Computer-Aided Design 58 (2015) 141–150

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Dynamic meshing for deformable image registration[☆]

Yiqi Cai[a], Xiaohu Guo[a,*], Zichun Zhong[a], Weihua Mao[b]

[a] *Department of Computer Science, University of Texas at Dallas, USA*

[b] *Department of Radiation Oncology, University of Texas Southwestern Medical Center, USA*

HIGHLIGHTS

- We study how the superimposed mesh structure would influence the Finite Element Method (FEM)-based image registration process.
- We propose a mesh generation algorithm based on how the mesh will influence the registration process, using the discrete Centroidal Voronoi Tessellation idea.
- We present a parallel algorithm to compute and update the mesh structure efficiently during image registration.

ARTICLE INFO

Keywords:
Deformable image registration
Centroidal Voronoi Tessellation (CVT)
Dynamic meshing
GPU

ABSTRACT

Finite element method (FEM) is commonly used for deformable image registration. However, there is no existing literature studying how the superimposed mesh structure would influence the image registration process. We study this problem in this paper, and propose a dynamic meshing strategy to generate mesh structure for image registration. To construct such a dynamic mesh during image registration, three steps are performed. Firstly, a density field that measures the importance of a pixel/voxel's displacement to the registration process is computed. Secondly, an efficient contraction–optimization scheme is applied to compute a discrete Centroidal Voronoi Tessellation of the density field. Thirdly, the final mesh structure is constructed by its dual triangulation, with some post-processing to preserve the image boundary. In each iteration of the deformable image registration, the mesh structure is efficiently updated with GPU-based parallel implementation. We conduct experiments of the new dynamic mesh-guided registration framework on both synthetic and real medical images, and compare our results with the other state-of-the-art FEM-based image registration methods.

1. Introduction

Finite element method (FEM) is a widely used technique to solve deformable image registration problems in the computer graphics, computer vision and medical imaging community. By introducing a superimposed mesh structure in the image domain, the speed and accuracy of image registration significantly depends on the mesh structure.

The traditional meshing methods usually target at generating a high quality mesh whose boundary resembles the surface of the biological object in one image [1]. Usually when the node number is fixed, the requirement of better surfaces approximation always implies a deterioration of mesh quality. There are a lot of algorithms [2–8] proposed to strive for the trade-off between these two conflicting factors.

We believe meshing for deformable image registration should focus on the registration process instead of one image (source or target image). Since the registration process is a dynamic process, the mesh structure should be updated accordingly. Three steps are performed in our algorithm for the dynamic mesh generation. Firstly, a density field that measures the importance of a pixel/voxel's displacement to the registration process is computed. Secondly, an efficient contraction–optimization scheme is applied to compute a discrete Centroidal Voronoi Tessellation (CVT) of the density field. Thirdly, the final mesh structure is constructed by its dual triangulation, with some post-processing to preserve the image boundary.

The main contribution of our work includes two aspects: on the one hand, we propose a mesh generation algorithm based on how the mesh will influence the registration process; on the other hand,

☆ This paper has been recommended for acceptance by Dr. Vadim Shapiro.
* Corresponding author.
E-mail addresses: yiqi.cai@utdallas.edu (Y. Cai), xguo@utdallas.edu (X. Guo), zichunzhong@utdallas.edu (Z. Zhong), weihua.mao@utsouthwestern.edu (W. Mao).

http://dx.doi.org/10.1016/j.cad.2014.08.009
0010-4485/© 2014 Elsevier Ltd. All rights reserved.

we present a parallel algorithm to compute and update the mesh structure efficiently.

2. Related work

2.1. FEM-based deformable image registration

Deformable image registration is a challenging research problem in the fields of image processing and medical imaging. As described in the book of Modersitzki [9], given the source image **S** and the target image **T**, the goal of registration is to establish the transformation **W** that optimizes an objective function, which includes two terms: one is the image matching criteria (image similarity) $\mathcal{D}(\mathbf{T}, \mathbf{S} \circ \mathbf{W})$, and the other is the weighted regularity of the transformation $\mathcal{R}(\mathbf{W})$:

$$\mathcal{D}(\mathbf{T}, \mathbf{S} \circ \mathbf{W}) + \beta \mathcal{R}(\mathbf{W}), \tag{1}$$

where β is the weight to balance the image matching criteria and regularity.

The transformation **W** is a mapping function that relates the source and target images together. Usually, it can be considered as a backward mapping, i.e. for every position **x** in the target domain, it is mapped to the corresponding location in the source image **S** through a displacement vector $\mathbf{u}(\mathbf{x})$:

$$\mathbf{W}(\mathbf{x}) = \mathbf{x} + \mathbf{u}(\mathbf{x}). \tag{2}$$

In this way, the deformed source image $\mathbf{S}\circ\mathbf{W}$ is aligned in the target domain. The term "deformable" in this paper is used to denote that the transformation **W** is non-rigid and spatially varying.

Many deformable image registration algorithms have been proposed in the past two decades. We refer the readers to a more complete survey by Aristeidis et al. [10]. In general, the deformable image registration algorithms mainly have three components: (1) a deformation model, (2) an image matching criteria, and (3) an optimization method.

The deformation model captures the nature of the transformation **W** to be recovered, and it plays two important roles in the registration problem. Firstly, for image registration problem, there is no unique transformation **W** to align the deformed source image $\mathbf{S}\circ\mathbf{W}$ with the target image **T**. The deformation model determines the regularity term $\mathcal{R}(\mathbf{W})$ that penalizes unwanted transformations. Thus, along with the proposed image matching criteria $\mathcal{D}(\mathbf{T}, \mathbf{S}\circ\mathbf{W})$, the image registration problem is converted to an optimization problem to obtain a physically-plausible solution. Secondly, the deformation model specifies the solution space of the transformation **W**. Usually Finite Element Method (FEM) is exploited [11–15] to achieve a balance between the efficiency of the optimization and the richness of the deformation model's description. By introducing a superimposed mesh structure on the image, FEM-based registration specifies the displacement vectors on the mesh nodes and the displacement field $\mathbf{u}(\mathbf{x})$ over the image domain is interpolated by the shape function [16], usually predefined by mesh elements. Therefore, FEM restricts the solution space of the image registration problem with much fewer degrees of freedom and ensures the smoothness of the transformation **W** by interpolation.

2.2. Mesh generation for deformable image registration

Generally speaking, the previous mesh generation algorithms for deformable image registration mainly concern two aspects: fidelity and quality. Fidelity measures the accuracy of the mesh boundary in capturing the anatomical structure, e.g., a mesh having its boundary conforming to the surface of underlying biological object is considered a good mesh. Quality measures the shape of mesh elements, such as the minimum angle in triangular elements in 2D case [17] or minimum dihedral angle of the tetrahedral elements in 3D case [1].

Mesh generation methods can be mainly divided into two categories: surface meshing and volumetric meshing. For surface meshing methods, they usually recover an explicit representation of the object surface, or construct an implicit function to describe the object surface from the source image [2], and then use the Marching Cubes algorithm to extract the iso-surface from the image [18]. As for the volumetric meshing methods, there are some recent literature about generating high-quality meshes, such as regular tilings and adaptive elements by using the octree method [3,4]. Delaunay refinement method [19,6] is an incremental mesh construction approach, which has been successfully applied to many practical applications. There are also several works [5–7] discussing about producing the volumetric mesh while preserving the object boundary and surfaces.

In this paper, we are going to show that fidelity may not be an important factor for mesh generation in deformable image registration. Instead, a mesh guided by the potential deformation vector field will be more important in providing sufficient degrees of freedom to drive the deformation.

2.3. Centroidal Voronoi Tessellation

To generate high-quality meshes, Centroidal Voronoi Tessellation (CVT) [20] is an effective tool that has been widely used. Given a set of sites $\mathbf{X} = \{\mathbf{x}_i | i = 0 \ldots n-1\}$ to sample the domain Ω, CVT is a special type of Voronoi Diagram, where each site $\mathbf{x}_i$ coincides exactly with the centroid of its Voronoi cell. There are two main approaches to compute CVT: one is the Lloyd relaxation [21], and the other is a quasi-Newton energy optimization solver [22].

For surface meshing, Peyre et al. [23] used a geodesic Voronoi Diagram over the surface to generalize CVT. However, the computations of geodesic Voronoi Diagram are very complicated, so Edelsbrunner and his co-authors computed the Restricted Voronoi Diagram (RVD) or Restricted Delaunay Triangulation (RDT) [24], instead of the geodesic Voronoi Diagram. A Restricted Centroidal Voronoi Tessellation is provided in [25], which requires all the sites to be constrained on the surface. Yan et al. [17] computed the CVT in 3D space via the 3D Euclidean distance as an approximation, and then intersected it with the surface. An alternative approach is to compute the CVT in the 2D parametric domain of the surface [26–28].

