Summer 2016

Tissue preserving deformable image registration for 4DCT pulmonary images

Bowen Zhao
University of Iowa

TISSUE PRESERVING DEFORMABLE IMAGE REGISTRATION FOR 4DCT PULMONARY IMAGES

by

Bowen Zhao

A thesis submitted in partial fulfillment of the
requirements for the Master of Science
degree in Electrical and Computer Engineering
in the Graduate College of
The University of Iowa

August 2016

Thesis Supervisor: Gary Christensen

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

MASTER'S THESIS

This is to certify that the Master's thesis of

Bowen Zhao

has been approved by the Examining Committee for the thesis requirement for the Master of Science degree in Electrical and Computer Engineering at the August 2016 graduation.

Thesis committee: ______________________________
Gary Christensen, Thesis Supervisor

____
Joseph Reinhardt

____
Hans Johnson

ACKNOWLEDGEMENTS

I would like to first of all thank Professor Gary Christensen for his patient guidance and insightful advice. His suggestions and ideas provided me with much inspiration in my work. I also want to express my gratitude toward Professor Joseph Reinhardt and Professor John Bayouth for their support. Our weekly meeting and discussion greatly widened my horizon in related areas and presented me with fresh perspectives regarding my research. To no lesser extent, I want to thank my colleagues and workmates Joo Hyun Song, Sarah Gerard, Yue Pan, Ali Gayhoor, Wei Shao, Kaifang Du and Taylor Patton for their help and friendship. Our extensive discussions about study and life taught me a lot and helped deepen my understanding of my area of research. Also, I want to thank Professor David Stewart, Professor Hans Johnson and Professor Mathews Jacob for their wonderful lectures, which helped me establish necessary academic foundations for my research. Much appreciation goes to Stefan Klein, Marius Staring, Coert Metz and Professor Geoffrey Hugo, all of whom I haven't met in person. But I sincerely appreciate the help they provided to me through email correspondence during my work on this thesis.

In the end, I want to thank my parents and my wife, Sherry, for their selfless and unconditional support throughout the past three years. They helped me through lots of difficulties in my times of need.

This work is supported in part by NIH grant CA166703 and CA166119.

ABSTRACT

This thesis mainly focuses on proposing a 4D (three spatial dimensions plus time) tissue-volume preserving non-rigid image registration algorithm for pulmonary 4D computed tomography (4DCT) data sets to provide relevant information for radiation therapy and to estimate pulmonary ventilation. The sum of squared tissue volume difference (SSTVD) similarity cost takes into account the CT intensity changes of spatially corresponding voxels, which is caused by variations of the fraction of tissue within voxels throughout the respiratory cycle. The proposed 4D SSTVD registration scheme considers the entire dynamic 4D data set simultaneously, using both spatial and temporal information. We employed a uniform 4D cubic B-spline parametrization of the transform and a temporally extended linear elasticity regularization of deformation field to ensure temporal smoothness and thus biological plausibility of estimated deformation. A multi-resolution multi-grid registration framework was used with a limited-memory Broyden Fletcher Goldfarb Shanno (LBFGS) optimizer for rapid convergence rate, robustness against local minima and limited memory consumption. The algorithm was prototyped in Matlab and then fully implemented in C++ in Elastix package based on the Insight Segmentation and Registration Toolkit (ITK). We conducted experiments on 2D+t synthetic images to demonstrate the effectiveness of the proposed method. The 4D SSTVD algorithm was also tested on clinical pulmonary 4DCT data sets in comparison with existing 3D pairwise SSTVD algorithm and 4D sum of squared difference (SSD) algorithm. The mean landmark error and mean landmark irregularity were calculated based on manually annotated landmarks on publicly available 4DCT data sets to evaluate the accuracy and temporal smoothness of the registration results. A 4D landmarking software tool was also designed and implemented in Java as an ImageJ plug-in to help facilitate the landmark labeling process in 4DCT data sets.

PUBLIC ABSTRACT

Four dimensional (three spatial dimensions plus time) computed tomography (4DCT) image of the lung can provide numerous valuable information. For example, we can study the structures of vascular system, airway system, fissures, and tumors of the lung using image segmentation techniques. Also, since the 4DCT image contains a series of 3DCT lung images representing the shapes of the lung at different breathing phases within the respiratory cycle, we can use image registration technique to estimate how the lung changes its shape from one breathing phase to another within the breathing cycle. In addition to estimating the transformation of the lung during breathing, we can obtain information about local ventilation behavior (the amount of expansion or contraction) by calculating the Jacobian of the estimated transform. The assessment of local ventilation would in turn provide important guidance for radiation therapy.

The work of this thesis is focused on image registration of 4DCT lung images. The aim, of course, is to estimate accurate and biologically meaningful transforms of the lung among different breathing phases, where "biologically meaningful" means the estimated transforms should be relatively smooth and no folding of lung tissue should result from the transforms. We need to take into account the characteristic of CT image, that is, the image intensity (CT number) is proportional to the density of the material being imaged. As the lung ventilates, air flows in and out, making the density of the lung vary periodically. Therefore, CT image intensities of spatially corresponding point within the lung will also vary periodically, and the 4DCT lung image represents one period of the infinite cyclic motion.

In this thesis, we present a new registration method to deal with 4DCT lung image registration. This method takes into account the intensity variations of spatially corresponding points and considers both spatial and temporal information within the 4DCT lung image simultaneously. Experiments on synthetic and clinical 4DCT images show that the proposed algorithm strikes a good balance between registration accuracy and smoothness of transformation over time, both of which were measured using manually annotated landmarks. A software tool was also developed to help experts manually label landmarks in 4DCT images.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

In pulmonary radiation therapy, target localization and motion tracking of parenchyma, as well as tumor regions, are critically important for guiding radiation on pathology, minimizing dose for normal tissue and estimating regional lung ventilation[13]. Time-resolved dynamic imaging data sets have become increasingly available with the advancement of medical imaging techniques[9], making it feasible to track and model pulmonary motion. Pulmonary 4DCT allows for reconstruction of multiple 3D volumes corresponding to different breathing phases throughout the respiratory cycle. Non-rigid image registration is used to estimate and track correspondences between these phase images.

The CT number of lung parenchyma varies across the breathing phases due to changes in fraction of tissue within voxels, which is caused by air flowing in and out of the lung throughout the respiratory cycle. Non-rigid registration of 4DCT pulmonary images is a difficult problem due to the fact that spatially corresponding points belonging to different breathing phases have dissimilar image intensities because of lung ventilation. To compensate for these intensity variations, and to enforce the principle of tissue conservation, Yin et al.[20] and Gorbunova et al.[6] independently proposed an intensity-based registration similarity cost function that accounts for these intensity changes across breathing phases. We will refer to this previously established cost function as the sum of squared tissue volume differences (SSTVD). The SSTVD cost function achieves desirable registration results for pulmonary CT image alignment compared to the conventional sum of squared (intensity) difference (SSD) cost function [20],[6] which does not account for intensity changes. In this paper, we extended the SSTVD registration framework from 3D to 4D.

4D registration techniques have become more common in recent years[11][19]. Unlike traditional registration schemes which perform a sequence of 3D pair-wise registrations, either with respect to a common reference or between consecutive time point images, 4D registration frameworks consider the information contained within the entire time-resolved imaging data set simultaneously without bias toward any chosen reference image. One advantage of 4D image registration is that it avoids accumulation of numerical and discretization errors associated with 3D sequential registration. Additionally, 4D registration frameworks provide smooth and consistent displacement fields along spatial and temporal dimensions[16]. Thus, 4D registration would provide more biologically reasonable tissue motion tracking than 3D pair-wise registration. This could be used to improve radiation therapy plans and dose delivery during breathing [10].

The advantage of one registration algorithm over another can only be quantitatively shown through certain evaluation measures. One of the most effective means for assessing the performance of registration algorithms is anatomical landmarks. The average distance between locations of true landmarks on a certain breathing phase image and those of deformed landmarks on the same phase resulting from estimated transforms among the phases serves as a good measurement for registration accuracy. Meanwhile, the average magnitude of acceleration the deformed landmarks retain as they traverse through the 4D data set along their estimated trajectories is a meaningful measurement of the temporal smoothness of the transform estimated from registration.

The importance of anatomical landmarks for registration evaluation is acknowledged but the difficulty of their acquisition remains. In the case of pulmonary 4DCT data sets, branch points of vessels are among the most common types of landmarks to be labeled. However, because of lung motion within the respiratory cycle, locations for corresponding landmarks will be constantly changing throughout the 4D data set. Also, the fact that the observer needs to go through the CT image voxel by voxel during the landmarking task makes him/her frequently miss the forest for the trees, and "get lost" in the image volume.

In this thesis, we propose a 4D tissue preserving registration algorithm (4D SSTVD) which incorporates the merits of SSTVD registration similarity cost with a 4D transformation model. The resulting registration framework is more suitable for 4DCT pulmonary image registration than either pairwise 3D SSTVD or 4D SSD [11] registration. Then, we demonstrate the concepts, design and usage of a 4D landmarking software, which is able to aid the user to label landmarks accurately and efficiently on 4DCT data sets. And the 4D landmarks can in turn serve as evaluation measures for registration results.

