. Diffeomorphic registration using B-splines. In Rasmus Larsen, Mads Nielsen, and Jon Sporring, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2006*, volume 4191 of *Lecture Notes in Computer Science*, pages 702–709. Springer Berlin Heidelberg, 2006.

[16] D. Rueckert, L.I. Sonoda, C. Hayes, D.L.G. Hill, M.O. Leach, and D.J. Hawkes. Nonrigid registration using free-form deformations: application to breast MR images. *Medical Imaging, IEEE Transactions on*, 18(8):712–721, Aug 1999.

[17] R. Szeliski and J. Coughlan. Spline-based image registration. *International Journal of Computer Vision*, 22(3):199–218, 1997.

[18] J.-P. Thirion. Image matching as a diffusion process: an analogy with Maxwell's demons. *Medical Image Analysis*, 2(3):243 – 260, 1998.

[19] N.J. Tustison, B.B. Avants, and J.C. Gee. Directly manipulated free-form deformation image registration. *Image Processing, IEEE Transactions on*, 18(3):624–635, March 2009.

[20] M. Unser. Splines: a perfect fit for signal and image processing. *Signal Processing Magazine, IEEE*, 16(6):22–38, Nov 1999.

[21] B. C. Vemuri, J. Ye, Y. Chen, and C. M. Leonard. Image registration via level-set motion: applications to atlas-based segmentation. *Medical Image Analysis*, 7(1):1 – 20, 2003.

[22] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache. Non-parametric diffeomorphic image registration with the demons algorithm. In N. Ayache, S. Ourselin, and A. Maeder, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, volume 4792 of *Lecture Notes in Computer Science*, pages 319–326. Springer Berlin Heidelberg, 2007.

[23] T. Vercauteren, X. Pennec, A. Perchant, and N. Ayache. Diffeomorphic demons: efficient non-parametric image registration. *NeuroImage*, 45(1, Supplement 1):S61 – S72, 2009. Mathematics in Brain Imaging.

[24] C. V. Verhoosel, G. J. van Zwieten, B. van Rietbergen, and R. de Borst. Image-based goal-oriented adaptive isogeometric analysis with application to the micro-mechanical modeling of trabecular bone. *Computer Methods in Applied Mechanics and Engineering*, 284:138–164, February 2015.