For volume meshing, Du and Wang [29] discussed an algorithm for tetrahedral meshing based on CVT in a 3D domain. Yan et al. [30] presented an efficient algorithm to compute the clipped Voronoi diagram with respect to a 3D volume and then generate its dual mesh. Alliez et al. [31] proposed a variational isotropic tetrahedral meshing method which minimizes an Optimal Delaunay Triangulation (ODT) energy.

This paper uses a discrete CVT approach [32–34] to generate dynamic mesh in each iteration of deformable image registration. Since the image domain is typically discretized into pixels (2D) or voxels (3D), the discrete Voronoi diagram can be computed by classifying the pixels/voxels into clusters, according to their distances w.r.t. the sites. An "energy increase rule" is proposed in this paper to efficiently swap pixels between neighboring clusters, in order to optimize the discrete CVT energy.

2.4. Other related works

This paper mainly studies the problem of distributing degrees of freedom during image registration. Our dynamic meshing method is different from the topological control techniques [35] for viscoelastic fluid simulation, the content-aware framework [36] for image wrapping, or the coupled simulation method [37] for interactive simulation of the surgical needle.

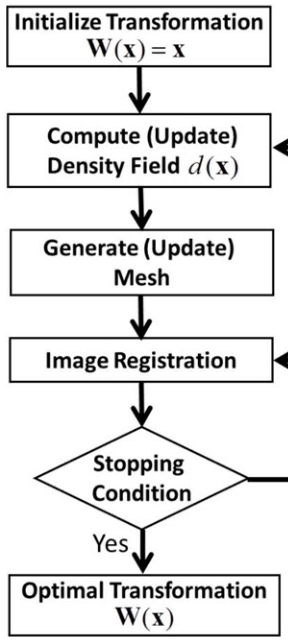

Initialize Transformation $\mathbf{W}(\mathbf{x}) = \mathbf{x}$
Compute (Update) Density Field $d(\mathbf{x})$
Generate (Update) Mesh
Image Registration
Remesh
Stopping Condition
Yes
No
No
Yes
Optimal Transformation $\mathbf{W}(\mathbf{x})$

Fig. 1. Flow chart of the proposed deformable image registration framework based on dynamic meshing.

3. Image registration framework

Fig. 1 illustrates a flow chart of the proposed deformable image registration framework based on dynamic meshing.

In this section, we present the general settings of the image registration algorithm in our implementation. We made two assumptions of the input images: first, the whole image region is the region of interest to solve the displacement vector field. Secondly, the two images should contain the same objects and the deformation is small enough w.r.t. the image size. Since we only focus on how the superimposed mesh structure influences the registration process, for the ease of presentation, we fix all other registration components, such as: the image matching criteria $\mathcal{D}$, transformation regularity $\mathcal{R}$, and optimization method. It should be noted that the proposed concepts and methods in the following sections can be extended to other regularizers and any differentiable image matching criteria (i.e. the image matching criteria is C^1 continuous).

We develop a multi-resolution sum of squared differences (MSSD) as image matching criteria. The main difference between MSSD and the traditional sum of squared differences (SSD) criteria is that MSSD uses an underlying multi-resolution scheme. Furthermore, in the traditional multi-resolution scheme, images are represented by pyramids. Image registration follows a multi-level approach: firstly, coarser levels are served to register the image and the result is considered as the initial guess of the finer levels. In order to avoid the inconsistency of the stopping conditions of different levels (i.e. we optimize different objective functions in different levels), we sum the objective function in each pyramid level and register all image pyramids simultaneously:

$$\mathcal{D}(\mathbf{T}, \mathbf{S} \circ \mathbf{W}) = \sum_i \sum_{\mathbf{x} \in \mathbf{T^i}} (\mathbf{T^i}(\mathbf{x}) - \mathbf{S^i}(\mathbf{W}(\mathbf{x})))^2, \tag{3}$$

where i is the pyramid level, $\mathbf{T^i}$ and $\mathbf{S^i}$ are the target image pyramid and source image pyramid at the ith level, respectively.

In the above statement, we have demonstrated the objective function of our proposed image registration method. Now we will discuss how to compute the optimal transformation **W** in the following of this section.

We use the linear elastic potential [38] to model the transformation regularizer. For this particular choice, the Navier–Lame equation [9] characterizes the force equilibrium between the stress of the elastic body and external force as:

$$(\lambda + \mu)\nabla(\mathbf{div}\,\mathbf{u}) + \mu\Delta\mathbf{u} = \mathbf{f}, \tag{4}$$

where λ and μ are Young's modulus and shear modulus of the elastic body, respectively. $\mathbf{u}$ is the vector of displacements at image pixels/voxels. $\nabla(\mathbf{div}\,\mathbf{u})$ is the gradient of the divergence of $\mathbf{u}$. $\Delta\mathbf{u}$ is the Laplacian operator of $\mathbf{u}$. $\mathbf{f}$ is the external force derived from the first-order derivative of the image matching criteria, in our case, i.e.:

$$\mathbf{f}(\mathbf{x}) = \sum_i 2(\mathbf{T^i}(\mathbf{x}) - \mathbf{S^i}(\mathbf{W}(\mathbf{x})))\nabla\mathbf{S^i}(\mathbf{W}(\mathbf{x})). \tag{5}$$

Following the general discretization approach with linear elements [16], a global equation describing the force equilibrium of mesh nodes can be written in a matrix form as:

$$\hat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}}, \tag{6}$$

where $\hat{\mathbf{K}}$ is the stiffness matrix, $\hat{\mathbf{u}}$ is the displacement vector at mesh nodes, and $\hat{\mathbf{f}}$ is the force vector at mesh nodes.

Note that we fix the boundary nodes of the mesh, i.e., we constrain the displacements of image boundary as zero throughout the registration process. In this way, the stiffness matrix $\hat{\mathbf{K}}$ is guaranteed to be full-rank. We adopt the fixed-point iteration [9] to solve the nodes' displacement vectors $\hat{\mathbf{u}}^{(k)}$ at iteration k. Given $\mathbf{W}^{(k)}(\mathbf{x})$ as the transformation at the kth step, the mesh structure is dynamically updated (as described in Section 4), leading to an updated stiffness matrix $\hat{\mathbf{K}}^{(\mathbf{k})}$. The force vector $\hat{\mathbf{f}}^{(k)}$ is reassembled using the nodes at the current transformation, and the nodes' displacement vector $\hat{\mathbf{u}}^{(k+1)}$ is solved by:

$$\hat{\mathbf{K}}^{(\mathbf{k})}\hat{\mathbf{u}}^{(k+1)} = \hat{\mathbf{f}}^{(k)}. \tag{7}$$

For updating the transformation $\mathbf{W}^{(k+1)}$ iteratively, we first interpolate the displacement field $\mathbf{u}^{(k+1)}$ over the image domain from the displacements on nodes $\hat{\mathbf{u}}^{(k+1)}$. According to [39]: for a smooth vector field, the Jacobian matrix of its exponential map is positive definite everywhere. Thus to get a diffeomorphic spatial transformation, instead of treating $\mathbf{u}^{(k+1)}$ as an additive deformation, its exponential map $\exp(\mathbf{u}^{(k+1)})$ is computed by the *diffeomorphic updating rule* [39], the transformation $\mathbf{W}^{(k+1)}$ is then updated through composition:

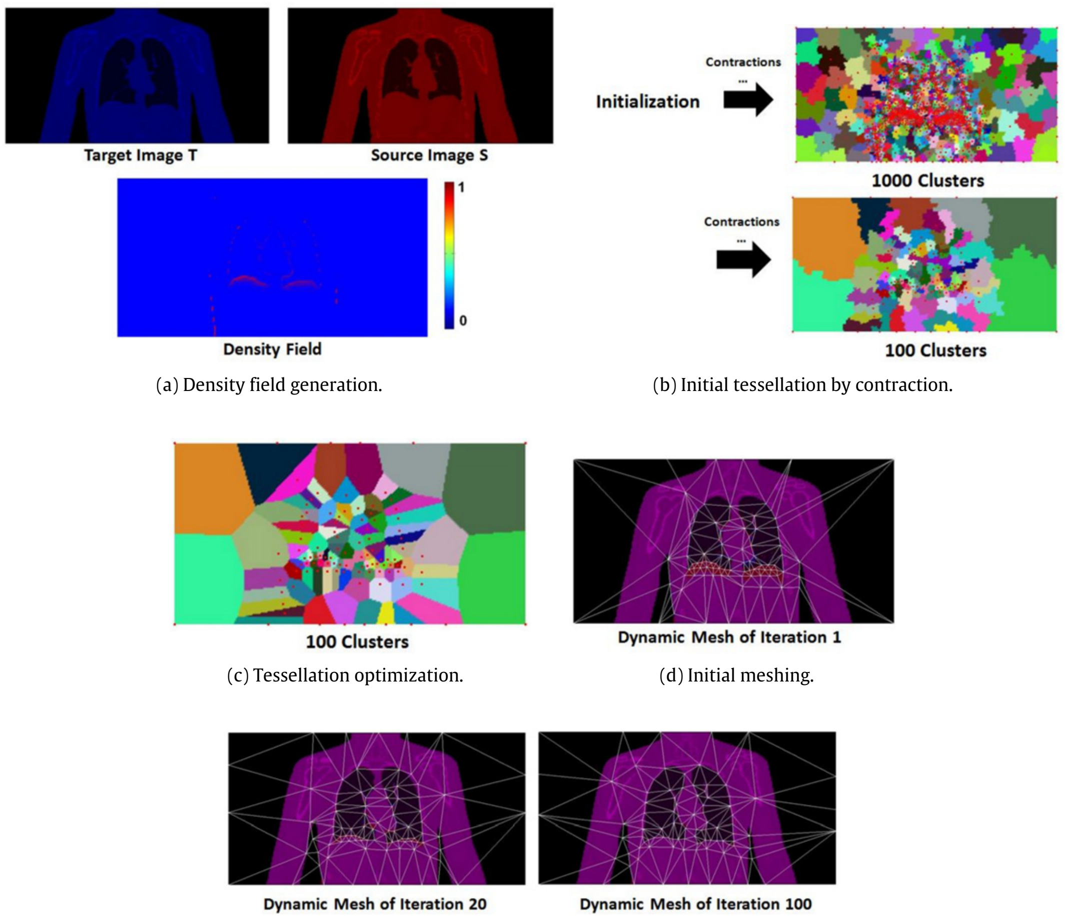
$$\mathbf{W}^{(k+1)} = \mathbf{W}^{(k)} \circ \exp(\mathbf{u}^{(k+1)}). \tag{8}$$

Note that this diffeomorphic updating rule will guarantee that the resulting transformation $\mathbf{W}^{(k+1)}$ is a diffeomorphism, i.e., there will be no flip-overs in the mapping.

4. Dynamic mesh generation algorithm

In general, the superimposed mesh plays a dual role in the image registration process. On the one hand, the discretization of the Navier–Lame equation (Eq. (4)) based on the mesh should guide the solution towards the physically meaningful solution that we are interested in. On the other hand, the registration process will be smooth by interpolating the displacement vector from the values solved on mesh nodes, even if the real deformation is complicated.

Targeting at producing high fidelity meshes, meshing algorithms usually try to resemble the surface of the biological object. However, they usually ignore the fact that deformation does not always happen on the surface, so that sampling much denser nodes on the surface of object is wasteful and inefficient. Meanwhile, large deformation can also occur inside the object. Thus, we



(a) Density field generation. (b) Initial tessellation by contraction.

(c) Tessellation optimization. (d) Initial meshing.

(e) Dynamic meshing.

Fig. 2. The partial results in each step of the proposed dynamic meshing for 2D deformable image registration.

propose the idea behind our algorithm: mesh generation should depend on the registration process, instead of only resembling the object in one static image (either the source or target image). Since the image registration is a dynamic process, the meshing should be dynamic throughout the registration as well.

4.1. *Scheme*

Given a source image **S** and a target image **T**, in this section, we illustrate the dynamic mesh generation approach for deformable image registration. Fig. 2 shows the partial results in each step of the proposed algorithm.

(1) *Density field generation* (Fig. 2(a)): a density field is computed to quantify the importance of each pixel/voxel in the current step of deformable registration. This is introduced in Section 4.2.
(2) *Initial tessellation by contraction* (Fig. 2(b)): an efficient contraction–optimization scheme is utilized to compute the initial tessellation based on the computed density field. The details are provided in Section 4.4.
(3) *Tessellation optimization* (Fig. 2(c)): we optimize the tessellation by minimizing the objective function given in Section 4.2. Details of the optimization are introduced in Section 4.5.
(4) *Initial meshing* (Fig. 2(d)): an initial mesh can be computed according to the optimized tessellation (Section 4.6).
(5) *Dynamic meshing* (Fig. 2(e)): in each iteration of the deformable image registration, we use the intermediate transformation $\mathbf{W}'$ to recalculate the density field with the warped source image $\mathbf{S} \circ \mathbf{W}'$. The tessellation and the mesh will be updated according to the new density field. More details are discussed in Section 4.7.

4.2. *Density field generation*

Since the goal of image registration is to find a physical plausible transformation **W** that optimizes the matching criteria $\mathcal{D}(\mathbf{T}, \mathbf{S}\circ\mathbf{W})$, the first-order derivative of the image matching criteria in Eq. (5), which is a force vector field $\mathbf{f}(\mathbf{x})$ for each pixel/voxel, is closely related to the registration process: for each pixel/voxel $\mathbf{x}$, the direction of $\mathbf{f}(\mathbf{x})$ points to the largest energy decreasing direction, and its L^2 norm denotes to the rate of the energy decrement.

We quantify the L^2 norm of $\mathbf{f}(\mathbf{x})$ on each pixel/voxel as its *importance for the registration process*, in the sense of minimizing the matching criteria in Eq. (3). Thus, we define the *density field for registration process* as:

$$d(\mathbf{x}) = \|\mathbf{f}(\mathbf{x})\|^2 + \alpha, \tag{9}$$

where α is a constant base value for the importance of the registration. From Fig. 2(d), we can obviously see that more degrees of freedom are desired at higher density regions.

4.3. *Objective function for CVT-based meshing*

The density field is used to drive the computation of discrete CVT for generating a density-guided mesh [20]. Let Ω be a discrete domain represented by image pixels/voxels, given a density field $d(\mathbf{x})$ defined on the domain Ω with positions of the pixels/voxels $\mathbf{x}$, our goal is to find the optimal tessellation $\{\mathbf{C}_i\}_{i=1}^n$ (a tessellation is a number of clusters and C_i is the cluster i, with n being the number of clusters) and its corresponding optimal sites $\{\mathbf{z}_i\}_{i=1}^n$ ($\mathbf{z}_i$ is the site corresponding to cluster i), by minimizing the following energy function:

$$E_{\text{tess}}(\{\mathbf{C}_i\}_{i=1}^n, \{\mathbf{z}_i\}_{i=1}^n) = \sum_{i=1}^{n} E_i(\mathbf{C}_i, \mathbf{z}_i) = \sum_{i=1}^{n} \sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x}) \|\mathbf{x} - \mathbf{z}_i\|^2, \tag{10}$$

where $E_i(\mathbf{C}_i, \mathbf{z}_i)$ is the clustering energy of cluster i with site $\mathbf{z}_i$. $E_{\text{tess}}(\{\mathbf{C}_i\}_{i=1}^n, \{\mathbf{z}_i\}_{i=1}^n)$ is the total energy of the tessellation. The tessellation satisfies $\mathbf{C}_i \cap \mathbf{C}_j = \emptyset$ for $i \neq j$, and $\cup_i \mathbf{C}_i = \Omega$.

For each cluster C_i, we define its mass m_i and its centroid $\mathbf{r}_i$ by:

$$m_i = \sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x}), \tag{11}$$

$$\mathbf{r}_i = \frac{\sum_{\mathbf{x}\in\mathbf{C}_i} \mathbf{x} d(\mathbf{x})}{\sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x})}. \tag{12}$$

We point out the following two observations: first, $E_i(\mathbf{C}_i, \mathbf{z}_i)$ is optimized when the site $\mathbf{z}_i$ coincides with the centroid $\mathbf{r}_i$. Secondly, the energy change of $E_i(\mathbf{C}_i, \mathbf{z}_i)$ due to the misalignment of the site $\mathbf{z}_i$ and the centroid $\mathbf{r}_i$ can be efficiently computed by a multiplication of the cluster mass m_i and the squared distance of the misaligned vector $\|(\mathbf{r}_i - \mathbf{z}_i)\|^2$, instead of summing over the energy on each individual pixel/voxel. We denote the second observation as the *energy increase rule*. As shown in Sections 4.4 and 4.5, this *energy increase rule* facilitates our optimization of Eq. (10). The remainder of the section is a simple proof of the two observations.