CHAPTER 2
FOUR DIMENSIONAL TISSUE PRESERVATION NON-RIGID IMAGE REGISTRATION FRAMEWORK

In essence, image registration is a large scale numerical optimization problem with specialized objective functions. The scale is large in the sense that the domain of the problem could be of enormously high dimension, and every evaluation of the objective function as well as its derivative could require the involvement of large number of samples, costing considerable amount of computation time.

The imperative task of image registration is to estimate a geometric transform $\mathbf{T}$, parametric or non-parametric, that deforms a "moving image" I_m to match with a "fixed image" I_f. How well the transform achieves this goal is in turn measured by an objective or cost function in the form of Equ. 2.1:

$$Cost(\mathbf{T}) = Dissimilarity(\mathbf{T}) + \lambda \cdot Regularization(\mathbf{T}) \tag{2.1}$$

where the dissimilarity term measures how much the images to be registered are out of alignment, while the regularization term measures the smoothness and/or magnitude of the estimated transform, preventing over-fitting and maintaining certain topological properties of the images during the optimization process. We would expect a successful registration process to yield a transform $\mathbf{T}$ that minimizes the dissimilarity between the moving and the fixed images and retain acceptable regularity at the same time. If we employ parameterization for the transform, then the parameter space is the domain (or search space) of this optimization problem.

In general, the image registration framework can be modularized and demon-

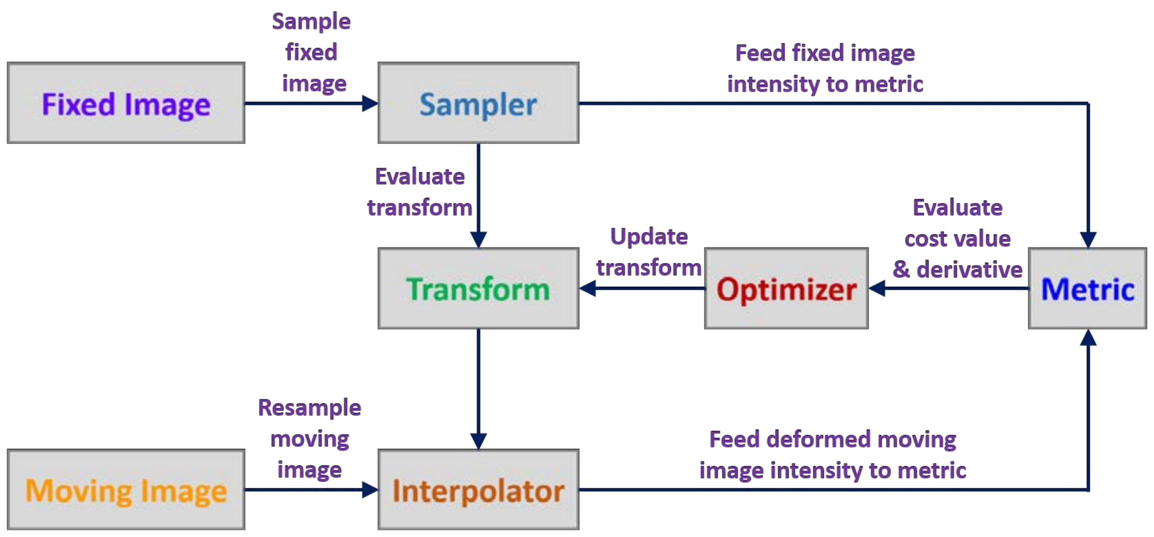


Figure 2.1: Modularized image registration framework.

strated as in Fig. 2.1. The sampler for fixed image is often used when we are dealing with images containing huge number of voxels, e.g., 4DCT data sets, where evaluating the cost function value and derivative based on all image voxels at every iteration could be computationally prohibitive. The 4D tissue volume preserving image registration algorithm proposed in this thesis mainly focuses on the metric module.

2.1 Methods

Our proposed 4D SSTVD image registration method minimizes a 4D SSTVD registration cost function that consists of an SSTVD similarity cost and a temporally extended linear elastic regularization cost. In this paper, we denote the generic spatio-temporal coordinate in the target image domain as $\mathbf{x} = (x_1, x_2, x_3, x_4)^T \equiv (x, y, z, t)^T$, where x_1, x_2, x_3, x_4 will be used interchangeably with x, y, z, t for notational convenience. And the generic spatio-temporal coordinate in the moving image domain is $\mathbf{y} = (y_1, y_2, y_3, y_4)^T$. The concept of ***time*** is identified with that of ***breathing phases*** for the 4DCT image, indicating that the 3D image at ***time point t*** is the same as the ***phase t*** 3D image. Also, the term ***fixed image*** will be used interchangeably with ***target image*** in the registration framework.

2.1.1 Review of SSTVD Similarity Cost Function

The underlying assumption behind SSTVD similarity cost is that the total tissue volume will remain roughly the same throughout the respiratory cycle while the total lung volume will change as air flows in and out of the lung. Let $I_f^{HU} : \Omega_f \to \mathbb{R}, \mathbf{x} \mapsto I_f^{HU}(\mathbf{x})$ denote the fixed image in Hounsfield units and $I_m^{HU} : \Omega_m \to \mathbb{R}, \mathbf{y} \mapsto I_m^{HU}(\mathbf{y})$ the moving image in Hounsfield units. Following the rationale presented in the work of Yin et al.[20], an intensity linear transformation is performed on these images and consequently the intensity value at each voxel is converted from radiodensity in Hounsfield units to actual tissue volume in mm^3. Specifically, we have:

$$\begin{aligned} I_f(\mathbf{x}) &= v_f \cdot r_f(\mathbf{x}) = v_0 \cdot r_f(\mathbf{x}) \\ I_m(\mathbf{y}) &= v_m \cdot r_m(\mathbf{y}) = v_0 \cdot r_m(\mathbf{y}) \end{aligned} \tag{2.2}$$

where $HU_{tissue} = 55$ and $HU_{air} = -1000$ are the radiodensity values of tissue and air in Hounsfield units. $r_f(\mathbf{x})$ and $r_m(\mathbf{y})$ with

$$\begin{aligned} r_f(\mathbf{x}) &= \frac{I_f^{HU}(\mathbf{x}) - HU_{air}}{HU_{tissue} - HU_{air}} = \frac{I_f^{HU}(\mathbf{x}) + 1000}{1055} \\ r_m(\mathbf{y}) &= \frac{I_m^{HU}(\mathbf{y}) - HU_{air}}{HU_{tissue} - HU_{air}} = \frac{I_m^{HU}(\mathbf{x}) + 1000}{1055} \end{aligned} \tag{2.3}$$

represent the fraction of tissue within a standard image voxel centered at $\mathbf{x}$ and $\mathbf{y}$ for the fixed and moving image, respectively. v_f and v_m are the volumes of a standard voxel in the fixed and moving image, respectively. For images in the same 4DCT data set, we usually have $v_f = v_m = v_0$, i.e., standard voxels have constant volume in both the fixed and moving image, which can be calculated from physical spacing of the data set. The resulting images $I_f : \Omega_f \to \mathbb{R}$ and $I_m : \Omega_m \to \mathbb{R}$ are named tissue-volume images and it is these two images that we are trying to register in the SSTVD registration framework.

Let $\mathbf{T} : \Omega_f \to \Omega_m, \mathbf{x} \mapsto \mathbf{T}(\mathbf{x})$ be the geometric transform deforming I_m to match I_f. The SSTVD similarity cost is thus given by:

$$\begin{aligned} C(\mathbf{T}) &= \frac{1}{|\Omega_f|} \sum_{\mathbf{x}\in\Omega_f} \Big(v_0 \cdot r_f(\mathbf{x}) - (|J_{\mathbf{T}}|(\mathbf{x}) \cdot v_0) \cdot r_m(\mathbf{T}(\mathbf{x})) \Big)^2 \\ &= \frac{1}{|\Omega_f|} \sum_{\mathbf{x}\in\Omega_f} \Big(I_f(\mathbf{x}) - |J_{\mathbf{T}}|(\mathbf{x}) \cdot I_m(\mathbf{T}(\mathbf{x})) \Big)^2 \end{aligned} \tag{2.4}$$

where $v_0 \cdot r_f(\mathbf{x})$ is the volume of tissue within a standard voxel centered at $\mathbf{x}$ in the fixed tissue-volume image I_f, while $|J_{\mathbf{T}}|(\mathbf{x}) \cdot v_0 \cdot r_m(\mathbf{T}(\mathbf{x}))$ is the volume of tissue within a standard voxel centered at $\mathbf{x}$ in the deformed moving tissue-volume image. The spatial Jacobian of transform $|J_{\mathbf{T}}|(\mathbf{x})$ is introduced because the mathematical interpretation of transform Jacobian $|J_{\mathbf{T}}|(\mathbf{x})$ is the contraction or expansion of an infinitesimal local neighborhood of $\mathbf{x}$ induced by $\mathbf{T}$. Let $\mathbf{R}$ denote the region occupied by the standard voxel centered at $\mathbf{x}$ in the fixed image domain, whose volume is v_0. Then $\mathbf{T}(\mathbf{R}) \ni \mathbf{T}(\mathbf{x})$ would be the transformed region occupied by the "deformed voxel" in the moving image domain, whose volume can be approximated by $|J_{\mathbf{T}}|(\mathbf{x}) \cdot v_0$, as illustrated in

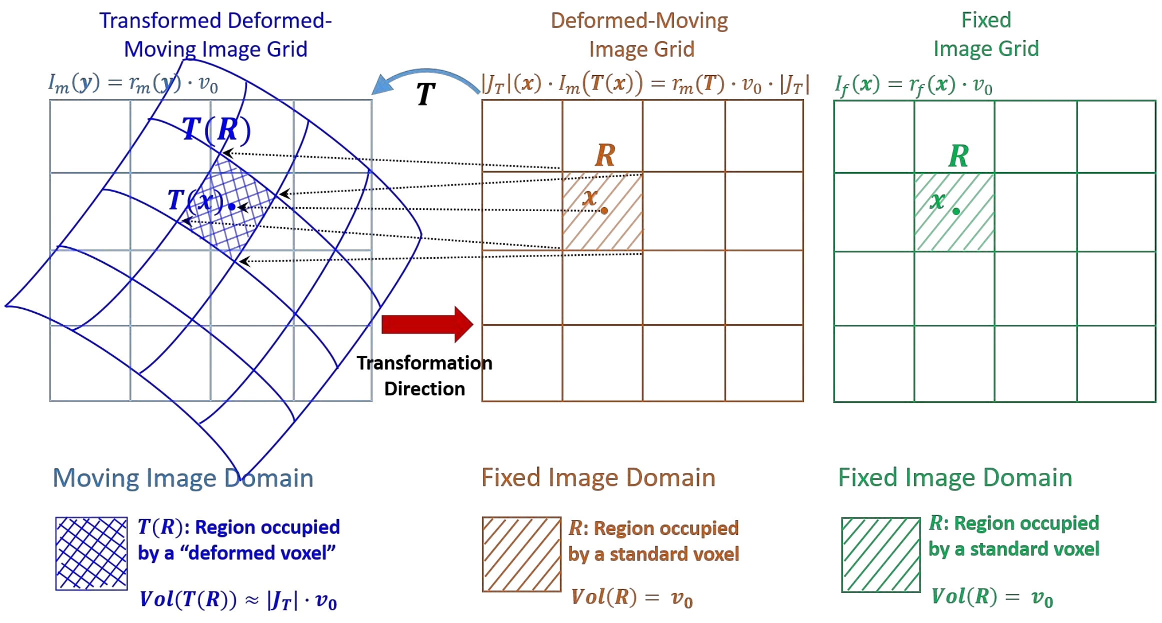

Transformed Deformed-Moving Image Grid
$I_m(y) = r_m(y) \cdot v_0$
$T(R)$
$T(x)$
T
Deformed-Moving Image Grid
$|J_T|(x) \cdot I_m(T(x)) = r_m(T) \cdot v_0 \cdot |J_T|$
R
x
Fixed Image Grid
$I_f(x) = r_f(x) \cdot v_0$
R
x
Transformation Direction
Moving Image Domain
Fixed Image Domain
Fixed Image Domain
$T(R)$: Region occupied by a "deformed voxel"
$Vol(T(R)) \approx |J_T| \cdot v_0$
R: Region occupied by a standard voxel
$Vol(R) = v_0$
R: Region occupied by a standard voxel
$Vol(R) = v_0$

Figure 2.2: Illustration of SSTVD cost and the "deformed voxel".

Figure 2.2. In the SSTVD registration scheme, the term $|J_{\mathbf{T}}|(\mathbf{x}) \cdot I_m(\mathbf{T}(\mathbf{x}))$ as a whole serves as the deformed moving image, instead of $I_m(\mathbf{T}(\mathbf{x}))$ alone.

Compared to conventional SSD similarity cost function, which is given as below (by abuse of notation for I_f and I_m):

$$C_{SSD}(\mathbf{T}) = \frac{1}{|\Omega_f|} \sum_{\mathbf{x} \in \Omega_f} \Big(I_f(\mathbf{x}) - I_m(\mathbf{T}(\mathbf{x})) \Big)^2 \tag{2.5}$$

the additional Jacobian modification $|J_{\mathbf{T}}|(\mathbf{x})$ will help compensate for the intensity changes of spatially corresponding voxels in a 4DCT image data set, which is caused by periodic variations in the fraction of tissue within voxels during ventilation.

2.1.2 General Concept of 4D Registration Framework

Before we talk about specific aspects of the 4D SSTVD registration framework, we shall briefly introduce the general concepts of the 4D registration framework. The one key difference between 4D and 3D image registration is that 4D registration uses the information contained within the entire 4D data set. Specifically, the 4D image itself will be treated as the moving image while the target image is not explicitly given. In our proposed algorithm, the temporal average of the 4D deformed moving tissue-volume image will be used as the target image. In essence, the algorithm is trying to minimize the temporal variance of the 4D deformed moving tissue-volume image. Conceptually, we can think of the target image also as a 4D image which the temporal duplication of the 3D temporal average image. And the estimated 4D transform will deform the 4D moving image to match the conceptual 4D target image. Based on this setup, at the beginning of the algorithm when the 4D transform is just identity, the target image is the temporal mean of the original 4D image. As the algorithm progresses, the target image is actually not stationary. It will be iteratively updated and gradually stabilize toward a previously unknown average shape as the estimated 4D transform converges toward an optimum. The estimated 4D transform, deforming the original 4D image into the average shape will be called the "forward transform". While the forward transform is acquired, we would expect to have achieved a relatively stable target image, representing the temporal mean of deformed moving tissue-volume image.

In practice, it is usually the 3D spatial transforms between pairs of specific breathing phase images that we are interested in. Therefore, we will also need to estimate a 4D “inverse transform” deforming the previously obtained target image into the original 4D moving image. After we estimated both the forward transform and its corresponding inverse transform, we can easily obtain 3D spatial transforms between any two phase images by composing the above two 4D transforms.

The whole process can be illustrated in Fig 2.3-2.7, in which $\mathbf{T}$ and $\mathbf{S}$ represent the forward and inverse 4D transforms. $\mathbf{x}$ and $\mathbf{y}$ represent generic 4D coordinate vectors in the target and moving image domain. $\mathbf{x}_s$ and $\mathbf{y}_s$ denote the corresponding 3D spatial coordinate vectors in the target and moving image domain. Here, we are just pictorially demonstrating the general concepts about 4D registration. All these terms will be more rigorously defined in next few subsections.

2.1.3 4D SSTVD Similarity Cost Function

Inspired by the work of Yin et al.[20], Gorbunova et al.[6] and Metz et al.[11], the proposed 4D SSTVD similarity cost function takes advantage of the fact that the tissue volume can be roughly considered as constant [20] throughout the respiratory cycle and it uses the spatial and temporal information within the 4D data set simultaneously.

Before proceeding to the actual formulation of 4D SSTVD cost, we shall define some notations. Define the 4DCT image domain as $\Omega = \Omega_s \times \Omega_t$, where $\Omega_s = \Omega_x \times \Omega_y \times \Omega_z$ is the spatial domain and $\Omega_t = \{0, 1, ..., |\Omega_t| - 1\}$ is the temporal domain. Also, let $\mathbf{x}_s = (x, y, z)^T \in \Omega_s$ be the vector of spatial coordinates, $t \in \Omega_t$