Consider the energy for cluster C_i:

$$E_i(\mathbf{C}_i, \mathbf{z}_i) = \sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x}) \|(\mathbf{x} - \mathbf{r}_i)\|^2 + \sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x}) \|(\mathbf{r}_i - \mathbf{z}_i)\|^2 + 2 \sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x}) \langle \mathbf{x} - \mathbf{r}_i, \mathbf{r}_i - \mathbf{z}_i \rangle. \tag{13}$$

Note that the third part of the above energy is essentially zero:

$$\sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x}) \langle \mathbf{x} - \mathbf{r}_i, \mathbf{r}_i - \mathbf{z}_i \rangle = \sum_{\mathbf{x}\in\mathbf{C}_i} \langle d(\mathbf{x})(\mathbf{x} - \mathbf{r}_i), \mathbf{r}_i - \mathbf{z}_i \rangle = 0. \tag{14}$$

Thus the clustering energy $E_i(\mathbf{C}_i, \mathbf{z}_i)$ consists of two parts: the first part is the compression energy $A(\mathbf{C}_i)$, which measures the sum of weighted squared distances from the centroid $\mathbf{r}_i$ to individual pixel/voxel $\mathbf{x}$:

$$A(\mathbf{C}_i) = \sum_{x\in\mathbf{C}_i} d(\mathbf{x}) \|(\mathbf{x} - \mathbf{r}_i)\|^2. \tag{15}$$

The second part is the misalignment energy $B(m_i, \mathbf{r}_i, \mathbf{z}_i)$, which measures the energy increased due to the misalignment of the site $\mathbf{z}_i$ and the centroid $\mathbf{r}_i$:

$$B(m_i, \mathbf{r}_i, \mathbf{z}_i) = \sum_{x\in\mathbf{C}_i} d(x) \|(\mathbf{r}_i - \mathbf{z}_i)\|^2 = m_i \|(\mathbf{r}_i - \mathbf{z}_i)\|^2. \tag{16}$$

The misalignment energy achieves its minimum 0, if and only if the site $\mathbf{z}_i$ is located at the centroid $\mathbf{r}_i$. Any misalignment will result in the energy increase by a multiplication of the cluster mass m_i and the squared distance of the misaligned vector $\|(\mathbf{r}_i - \mathbf{z}_i)\|^2$.

4.4. Initial tessellation by contraction

Inspired by the surface simplification idea [40], we utilize a contraction operation to partition the pixels/voxels into n clusters, where n is a user specified number of the mesh nodes. The initial tessellation that conforms to the density distribution $d(\mathbf{x})$ is achieved by successively applying the contraction operation.

At first, each pixel/voxel can be treated as a cluster, which may form pairs with its direct neighboring clusters. When a pair of clusters is contracted to a new cluster, a contraction cost is associated with this contraction operation. For a contraction operation $(\mathbf{C}_i, \mathbf{C}_j) \to \mathbf{C}_k$, the contraction cost is defined as: $E_k(\mathbf{C}_k, \mathbf{r}_k) - E_i(\mathbf{C}_i, \mathbf{r}_i) - E_j(\mathbf{C}_j, \mathbf{r}_j)$. All possible contraction operations, with the corresponding costs as the key factor, are inserted into a contraction heap, which records all the possible contractions in current tessellation.

Then, the total number of clusters is reduced to n by iteratively performing the least-cost contraction in the heap. Each time after the least-cost pair is selected, only a local update is needed to maintain the validity of the contraction heap: the remaining pairs of the two contracted clusters in the heap are deleted, and the potential contractions between the new cluster and its direct neighbors are inserted. This step is iteratively performed until the number of clusters is reduced to n.

The most time-consuming part in this step is the computation of the contraction cost. As shown in the Appendix, this cost can be calculated efficiently by using the *energy increase rule* as mentioned in Section 4.3. In our implementation, to speed up the contraction process in 3D applications where there are huge number of voxels, each block of voxels is treated as a cluster at the beginning.

4.5. Tessellation optimization

There is no surprise that the initial tessellation by greedily contracting the least-cost pair of clusters cannot minimize the objective function in Eq. (10). A pixel/voxel cannot freely decide which cluster it should reside in among the final n clusters. Each time a cluster pair is contracted, the pixels/voxels in the two clusters are bound to reside in the same cluster. Thus we relax the binding between pixels/voxels and clusters in the tessellation optimization.

Tessellation $\{\mathbf{C}_i\}_{i=1}^n$ is updated according to the testing of the boundary pixels/voxels of each cluster. Here a boundary pixel/voxel of $\mathbf{C}_i$ is the one that has at least a neighboring pixel/voxel which does not belong to $\mathbf{C}_i$. For a boundary pixel/voxel $\mathbf{x} \in \mathbf{C}_i$, we denote the set of clusters that the neighboring pixels/voxels of $\mathbf{x}$ reside in as $\mathbf{B}_\mathbf{x}$. We need to test whether changing its clustering to $C_j \in \mathbf{B}_\mathbf{x}$ will decrease the energy in Eq. (10) or not. Actually, this energy change is a local operation, which only involves two clusters. To be more precise, as shown in the Appendix, the energy change can be efficiently computed by the *energy increase rule* as mentioned in Section 4.3. If the energy change is less than 0, it means the pixel/voxel swap can decrease the energy of the objective function so as to optimize the tessellation. Otherwise, the current tessellation is best suitable for the tested pixel/voxel, and there is no further operation needed. If there is more than one cluster in $\mathbf{B}_\mathbf{x}$ that can optimize the tessellation, we choose the one that can lead to the largest decrease of energy.

We propose a parallel algorithm for tessellation optimization in Algorithm 1. During each iteration, first we assign all the possible swaps for the boundary pixels/voxels in parallel by performing the swap testing. Note that the swap-testing operation for boundary pixels/voxels can be performed parallelly in each iteration. Then, we change the cluster ID for these pixels/voxels and update their mass and corresponding centroids for all clusters.

When there is no swap that can optimize the tessellation, the objective function in Eq. (10) reaches its local minimum. The corresponding tessellation is the locally-optimized CVT that conforms to the density field $d(\mathbf{x})$.

4.6. Initial meshing

With the optimized tessellation computed in the previous step, we can place mesh nodes at the position of sites and connect mesh nodes using the adjacency information of the tessellation. The only challenge here is that the mesh does not cover the whole region of interest, particularly at the region of clusters on the boundary

Table 1
Computational statistics of 2D and 3D image registration experiments by our proposed dynamic meshing method.

Experiments	Data resolution	# Iter.	# Nodes	Total time (s)	Regist. time (s)	Initial tessell. time (s)	Dynamic meshing time		
							Tessell. optimization (s)	Mesh construction (s)	Bary. computation (s)
Circle to leaf	1024×256	160	100	17.05	6.04	1.67	8.99	0.35	0.72
Circle to epsilon	512×512	400	100	28.06	11.60	1.59	12.68	0.82	1.37
3D Lung Respiration	$256 \times 256 \times 132$	100	3000	114.53	30.35	1.73[a]	73.28	4.06	5.02

Note: # Iter.: number of iterations in registration. Regist. time: registration time. Bary. computation: voxel's barycentric coordinate computation. Registration time and dynamic meshing time include all iterations' time during the registration.
[a] In order to speed up the contraction process, at the beginning, we treat each block of voxels (including $4 \times 4 \times 4$ voxels), instead of each voxel, as a cluster.