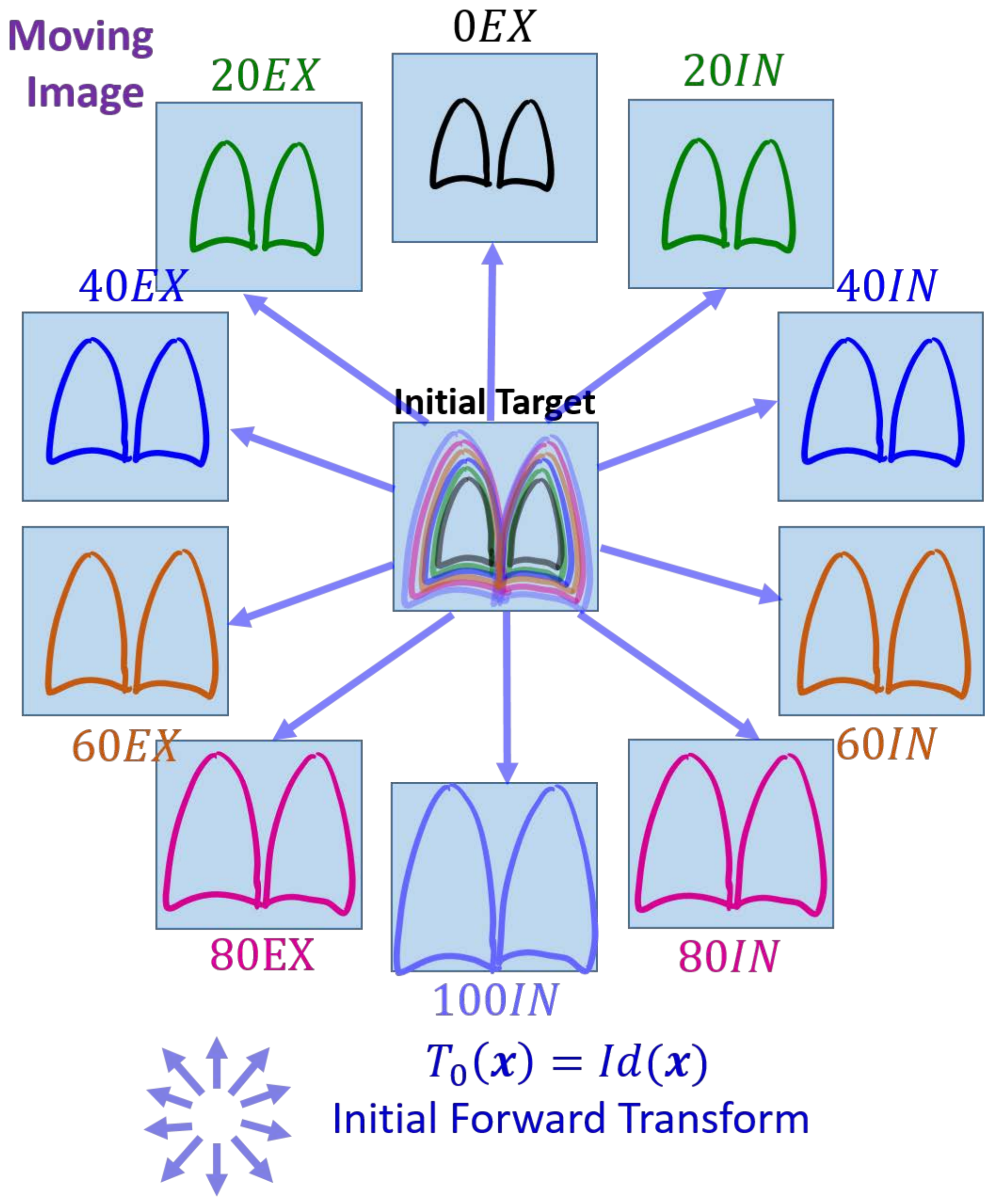


Figure 2.3: 4D image registration concepts: 1. At the beginning of the 4D registration, the initial transform is identity and the target image is just the temporal average of all phases in the 4D moving image. So the target is blurry.

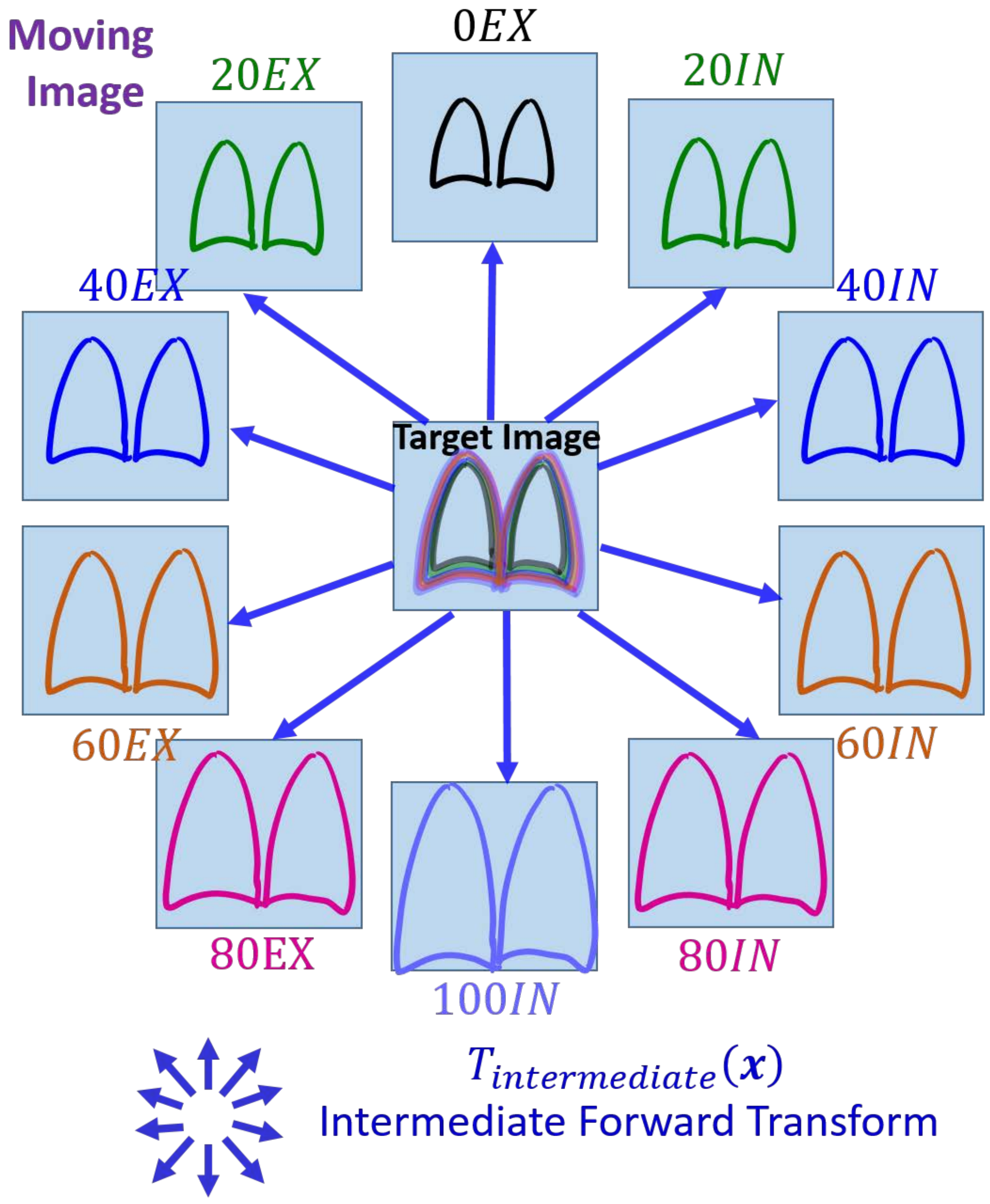

Moving Image
0EX
20EX
20IN
40EX
40IN
Target Image
60EX
60IN
80EX
80IN
100IN
$T_{intermediate}(\boldsymbol{x})$
Intermediate Forward Transform

Figure 2.4: 4D image registration concepts: 2. During the 4D registration process, the temporal variance of the 4D deformed moving image is being minimized. So the target image, being the temporal mean of the 4D deformed moving image, is getting sharper.

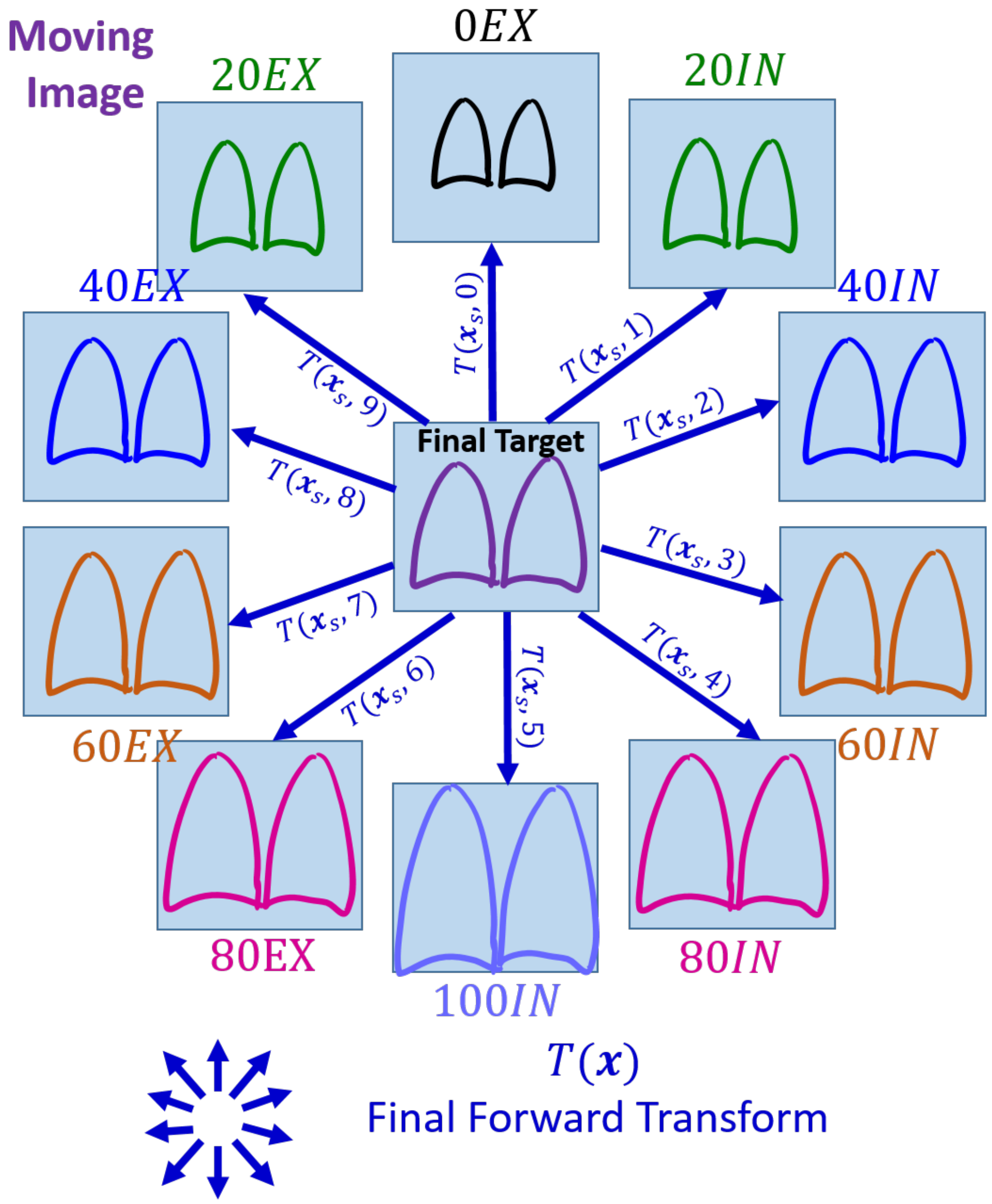


Figure 2.5: 4D image registration concepts: 3. After the 4D registration stops, the temporal variance of the 4D deformed moving image has been reduced to small value, and the target image has become sharp and clear. The 4D forward transform deforming the 4D moving image to the target has been estimated.

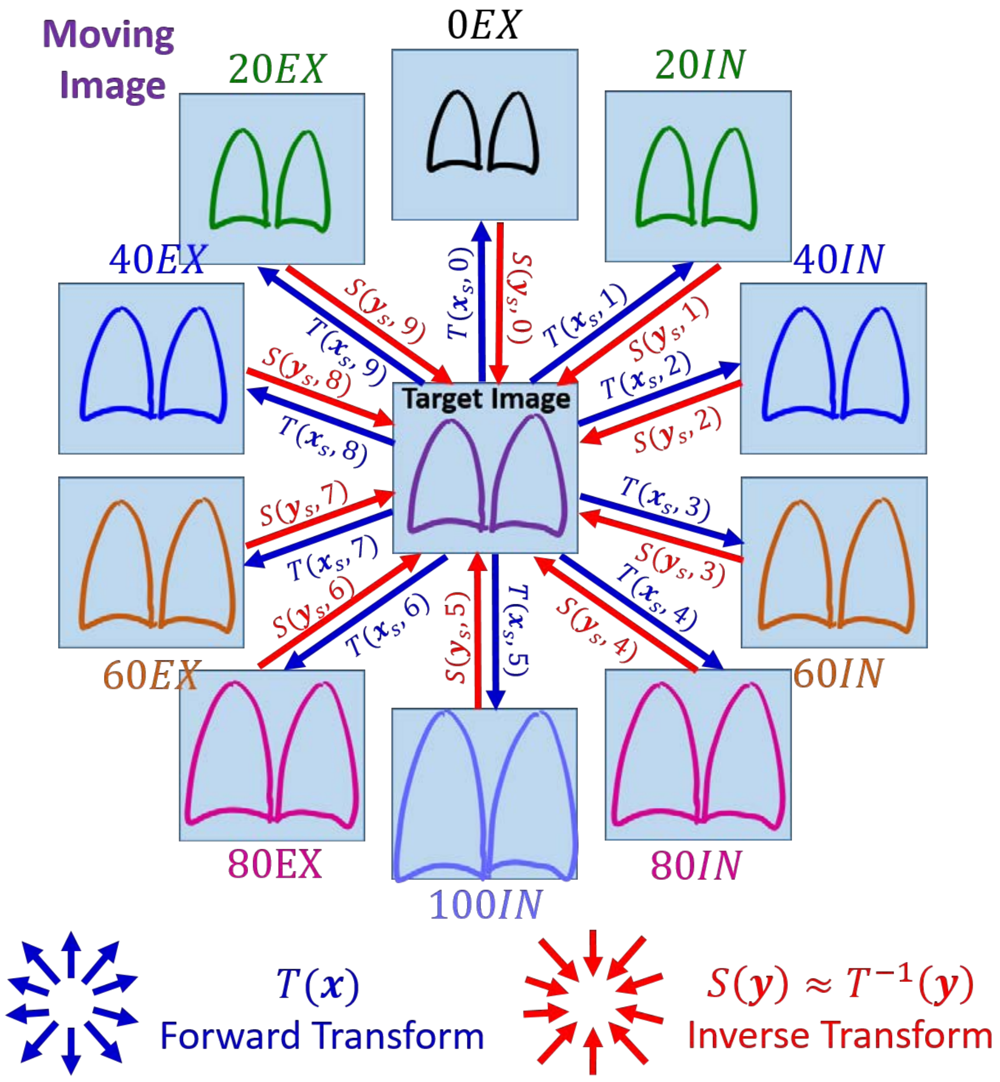


Figure 2.6: 4D image registration concepts: 4. Based on the 4D forward transform, the 4D inverse transform is estimated by minimizing the displacement of any point in the target image domain after that point gets mapped consecutively by the forward and the inverse transforms to come back to the target image domain.

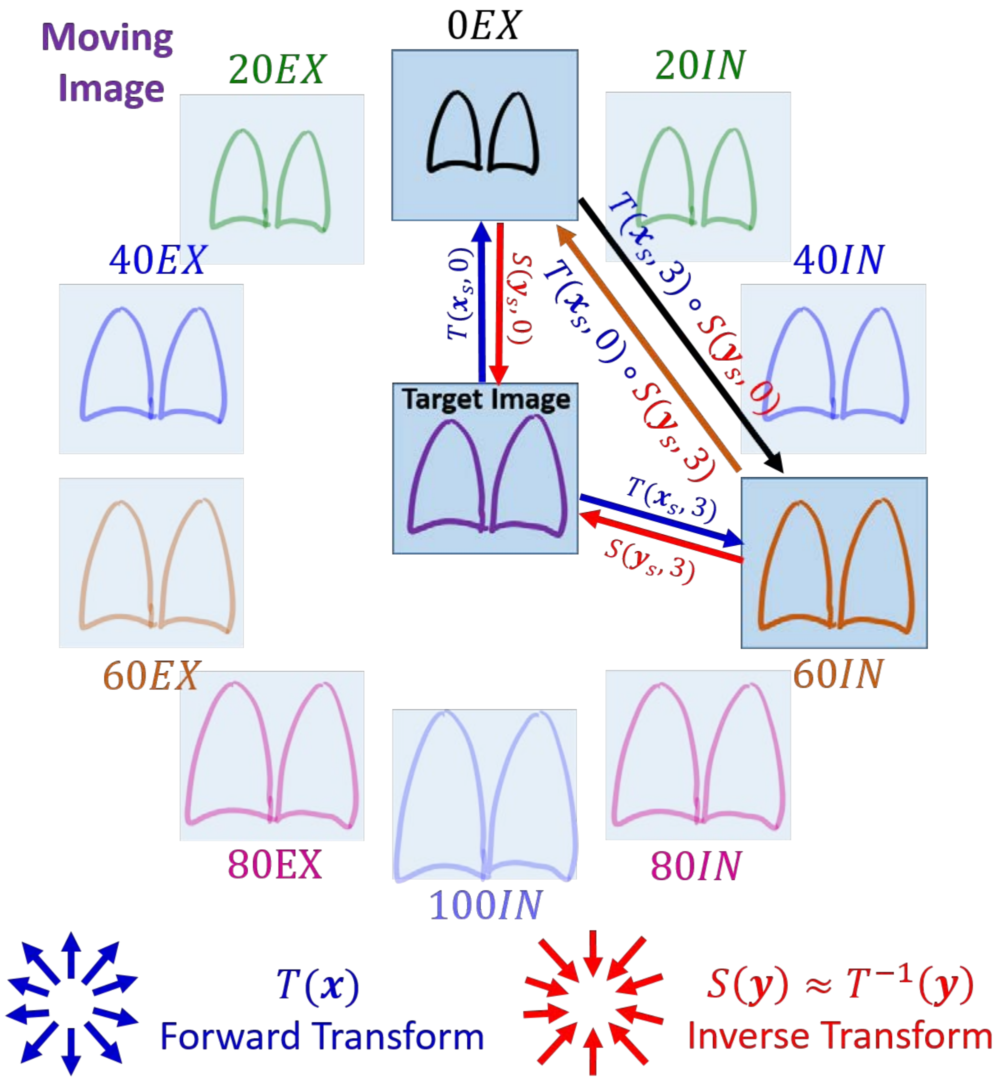


Figure 2.7: 4D image registration concepts: 5. Once both the forward and the inverse 4D transforms have been estimated, we can compose them to acquire the 3D spatial transforms between any two 3D phase images.

the time/phase coordinate, and thus $\mathbf{x} = (\mathbf{x}_s^T, t)^T \in \Omega$ represents the generic spatio-temporal coordinates of a point in the 4DCT image domain. Denote $\mathbf{T} : \Omega_s \times \Omega_t \to \Omega_s \times \Omega_t$ as the transformation from fixed/target image domain to moving image domain with the constraint that $\mathbf{T}(\mathbf{x}_s, t) = (\phi(\mathbf{x}_s)^T, t)^T$ where $\phi : \Omega_s \to \Omega_s$, i.e., no temporal displacement should occur. This is to ensure that any voxel within a certain phase image will remain in that phase without being temporally deformed and moving forward or backward in time. Finally, let $|J_{\mathbf{T}}|(\mathbf{x})$ be the local Jacobian of transformation $\mathbf{T}$ evaluated at $\mathbf{x}$.