Algorithm 1 Tessellation optimization algorithm

```
1:  repeat
2:      for each cluster $\mathbf{C}_i$ in parallel do
3:          for each pixel/voxel $\mathbf{x} \in \mathbf{C}_i$ in parallel do
4:              initialize the best suited cluster ID $c_{best} = i$;
5:              initialize the decreased energy $d_{best} = 0$;
6:              collect the set of clusters $\mathbf{B_x}$ that all neighboring pixels/voxels of $\mathbf{x}$ reside;
7:              for each $\mathbf{C}_j \in \mathbf{B_x}$ and $\mathbf{C}_j \neq \mathbf{C}_i$ do
8:                  calculate the energy decrease $d_j$ by swapping $\mathbf{x}$ from $\mathbf{C}_i$ to $\mathbf{C}_j$
9:                  if $d_j < d_{best}$ then
10:                     $d_{best} \leftarrow d_j$
11:                     $c_{best} \leftarrow j$
12:                 end if
13:             end for
14:             if $c_{best} \neq i$ then
15:                 $\mathbf{x}$ should be swapped to cluster $c_{best}$
16:             end if
17:         end for
18:     end for
19:     for each cluster $\mathbf{C}_i$ in parallel do
20:         update the mass $m_i$ and centroid $\mathbf{r}_i$
21:     end for
22: until the change of Eq. (10) is less than $\epsilon$
```

and corners of the image. This is because the sites always will be set at the positions of the centroids. This problem can be handled through a post-processing by using Constrained Delaunay triangulation (CDT). For the clusters on the boundary and corners of the image, we treat the mesh nodes differently: (1) for the sites of the boundary clusters, we set it as the centers of all boundary pixels/voxels (for instance: it is the middle point of an edge or a face); (2) for the corner clusters, usually each cluster only contains one corner of the image, the node is set exactly at the corner. Since the nodes have been moved, the connectivity defined by the adjacency of the original tessellation may not be a valid triangulation any more. We constrain the edges between the inner clusters using the adjacency information of the tessellation, while the edges associated with the nodes of the boundary or corners are re-triangled by CDT to maintain the "Delaunay-like" property.

4.7. Dynamic meshing

During the image registration process, with the update of the transformation $\mathbf{W}^{(k)}$ at each iteration of the registration, the importance field $\|\mathbf{g}(\mathbf{x})\|^2$ characterizing the importance of each pixel/voxel to minimize the matching criteria, changes in each step. So does the corresponding mesh.

During the registration process, we apply the iterative optimization algorithm to update the meshing. In each iteration with the current transformation $\mathbf{W}'$, the density field can be recomputed by the warped source image $\mathbf{S} \circ \mathbf{W}'$. We compute a new tessellation with the updated density field. The tessellation optimization can start from the CVT result of its earlier iteration. Finally, we can obtain the new dynamic mesh generated by CDT from the new optimized tessellation.

In order to apply the generated dynamic mesh in image registration, an additional step that computes the barycentric coordinate of each pixel/voxel is needed. We followed the parallel approach as [41] in our implementation.

5. Experiments and results

We adopt a hybrid implementation on both CPU and GPU. The initial tessellation step by contracting least-cost cluster pairs successively is implemented on CPU using C++, while the registration and the remainder of the dynamic meshing steps are implemented on GPU using CUDA 5.0.

For the hardware platform, the experiments are run on a desktop computer with Intel(R) Core i7-4770 CPU with 3.40 GHz, 32 GB DDR3 RAM, and NVIDIA GeForce GTX 660 GPU with 2 GB GDDR5 video memory.

5.1. Parameters and configuration

In this subsection, we introduce the parameters that we choose in our experiments.

In the registration setting, for the MSSD image matching criteria in Eq. (3), we set the pyramid level as 3 in all the experiments. Since the weight β balances the image matching criteria and the regularity of the transformation, and its value depends on the domain size, the number of mesh nodes, and the variation of the image intensity, we use an additional step to determine how much penalty should be employed, instead of using an absolute value. In the first iteration of registration, when the transformation $\mathbf{W}^{(1)}$ is at hand, the image matching criteria term $\mathcal{D}(\mathbf{T}, \mathbf{S} \circ \mathbf{W}^{(1)})$ and the regularity term $\mathcal{R}(\mathbf{W}^{(1)})$ are computed, and β is set to normalize the energy ratio $\frac{\beta \mathcal{R}(\mathbf{W}^{(1)})}{\mathcal{D}(\mathbf{T}, \mathbf{S} \circ \mathbf{W}^{(1)})}$ as 0.02, then it is fixed for the rest of the iterations.

In our dynamic meshing implementation, we first calculate the average value of the importance $\|\mathbf{f}(\mathbf{x})\|^2$. The constant base value α is set as 0.1 times the average value. For the parameter ϵ in Algorithm 1 during tessellation optimization, we set it as 0 to get the exact local minimum of Eq. (10). Note that the computation time and the accuracy of this step can be balanced by the value of ϵ, e.g., setting $\epsilon = 0.0001$ instead would accelerate the tessellation optimization step by allowing tessellation inaccuracy.

Table 1 gives the computational statistics of the 2D and 3D image registration experiments by our proposed dynamic meshing method. It is clear that our method has high speed both in the registration and dynamic meshing steps.

Table 2 gives the time performance with different number of nodes on the registration of 3D Lung Respiration with 100 iterations.

5.2. 2D image registration

Our first 2D experiment illustrates the main feature of our dynamic meshing framework. We use a circle with radius of 60 pixels

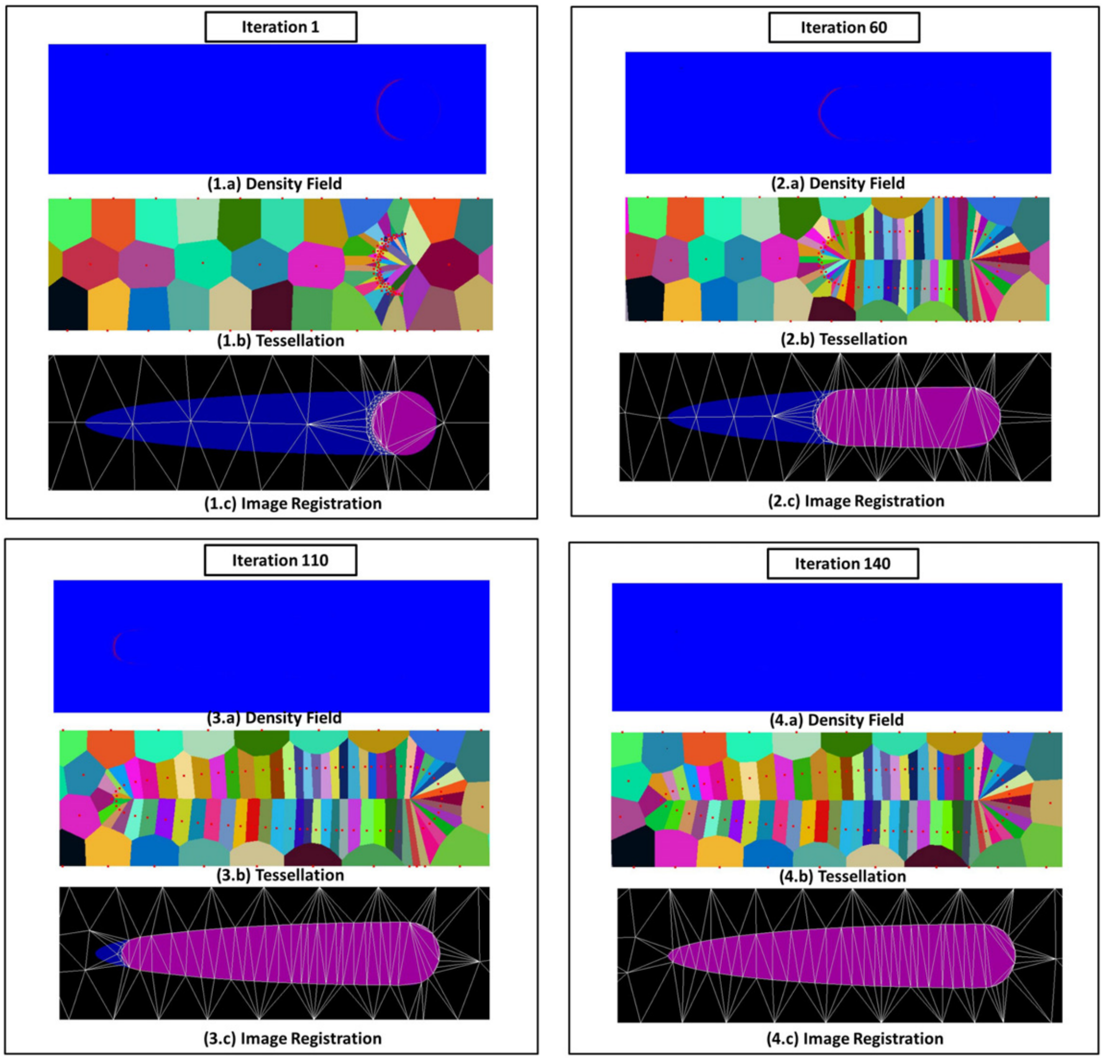


Fig. 3. The circle to "leaf" shape registration using dynamic mesh generation framework.

as source image, and the target image is a "leaf" shape, whose right half is the same semi-circle while the left half is a semi-ellipse with a semi-major axis of 600 pixels. We assign 100 nodes for the Circle to "Leaf" Shape registration experiment. In Fig. 3, we can clearly see that denser nodes are placed at the regions where local deformation is expected. In iteration 1, the right semi-circle only has the base value of the importance of the registration since the source image and the target image perfectly match at that region. As the registration goes on, once the boundary of the deformed image gets misaligned a little bit due to the interpolation from the displacement vector solved from mesh nodes, the density of the node in the misaligned region increases correspondingly. Thus more degrees of freedom will be generated automatically to "drive" the boundary aligned. At the end of the registration, the nodes are distributed evenly at the boundary of the deformed circle.