The original 4DCT image $I^{HU} : \Omega \to \mathbb{R}$ underwent the following linear transformation of intensity and was converted into 4D tissue-volume image $I : \Omega \to \mathbb{R}$ (the resulting tissue-volume image was windowed to get rid of intensity values falling outside of the range $[0, 1]$).

$$I(\mathbf{x}) = v_0 \cdot \frac{I^{HU}(\mathbf{x}) - HU_{air}}{HU_{tissue} - HU_{air}} = v_0 \cdot \frac{I^{HU}(\mathbf{x}) + 1000}{1055} \tag{2.6}$$

The proposed 4D SSTVD intensity cost function uses the 4D tissue-volume image as the moving image and temporal average of deformed moving tissue-volume image as the target image. Under this construction, the target image is given implicitly and will be iteratively updated with the optimization process until it converges to a relatively stable state. Note that at the beginning of the algorithm, the transformation is identity and the transform Jacobian is 1 everywhere, so the initial target image is just the temporal average of the entire 4D tissue-volume image.

$$C_{int}(\mathbf{T}) = \frac{1}{|\Omega_s|} \frac{1}{|\Omega_t|} \sum_{\mathbf{x}_s \in \Omega_s} \sum_{t \in \Omega_t} \left(\bar{K}(\mathbf{x}_s) - K(\mathbf{x}) \right)^2 \tag{2.7}$$

where

$$K(\mathbf{x}) = |J_{\mathbf{T}}|(\mathbf{x}) \cdot I(\mathbf{T}(\mathbf{x})) \tag{2.8}$$

is deformed moving tissue-volume image

$$\begin{aligned}\bar{K}(\mathbf{x}_s) &= \frac{1}{|\Omega_t|} \sum_{\tau \in \Omega_t} K(\mathbf{x}_s, \tau) \\ &= \frac{1}{|\Omega_t|} \sum_{\tau \in \Omega_t} \Big(|J_{\mathbf{T}}|(\mathbf{x}_s, \tau) \cdot I(\mathbf{T}(\mathbf{x}_s, \tau)) \Big) \end{aligned} \tag{2.9}$$

is temporal average of deformed moving tissue-volume image and serves as target image in the registration framework.

2.1.4 4D Transformation Model

The 4D transformation $\mathbf{T}$ is parameterized using a tensor product of four 1D cubic B-spline[14] kernels and is given by Equation 2.10. The notation $\mathbf{T}(\mathbf{x}, \mathbf{a})$ is used to emphasize the dependence of transformation $\mathbf{T}$ on spatio-temporal coordinates $\mathbf{x}$, as well as the parameters $\mathbf{a}$ used to parameterize the transformation. Similarly, by abuse of notation, the Jacobian of transform, the deformed moving tissue-volume image, and the temporal average of deformed moving tissue-volume image (the target image) will be denoted as $J_{\mathbf{T}}(\mathbf{x}, \mathbf{a})$, $K(\mathbf{x}, \mathbf{a})$ and $\bar{K}(\mathbf{x}_s, \mathbf{a})$.

$$\begin{aligned}
\mathbf{T}(\mathbf{x},\mathbf{a}) &= \mathbf{x}+\mathbf{u}(\mathbf{x},\mathbf{a}) \\
&= \mathbf{x}+\sum_{i=-1}^{N_x-2}\sum_{j=-1}^{N_y-2}\sum_{k=-1}^{N_z-2}\sum_{l=-1}^{N_t-2}\mathbf{a}_{i,j,k,l}B(\frac{x}{\delta_x}-i)B(\frac{y}{\delta_y}-j)B(\frac{z}{\delta_z}-k)B(\frac{t}{\delta_t}-l) \\
&= \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix} + \sum_{i=\lfloor\frac{x}{\delta_x}\rfloor-1}^{\lfloor\frac{x}{\delta_x}\rfloor+2}\sum_{j=\lfloor\frac{y}{\delta_y}\rfloor-1}^{\lfloor\frac{y}{\delta_y}\rfloor+2}\sum_{k=\lfloor\frac{z}{\delta_z}\rfloor-1}^{\lfloor\frac{z}{\delta_z}\rfloor+2}\sum_{l=\lfloor\frac{t}{\delta_t}\rfloor-1}^{\lfloor\frac{t}{\delta_t}\rfloor+2}\begin{bmatrix} a_{i,j,k,l,1} \\ a_{i,j,k,l,2} \\ a_{i,j,k,l,3} \\ 0 \end{bmatrix}\cdot \\
&\qquad B(\frac{x}{\delta_x}-i)B(\frac{y}{\delta_y}-j)B(\frac{z}{\delta_z}-k)B(\frac{t}{\delta_t}-l) \qquad (2.10)
\end{aligned}$$

In Equ. 2.10, $\mathbf{u}(\mathbf{x},\mathbf{a})$ is the displacement vector dependent on spatio-temporal coordinates $\mathbf{x}$ and B-spline coefficients $\mathbf{a}$. $\mathbf{a}_{i,j,k,l} \in \mathbb{R}^3 \times \{0\}$ is the B-spline coefficient vector at B-spline grid point $(i,j,k,l) \in \mathbb{Z}^4$. Note that setting the last coordinate of all four dimensional B-spline coefficient vectors $\mathbf{a}_{i,j,k,l}$ to 0 effectively prevents any temporal transformation. The variables N_α and δ_α for $\alpha \in \{x,y,z,t\}$ represent the number of grid points and grid spacing in the x, y, z, and t directions, respectively. The expressions of cubic B-spline kernel as well as its first and second order derivatives are given in Table 2.1. Their plots are shown in Figure 2.8. Even though we forbade the existence of any temporal displacement, we used the 4D B-spline kernel to ensure temporal smoothness of all spatial displacement components.

2.1.5 Temporally Extended Linear Elasticity Regularization

A temporally extended linear elastic model was used to regularize the 4D SSTVD image registration algorithm. The conventional spatial linear elasticity regularization is valid for small elastic deformations between a pair of phase images within

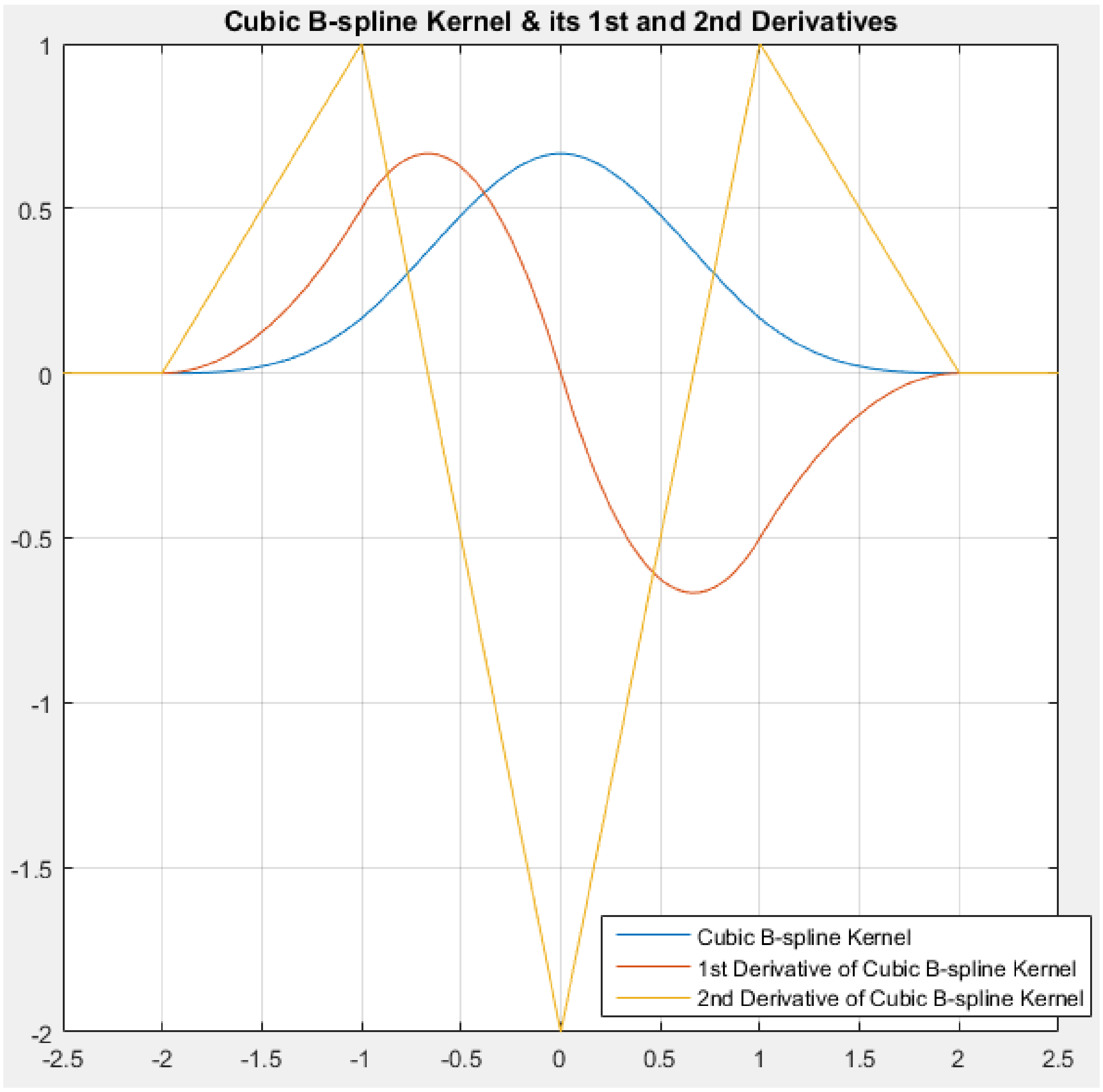


Figure 2.8: Plots of cubic B-spline kernel and its 1st and 2nd order derivatives.