In order to evaluate the usefulness of our proposed dynamic meshing method, we conduct a "Circle to Epsilon" experiment and compare our result with two widely used meshes: regular mesh and high fidelity mesh as shown in Fig. 5(a) and (b). Regular mesh is widely used in spline-base free form of deformations (FFDS) [42,43], while high fidelity mesh is the desired mesh in the traditional image registration algorithms [1,8], which can well capture the boundary of the object structure. However, both these two kinds of meshes are static.

The source image is set as a circle with radius of 120 pixels, while the target image is an epsilon shape. The challenge here is we set the entrance into the epsilon shape to be 5 pixel width. We use 100 nodes in all the three kinds of meshes. The regular mesh is generated by the 10×10 rectangular grid, while the high fidelity mesh is generated using the popular open source of the meshing software CGAL [44]. CGAL produces a mesh using the Delaunay refinement method. As it clearly shows in Figs. 4 and 5, regular mesh and high-fidelity mesh are trapped in the narrow entrance due to the limited degrees of freedom. However, our dynamic meshing approach is able to adjust its nodes to "fit" the extreme shape. Fig. 4(d) shows our final result of transformation is physically plausible.

5.3. 3D image registration

To illustrate the efficiency and usability of our method, we applied our dynamic meshing framework to 3D Lung Respiration volume data, which is a set of torso images acquired from a digital phantom, XCAT [45].

We compare our method with variational image meshing (VIM) [8], which preserves the boundary of the volume data. There are 3150 nodes used in their mesh, while 3000 nodes were used in ours. The noticeable difference is that our mesh is much denser at the bottom of the lung, which requires more degrees of freedom for the local deformation. Fig. 6 shows our dynamic meshing strategy can facilitate the registration process and lead to a good

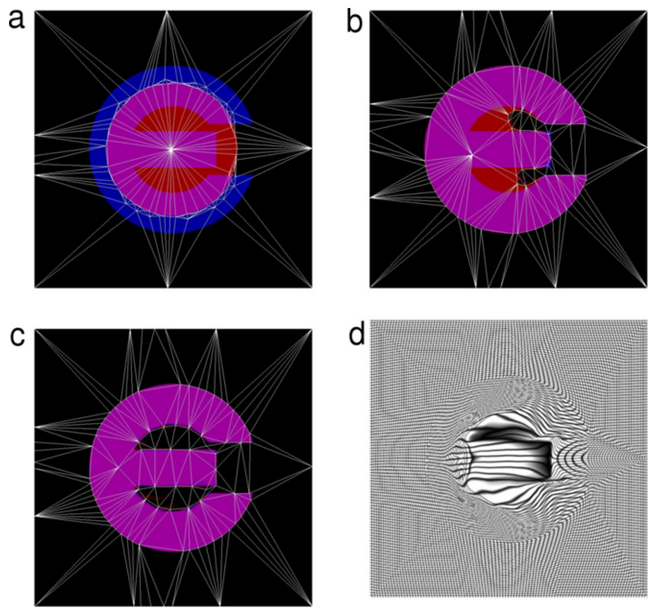


Fig. 4. Circle to Epsilon dynamic meshing. (a) Initial mesh of iteration 1. (b) Dynamic mesh of iteration 254. (c) Dynamic mesh of iteration 400. (d) Transformation of the dynamic meshing method at iteration 400.

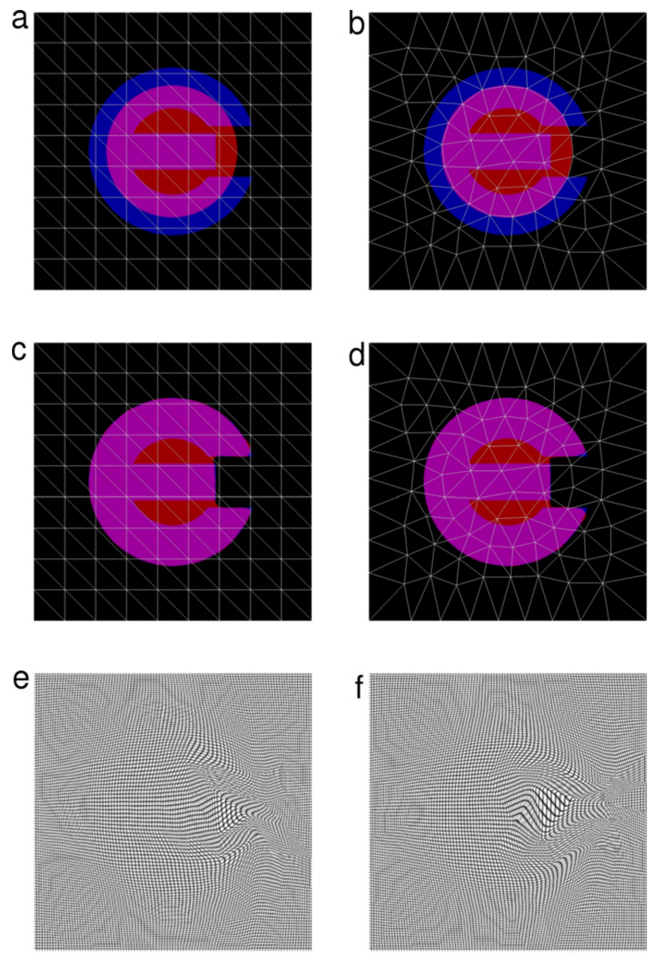


Fig. 5. Circle to Epsilon using static meshes. (a) Initial regular mesh. (b) Initial fidelity mesh. (c) Final registration result of the regular meshing method at iteration 400. (d) Final registration result of the fidelity meshing method at iteration 400. (e) Final transformation of the regular meshing method at iteration 400. (f) Final transformation of the fidelity meshing method at iteration 400.

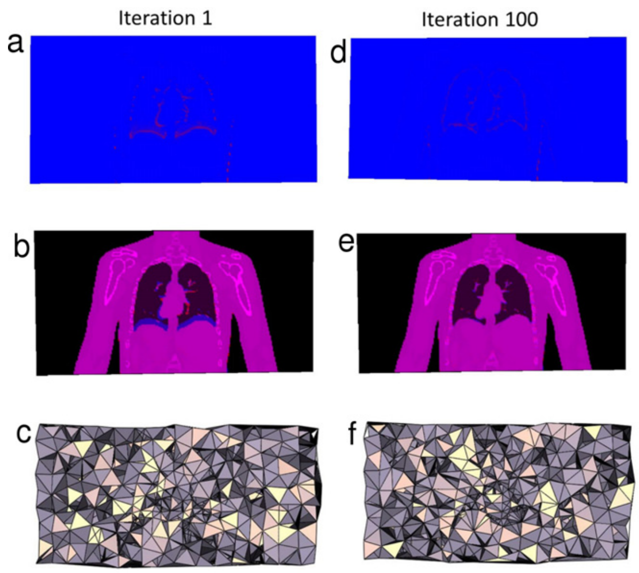


Fig. 6. 3D Lung Respiration. (a) Initial density field. (b) Registration result of iteration 1. (c) Dynamic mesh of iteration 1. (d) Density field of iteration 100. (e) Registration result of iteration 100. (f) Dynamic mesh of iteration 100.

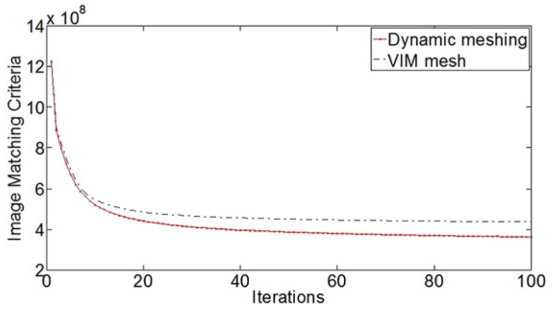


Fig. 7. Energy curves of the dynamic meshing method and the VIM mesh method.

registration result. Because it is difficult to visualize the difference between our dynamic mesh and VIM mesh in 3D case, we use the energy curves to compare. Fig. 7 shows the energy curves of the dynamic meshing method and the VIM mesh method and we can see that our proposed dynamic meshing method has faster convergence speed.

6. Limitation and future work

The major limitation of our work is our mesh construction. As shown in Eq. (10), the meshing objective function we proposed only considers the optimal tessellation $\{\mathbf{C}_i\}_{i=1}^n$ and the corresponding optimal sites $\{\mathbf{z}_i\}_{i=1}^n$. This strategy sets more degrees of freedom at regions where local deformation is expected. However, it does not take mesh element into account. The mesh elements are constructed directly by the dual graph and the CDT post-processing step described in Section 4.6.

The dual triangulation of CVT usually generates well-shaped elements. In our implementation, when the degrees of freedom are not large enough, we observe bad-shaped elements. Mesh quality deteriorates when we apply the CDT post-processing to cover the whole region of interest, i.e. small angles ($<15^\circ$) in triangular elements and small dihedral angle ($<10^\circ$) of tetrahedral elements. However it should be noted that bad-shaped elements will not cause flip-over in our solved transformation since the diffeomorphic update rule described in Section 3 always guarantees a diffeomorphic spatial transformation.

Table 2
Time performance with different number of nodes on the registration of 3D Lung Respiration with 100 iterations.

# Nodes	Initial tessell. time (s)	Dynamic meshing time		
		Tessell. optimization (s)	Mesh construction (s)	Bary. computation (s)
500	1.82	102.63	0.41	8.05
1000	1.80	81.75	1.13	6.37
3000	1.73	73.28	4.06	5.02
5000	1.66	74.11	8.74	4.33

Note: Bary. computation: voxel's barycentric coordinate computation. Dynamic meshing time includes all iterations' time during the registration.

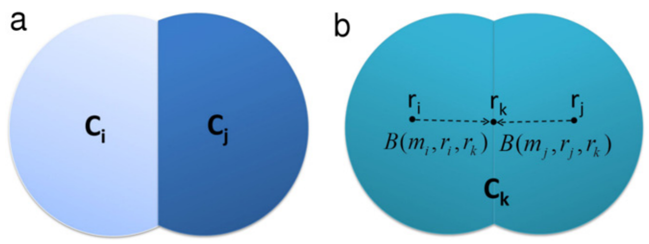


Fig. 8. Cluster-pair contraction during the initialization of tessellation. (a) Before contraction. (b) The contraction cost can be calculated efficiently by the sum of 'two misalignment energy terms.

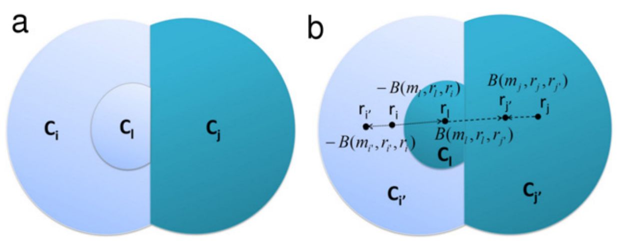


Fig. 9. Pixel-swapping during tessellation optimization. (a) Before voxel swap. (b) The change of energy during swap can be calculated efficiently.

Investigating how the mesh elements can influence the registration process and integrating the mesh elements into the efficient optimization scheme is an interesting direction for our future work.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions. This research work was partially supported by Cancer Prevention and Research Institute of Texas (CPRIT) under Grant No. RP110329, and National Science Foundation (NSF) under Grant Nos. IIS-1149737 and CNS-1012975.

Appendix

In this appendix, we show how the *energy increase rule* mentioned in Section 4.3 can be utilized to efficiently compute the cluster-pair contraction and cluster optimization.

A.1. Cluster-pair contraction $(\mathbf{C}_i, \mathbf{C}_j) \rightarrow \mathbf{C}_k$

If we contract a pair of clusters $(\mathbf{C}_i, \mathbf{C}_j)$ into $\mathbf{C}_k$, the new clustering energy of $\mathbf{C}_k$ with its centroid $\mathbf{r}_k$ is:

$$\begin{aligned} E_k(\mathbf{C}_k, \mathbf{r}_k) &= A(\mathbf{C}_k) \\ &= \sum_{\mathbf{x}\in\mathbf{C}_k} d(\mathbf{x})\|\mathbf{x}-\mathbf{r}_k\|^2 \\ &= \sum_{\mathbf{x}\in\mathbf{C}_i} d(\mathbf{x})\|\mathbf{x}-\mathbf{r}_k\|^2 + \sum_{\mathbf{x}\in\mathbf{C}_j} d(\mathbf{x})\|\mathbf{x}-\mathbf{r}_k\|^2 \\ &= A(\mathbf{C}_i) + B(m_i, \mathbf{r}_i, \mathbf{r}_k) + A(\mathbf{C}_j) + B(m_j, \mathbf{r}_j, \mathbf{r}_k). \end{aligned} \tag{17}$$

Therefore, the contraction cost is

$$\begin{aligned} A(\mathbf{C}_k) - (A(\mathbf{C}_i) + A(\mathbf{C}_j)) &= B(m_i, \mathbf{r}_i, \mathbf{r}_k) + B(m_j, \mathbf{r}_j, \mathbf{r}_k) \\ &= m_i\|(\mathbf{r}_i - \mathbf{r}_k)\|^2 + m_j\|(\mathbf{r}_j - \mathbf{r}_k)\|^2. \end{aligned} \tag{18}$$

As shown in Fig. 8, the contraction cost can be computed efficiently by keeping track of the mass and centroid of each cluster. It is simply the sum of two misalignment energy terms. After each contraction, the mass and centroid of the new cluster $\mathbf{C}_k$ can be updated by:

$$m_k = m_i + m_j, \tag{19}$$

$$\mathbf{r}_k = \frac{m_i\mathbf{r}_i + m_j\mathbf{r}_j}{m_i + m_j}. \tag{20}$$

A.2. Cluster optimization, swapping $\mathbf{C}_l$ from $\mathbf{C}_i$ to $\mathbf{C}_j$

Let us consider swapping a pixel $\mathbf{C}_l$ from $\mathbf{C}_i$ to $\mathbf{C}_j$. Suppose after swapping, $\mathbf{C}_i$ becomes $\mathbf{C}_{i'}$ and $\mathbf{C}_j$ becomes $\mathbf{C}_{j'}$, i.e., $\mathbf{C}_i = \mathbf{C}_{i'} \cup \mathbf{C}_l$, and $\mathbf{C}_{j'} = \mathbf{C}_j \cup \mathbf{C}_l$, as shown in Fig. 9.

From Eq. (17), the clustering energy of $\mathbf{C}_i$ and $\mathbf{C}_j$ are:

$$A(\mathbf{C}_i) = A(\mathbf{C}_{i'}) + B(m_{i'}, \mathbf{r}_{i'}, \mathbf{r}_i) + A(\mathbf{C}_l) + B(m_l, \mathbf{r}_l, \mathbf{r}_i). \tag{21}$$

$$A(\mathbf{C}_{j'}) = A(\mathbf{C}_j) + B(m_j, \mathbf{r}_j, \mathbf{r}_{j'}) + A(\mathbf{C}_l) + B(m_l, \mathbf{r}_l, \mathbf{r}_{j'}). \tag{22}$$

The change of the energy after swapping depends only on the masses and centroids:

$$\begin{aligned} &A(\mathbf{C}_{i'}) + A(\mathbf{C}_{j'}) - A(\mathbf{C}_i) - A(\mathbf{C}_j) \\ &\quad = B(m_j, \mathbf{r}_j, \mathbf{r}_{j'}) + B(m_l, \mathbf{r}_l, \mathbf{r}_{j'}) \\ &\qquad - B(m_{i'}, \mathbf{r}_{i'}, \mathbf{r}_i) - B(m_l, \mathbf{r}_l, \mathbf{r}_i) \\ &\quad = m_j\|\mathbf{r}_j - \mathbf{r}_{j'}\|^2 - m_{i'}\|\mathbf{r}_i - \mathbf{r}_{i'}\|^2 \\ &\qquad + m_l(\|\mathbf{r}_l - \mathbf{r}_{j'}\|^2 - \|\mathbf{r}_l - \mathbf{r}_i\|^2). \end{aligned} \tag{23}$$