	$B(x)$	$B'(x)$	$B''(x)$
$-2 \leq x \leq -1$	$(x+2)^3/6$	$(x+2)^2/2$	$x+2$
$-1 \leq x \leq 0$	$(-3x^3-6x^2+4)/6$	$(-3x^2-4x)/2$	$-3x-2$
$0 \leq x \leq 1$	$(3x^3-6x^2+4)/6$	$(3x^2-4x)/2$	$3x-2$
$1 \leq x \leq 2$	$-(x-2)^3/6$	$-(x-2)^2/2$	$-x+2$
$\|x\| > 2$	0	0	0

Table 2.1: Expressions of cubic B-spline kernel

and its 1st and 2nd order derivatives.

the breathing cycle[4]. Based on the spatial linear elastic model, we imposed an additional temporal smoothness regularization for spatial displacement components and formulated the temporally extended linear elasticity regularization cost as follows:

$$C_{reg}(\mathbf{a}) = \frac{1}{|\Omega|} \sum_{\mathbf{x} \in \Omega} ||(L\mathbf{u})(\mathbf{x}, \mathbf{a})||^2 \tag{2.11}$$

where the linear elasticity differential operator L is given by (omitting arguments $\mathbf{x}$ and $\mathbf{a}$)

$$\begin{aligned} L\mathbf{u} &= c_1(\nabla \cdot \nabla)\mathbf{u} + c_2\nabla(\nabla \cdot \mathbf{u}) + c_3\mathbf{u} \\ &= (L_1, L_2, L_3, L_4)^T \end{aligned} \tag{2.12}$$

where

$$\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \frac{\partial}{\partial t})^T$$

is the gradient operator. And the regularization components L_i's are given by

$$\begin{aligned} L_1 &= c_1(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_x}{\partial y^2} + \frac{\partial^2 u_x}{\partial z^2} + \frac{\partial^2 u_x}{\partial t^2}) \\ &+ c_2(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_y}{\partial x \partial y} + \frac{\partial^2 u_z}{\partial x \partial z}) + c_3 u_x \end{aligned} \tag{2.13}$$

$$\begin{aligned} L_2 &= c_1(\frac{\partial^2 u_y}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial^2 u_y}{\partial z^2} + \frac{\partial^2 u_y}{\partial t^2}) \\ &+ c_2(\frac{\partial^2 u_x}{\partial y \partial x} + \frac{\partial^2 u_y}{\partial y^2} + \frac{\partial^2 u_z}{\partial y \partial z}) + c_3 u_y \end{aligned} \tag{2.14}$$

$$\begin{aligned} L_3 &= c_1(\frac{\partial^2 u_z}{\partial x^2} + \frac{\partial^2 u_z}{\partial y^2} + \frac{\partial^2 u_z}{\partial z^2} + \frac{\partial^2 u_z}{\partial t^2}) \\ &+ c_2(\frac{\partial^2 u_x}{\partial z \partial x} + \frac{\partial^2 u_y}{\partial z \partial y} + \frac{\partial^2 u_z}{\partial z^2}) + c_3 u_z \end{aligned} \tag{2.15}$$

$$L_4 = c_2(\frac{\partial^2 u_x}{\partial t \partial x} + \frac{\partial^2 u_y}{\partial t \partial y} + \frac{\partial^2 u_z}{\partial t \partial z}) \tag{2.16}$$

The last regularization component L_4 only contains mixed second order partial derivative terms because temporal displacement u_t has been constrained to 0. The constants c_1, c_2 and c_3 are the weights that adjust the elasticity of the model. In our experiments, we chose $c_1 = 0.75$, $c_2 = 0.25$ and $c_3 = 0$. Substituting Equations. 2.13-2.15 into Eq. 2.11 gives

$$C_{reg}(\mathbf{a}) = \frac{1}{|\Omega|} \sum_{\mathbf{x} \in \Omega} ||(L\mathbf{u})(\mathbf{x}, \mathbf{a})||^2 = \frac{1}{|\Omega|} \sum_{\mathbf{x} \in \Omega} \sum_{i=1}^{4} L_i^2(\mathbf{x}, \mathbf{a}) \tag{2.17}$$

2.1.6 Optimization Strategy

The 4D SSTVD registration algorithm is implemented by finding the transformation parameters that minimize a linear combination of the intensity similarity cost and the linear elasticity penalty term

$$C_{total}(\mathbf{a}) = \lambda C_{int}(\mathbf{a}) + C_{reg}(\mathbf{a}) \tag{2.18}$$

where λ is a weight to be tuned. For the experiments in this paper, we chose $\lambda = 80$.

Minimizing the cost function necessitates acquiring its derivatives with respect to the transform parameters. Let $b_\alpha \triangleq a_{p,q,r,s,\alpha}, \alpha \in \{x, y, z\}$, be the α component of the B-spline coefficient vector $\mathbf{a}_{p,q,r,s}$ at grid location $[p, q, r, s] \in \mathbb{Z}^4$. Using the chain rule, product rule, and formulas from matrix calculus, the derivative of 4D SSTVD intensity cost is given by

$$\begin{aligned}
\frac{\partial C_{int}(\mathbf{a})}{\partial b_\alpha} &= \frac{\partial}{\partial b_\alpha} \frac{1}{|\Omega_s|} \frac{1}{|\Omega_t|} \sum_{\mathbf{x}_s \in \Omega_s} \sum_{t \in \Omega_t} \Big(\bar{K}(\mathbf{x}_s, \mathbf{a}) - K(\mathbf{x}, \mathbf{a}) \Big)^2 \\
&= \frac{2}{|\Omega_s|} \sum_{\mathbf{x}_s \in \Omega_s} \frac{1}{|\Omega_t|} \sum_{t \in \Omega_t} \Big[\Big(\bar{K}(\mathbf{x}_s, \mathbf{a}) - K(\mathbf{x}, \mathbf{a}) \Big) \cdot \Big(\frac{\partial}{\partial b_\alpha} \bar{K}(\mathbf{x}_s, \mathbf{a}) - \frac{\partial}{\partial b_\alpha} K(\mathbf{x}, \mathbf{a}) \Big) \Big] \\
&= \frac{2}{|\Omega_s|} \sum_{\mathbf{x}_s \in \Omega_s} \frac{1}{|\Omega_t|} \Big[\sum_{t \in \Omega_t} \Big(\bar{K}(\mathbf{x}_s, \mathbf{a}) \cdot \frac{\partial}{\partial b_\alpha} \bar{K}(\mathbf{x}_s, \mathbf{a}) \Big) - \sum_{t \in \Omega_t} \Big(\bar{K}(\mathbf{x}_s, \mathbf{a}) \cdot \frac{\partial}{\partial b_\alpha} K(\mathbf{x}, \mathbf{a}) \Big) \\
&\qquad - \sum_{t \in \Omega_t} \Big(K(\mathbf{x}, \mathbf{a}) \cdot \frac{\partial}{\partial b_\alpha} \bar{K}(\mathbf{x}_s, \mathbf{a}) \Big) + \sum_{t \in \Omega_t} \Big(K(\mathbf{x}, \mathbf{a}) \cdot \frac{\partial}{\partial b_\alpha} K(\mathbf{x}, \mathbf{a}) \Big) \Big] \\
&= \frac{2}{|\Omega_s|} \sum_{\mathbf{x}_s \in \Omega_s} \frac{1}{|\Omega_t|} \Big[\frac{\partial}{\partial b_\alpha} \bar{K}(\mathbf{x}_s, \mathbf{a}) \cdot \sum_{t \in \Omega_t} \Big(\bar{K}(\mathbf{x}_s, \mathbf{a}) - K(\mathbf{x}, \mathbf{a}) \Big) \\
&\qquad - \sum_{t \in \Omega_t} \Big(\bar{K}(\mathbf{x}_s, \mathbf{a}) - K(\mathbf{x}, \mathbf{a}) \Big) \cdot \frac{\partial}{\partial b_\alpha} K(\mathbf{x}, \mathbf{a}) \Big] \\
&= -\frac{2}{|\Omega_s|} \sum_{\mathbf{x}_s \in \Omega_s} \frac{1}{|\Omega_t|} \sum_{t \in \Omega_t} \Big[\Big(\bar{K}(\mathbf{x}_s, \mathbf{a}) - K(\mathbf{x}, \mathbf{a}) \Big) \cdot \frac{\partial}{\partial b_\alpha} K(\mathbf{x}, \mathbf{a}) \Big] \qquad (2.19)
\end{aligned}$$

where (omitting parameters $\mathbf{x}$ and $\mathbf{a}$),

$$\begin{aligned}
\frac{\partial}{\partial b_\alpha} K(\mathbf{x}, \mathbf{a}) &= \frac{\partial |J_T|}{\partial b_\alpha} \cdot I(\mathbf{T}) + |J_T| \cdot \frac{\partial I(\mathbf{T})}{\partial b_\alpha} \\
&=< \frac{\partial |J_T|}{\partial J_T}, \frac{\partial J_T}{\partial b_\alpha} > \cdot I(\mathbf{T}) + |J_T| \cdot \nabla I(\mathbf{T})^T \frac{\partial \mathbf{T}}{\partial b_\alpha} \\
&=< cof J_T, \frac{\partial J_T}{\partial b_\alpha} > \cdot I(\mathbf{T}) + |J_T| \cdot \nabla I(\mathbf{T})^T \frac{\partial \mathbf{T}}{\partial b_\alpha} \\
&= |J_T| \Big(< J_T^{-T}, \frac{\partial J_T}{\partial b_\alpha} > \cdot I(\mathbf{T}) + \nabla I(\mathbf{T})^T \frac{\partial \mathbf{T}}{\partial b_\alpha} \Big) \qquad (2.20)
\end{aligned}$$