The swapping is accepted only if the change of energy is less than zero. The masses and centroids for the corresponding new clusters can be updated by:

$$m_{i'} = m_i - m_l, \tag{24}$$

$$m_{j'} = m_j + m_l, \tag{25}$$

$$\mathbf{r}_{i'} = \frac{m_i\mathbf{r}_i - m_l\mathbf{r}_l}{m_i - m_l}, \tag{26}$$

$$\mathbf{r}_{j'} = \frac{m_j\mathbf{r}_j + m_l\mathbf{r}_l}{m_j + m_l}. \tag{27}$$

References

[1] Foteinos P, Liu Y, Chernikov A, Chrisochoides N. An evaluation of tetrahedral mesh generation for non-rigid registration of brain MRI. In: Computational biomechanics for medicine: soft tissues and the musculoskeletal system. 2010. p. 126–37.
[2] Fedorov A, Chrisochoides N. Tetrahedral mesh generation for non-rigid registration of brain MRI: analysis of the requirements and evaluation of solutions. In: Proceedings of the 17th international meshing roundtable. 2008. pp. 55–72.
[3] Zhang Y, Bajaj C, Sohn BS. 3D finite element meshing from imaging data. Comput Methods Appl Mech Engrg 2005;194(48–49):5083–106.
[4] Yerry MA, Shephard MS. Automatic three-dimensional mesh generation by the modified-octree technique. Internat J Numer Methods Engrg 1984;20(11): 1965–90.

[5] Zhang Y, Hughes T, Bajaj C. An automatic 3D mesh generation method for domains with multiple materials. Comput Methods Appl Mech Engrg 2010; 199(5–8):405–15.
[6] Dey TK, Janoos F, Levine JA. Meshing interfaces of multi-label data with Delaunay refinement. Eng Comput 2012;28(1):71–82.
[7] Meyer M, Whitaker R, Kirby RM, Ledergerber C, Pfister H. Particle-based sampling and meshing of surfaces in multimaterial volumes. IEEE Trans Vis Comput Graphics 2008;14(6):1539–46.
[8] Goksel O, Salcudean SE. Image-based variational meshing. IEEE Trans Med Imaging 2011;30(1):11–21.
[9] Modersitzki J. Numerical methods for image registration. Oxford University Press; 2004.
[10] Sotiras A, Davatzikos C, Paragios N. Deformable medical image registration: a survey. IEEE Trans Med Imaging 2013;32(7):1153–90.
[11] Ferrant M, Warfield SK, Guttmann CRG, Mulkern RV, Jolesz FA, Kikinis R. 3D image matching using a finite element based elastic deformation model. In: Medical image computing and computer-assisted intervention, MICCAI. 1999, p. 202–9.
[12] Samani A, Bishop J, Yaffe MJ, Plewes DB. Biomechanical 3-D finite element modeling of the human breast using MRI data. IEEE Trans Med Imaging 2001; 20(4):271–9.
[13] Del Palomar AP, Calvo B, Herrero J, López J, Doblaré M. A finite element model to accurately predict real deformations of the breast. Med Eng Phys 2008; 30(9):1089–97.
[14] Werner R, Ehrhardt J, Schmidt R, Handels H. Patient-specific finite element modeling of respiratory lung motion using 4D CT image data. Med Phys 2009; 36(5):1500–11.
[15] Zhang J, Wang J, Wang X, Feng D. The adaptive FEM elastic model for medical image registration. Phys Med Biol 2014;59(1):97–118.
[16] Hughes T. The finite element method: linear static and dynamic finite element analysis. Courier Dover Publications; 2012.
[17] Yan D, Lévy B, Liu Y, Sun F, Wang W. Isotropic remeshing with fast and exact computation of restricted Voronoi diagram. Comput Graph Forum 2009;28(5): 1445–54.
[18] Boyd SK, Müller R. Smooth surface meshing for automated finite element model generation from 3D image data. J Biomech 2006;39(7):1287–95.
[19] Shewchuk JR. Tetrahedral mesh generation by Delaunay refinement. In: Proceedings of the 14th symposium on computational geometry. 1998. pp. 86–95.
[20] Du Q, Faber V, Gunzburger M. Centroidal Voronoi tessellations: applications and algorithms. SIAM Rev 1999;41(4):637–76.
[21] Lloyd S. Least squares quantization in PCM. IEEE Trans Inform Theory 1982; 28(2):129–37.
[22] Liu Y, Wang W, Lévy B, Sun F, Yan D, Lu L, Yang C. On centroidal Voronoi tessellation—energy smoothness and fast computation. ACM Trans Graph 2009;28(4):101:1–101:17.
[23] Peyre G, Cohen L. Surface segmentation using geodesic centroidal tessellation. In: Proceedings of the 2nd international symposium on 3D data processing, visualization, and transmission. 2004. pp. 995–1002.
[24] Edelsbrunner H, Shah NR. Triangulating topological spaces. In: Proceedings of symposium on computational geometry. 1994. p. 285–92.
[25] Du Q, Gunzburger MD, Ju L. Constrained centroidal Voronoi tessellations for surfaces. SIAM J Sci Comput 2003;24(5):1488–506.
[26] Alliez P, Verdiére ÉC, Devillers O, Isenburg M. Centroidal Voronoi diagrams for isotropic surface remeshing. Graph Models 2005;67(3):204–31.
[27] Rong G, Jin M, Guo X. Hyperbolic centroidal Voronoi tessellation. In: Proceedings of symposium of solid and physical modeling, SPM. 2010. p. 117–26.
[28] Rong G, Jin M, Shuai L, Guo X. Centroidal Voronoi tessellation in universal covering space of manifold surfaces. Comput Aided Geom Design 2011;28(8): 475–96.
[29] Du Q, Wang D. Tetrahedral mesh generation and optimization based on centroidal Voronoi tessellations. Internat J Numer Methods Engrg 2003;56(9): 1355–73.
[30] Yan D, Wang W, Lévy B, Liu Y. Efficient computation of clipped Voronoi diagram for mesh generation. Comput-Aided Des 2013;45(4):843–52.
[31] Alliez P, Cohen-Steiner D, Yvinec M, Desbrun M. Variational tetrahedral meshing. ACM Trans Graph 2005;24(3):617–25.
[32] Valette S, Chassery JM, Prost R. Generic remeshing of 3D triangular meshes with metric-dependent discrete Voronoi diagrams. IEEE Trans Vis Comput Graphics 2008;14(2):369–81.
[33] Rong G, Liu Y, Wang W, Yin X, Gu X, Guo X. GPU-assisted computation of centroidal Voronoi tessellation. IEEE Trans Vis Comput Graphics 2011;17(3): 345–56.
[34] Shuai L, Guo X, Jin M. GPU-based computation of discrete periodic centroidal Voronoi tessellation in hyperbolic space. Comput-Aided Des 2013;45(2): 463–72.
[35] Wojtan C, Thürey N, Gross M, Turk G. Deforming meshes that split and merge. ACM Trans Graph 2009;28(3):76:1–76:10.
[36] Kaufmann P, Wang O, Sorkine-Hornung A, Sorkine-Hornung O, Smolic A, Gross M. Finite element image warping. Comput Graph Forum 2013;32(2): 31–9.
[37] Chentanez N, Alterovitz R, Ritchie D, Cho L, Hauser KK, Goldberg K, Shewchuk JR, O'Brien JF. Interactive simulation of surgical needle insertion and steering. ACM Trans Graph 2009;28(3):88:1–88:10.
[38] Sokolnikoff IS. Mathematical theory of elasticity. Krieger Pub. Co.; 1983.
[39] Vercauteren T, Pennec X, Perchant A, Ayache N. Diffeomorphic demons: efficient non-parametric image registration. NeuroImage 2009;45(1):S61–72.
[40] Garland M, Heckbert PS. Surface simplification using quadric error metrics. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH. 1997. p. 209–16.
[41] Segulja C, Han D, Abdelrahman TS, Brock K, Moseley J. Tetrahedral interpolation for deformable image registration on GPUs. In: Symposium on application accelerators in high performance computing. 2010.
[42] Sederberg TW, Parry SR. Free-form deformation of solid geometric models. ACM SIGGRAPH Comput Graph 1986;20(4):151–60.
[43] Sdika M. A fast nonrigid image registration with constraints on the Jacobian using large scale constrained optimization. IEEE Trans Med Imaging 2008; 27(2):271–81.
[44] CGAL, Computational Geometry Algorithms Library [Online]. Available: http://www.cgal.org.
[45] Segars W, Tsui B. MCAT to XCAT: the evolution of 4-D computerized phantoms for imaging research. Proc IEEE 2009;97(12):1954–68.