And derivative of the linear elasticity term is (omitting parameters $\mathbf{x}$ and $\mathbf{a}$):

$$\frac{\partial C_{reg}(\mathbf{a})}{\partial b_\alpha} = \frac{\partial}{\partial b_\alpha} \sum_{\mathbf{x}\in\Omega} ||L\mathbf{u}||^2 = 2 \sum_{\mathbf{x}\in\Omega} \sum_{i=1}^{4} \left(L_i \cdot \frac{\partial L_i}{\partial b_\alpha} \right) \tag{2.21}$$

Image registration is often an ill-posed problem with many locally optimal solutions. Without the presence of an explicit target image, 4D registration is even worse in this regard. For example, since the temporal average of the deformed moving tissue-volume images is used as the registration target, two transformations differing by any rigid motions will result in the same value of intensity based cost. Therefore, similar to the work of Balci et al.[1], we impose a further constraint that the temporal average of the displacement field at any spatial location should be zero. Specifically, after every iteration, the derivative with respect to every B-spline coefficient will be modified as below, so that the derivatives at any spatial grid point will sum up to zero across time, resulting in zero average spatial displacement at any spatial location.

$$\frac{\partial C}{\partial a_{p,q,r,s,\alpha}} \leftarrow \frac{\partial C}{\partial a_{p,q,r,s,\alpha}} - \frac{1}{|\Omega_t|} \sum_{\tau\in\Omega_t} \frac{\partial C}{\partial a_{p,q,r,\tau,\alpha}}, \quad \forall\ [p,q,r,s] \in \mathbb{Z}^4 \tag{2.22}$$

An LBFGS optimizer was used in the registration framework because of its rapid quadratic convergence rate and limited memory consumption.

2.1.7 Inverse Transform

In order to get pair-wise spatial transforms between any two phase images, it is necessary to estimate the inverse transform $\mathbf{S} : \Omega \rightarrow \Omega$ that deforms the temporal average of deformed moving tissue-volume image (the target image) to match the original 4D tissue-volume image (the moving image). Because the B-spline parameterization does not have a closed form inverse expression, we estimate the inverse transform separately using a finer B-spline grid[11]. We minimize the following distance cost[3] to find the inverse transformation

$$C_{inv}(\mathbf{a}') = \sum_{\mathbf{x}\in\Omega} ||\mathbf{S}(\mathbf{T}(\mathbf{x}), \mathbf{a}') - \mathbf{x}||^2 \tag{2.23}$$

where $\mathbf{a}'$ is the set of B-spline coefficients parameterizing the inverse 4D transform $\mathbf{S}$. After the optimal forward and inverse transforms $\hat{\mathbf{T}}$ and $\hat{\mathbf{S}}$ have been estimated, we can compose them to acquire spatial transforms between any pair of two time point images $\mu, \nu \in \Omega_t$.

$$\mathbf{T}_{\mu\to\nu}(\mathbf{y}_s) = (\hat{\mathbf{T}}_\mu \circ \hat{\mathbf{S}}_\nu)(\mathbf{y}_s) \tag{2.24}$$

where $\mathbf{y}_s = (y_1, y_2, y_3)^T$ is a spatial point in time point image ν. $\hat{\mathbf{T}}_\mu$ is the spatial transform acquired by evaluating the optimal 4D forward transform at time point μ and $\hat{\mathbf{S}}_\nu$ is the spatial transform acquired by evaluating the optimal 4D inverse transform at time point ν.

2.1.8 Implementation

The proposed 4D SSTVD algorithm was first prototyped in Matlab and tested on simple synthetic images. After it demonstrated its effectiveness, the algorithm was implemented using Elastix package[8] based on Insight Segmentation and Registration Toolkit (ITK) libraries. This algorithm was implemented with multi-threading support. The experiments using clinical 4DCT data sets were run on a machine with Intel i7 5930k CPU (6 cores, 12 threads @ 3.5GHz) and 64GB of DDR4 memory. For each 4DCT data set, we used a multi-grid multi-resolution scheme consisting of 8 resolutions from coarse to fine to estimate both the forward and inverse transforms. The whole process took approximately 2 to 3 hours for each 4DCT data set.

2.1.9 Evaluation Methods

Evaluation of 4D registration frameworks needs to consider both accuracy and temporal smoothness [11][1][19]. 4D registration may not necessarily yield the best accuracy of registration results, but it generates deformation fields that are a lot more temporally smooth and consistent. With comparable or slightly worse accuracy, a temporally smoother outcome is preferred because the resulting estimated motion is biologically more reasonable.

For registration accuracy evaluation, we used the Mean Landmark Error (MLE) [15]. MLE measures the average distance between expert-labeled landmark positions on all phase images (except for the one chosen as reference) and landmark positions obtained by transforming the landmarks from a chosen reference phase onto all other phases. Let $\mathbf{p}_{r,i}$ be the ith landmark on the chosen reference time point r and $\mathbf{T}_{r\to t}$ be the spatial transform from time point r to t. Then $\mathbf{T}_{r\to t}(\mathbf{p}_{r,i}; \mathbf{a})$ is the transformed location in time point t of the landmark $\mathbf{p}_{r,i}$, while $\mathbf{p}_{t,i}$ is the real location of the corresponding landmark in time point t. The Mean Landmark Error is formulated as in Equation 2.25.

$$MLE(\mathbf{a}) = \frac{1}{N(|\Omega_t| - 1)} \sum_{\Omega_t \ni t \neq r} \sum_{i \in N} ||\mathbf{T}_{r\to t}(\mathbf{p}_{r,i}; \mathbf{a}) - \mathbf{p}_{t,i}|| \qquad (2.25)$$

where N is the number of landmarks on every time point image.

For temporal smoothness evaluation, we used the Mean Irregularity of estimated landmark trajectories:

$$MIR(\mathbf{a}) = \frac{1}{N|\Omega_t|} \sum_{t \in \Omega_t} \sum_{i \in N} || \frac{\partial^2 \mathbf{T}_{r \to t}(\mathbf{p}_{r,i}; \mathbf{a})}{\partial t^2} || \tag{2.26}$$

where the second order partial derivatives are computed using centered finite difference assuming Neumann boundary conditions. Denote $\mathbf{q}_{t,i} = \mathbf{T}_{r \to t}(\mathbf{p}_{r,i}; \mathbf{a})$ and we have:

$$\begin{aligned} || \frac{\partial^2 \mathbf{T}_{r \to t}(\mathbf{p}_{r,i}; \mathbf{a})}{\partial t^2} || &= \sqrt{\sum_{\alpha \in \{x,y,z\}} \left(\frac{\partial^2 q_{t,i,\alpha}}{\partial t^2} \right)^2} \\ &= \sqrt{\sum_{\alpha} \left(\frac{q_{t+1,i,\alpha} - 2q_{t,i,\alpha} + q_{t-1,i,\alpha}}{(\Delta t)^2} \right)^2} + O((\Delta t)^2) \end{aligned} \tag{2.27}$$

This measure can be interpreted as the average magnitude of acceleration of landmarks while they traverse their estimated trajectories across time. A smaller MIR value would generally indicate smoother predicted landmark paths and is thus more biologically plausible because the lung would tend to move in a smooth fashion during ventilation.

Finally, to visually and qualitatively examine the result of registration, temporal mean and variance of the final deformed moving tissue-volume image before and after registration were generated. We expect accurate registration would sharpen the mean image and reduce the value of the variance image especially in regions where the mean image was originally blurry [1] [19]. 4D Jacobian image was also computed from the estimated 4D transform that deformed each phase within the original 4D tissue-volume image to match the extreme exhale phase. Temporal mean and variance of the 4D Jacobian image were overlaid on top of the extreme exhale phase image to qualitatively illustrate lung ventilation behavior.

2.2 Experiments and Results

To assess the performance of the proposed 4D tissue preservation registration framework, we performed experiments on 2D+t synthetic image and 4DCT pulmonary data sets, and compared the results of this algorithm with those of existing pairwise 3D SSTVD[20][6] and 4D SSD[11] methods. There exists a trade-off between registration accuracy and temporal smoothness, so we evaluated both of these factors to demonstrate that the proposed 4D SSTVD method achieves a good balance between them.

2.2.1 Synthetic Image Experiments with Matlab Implementation

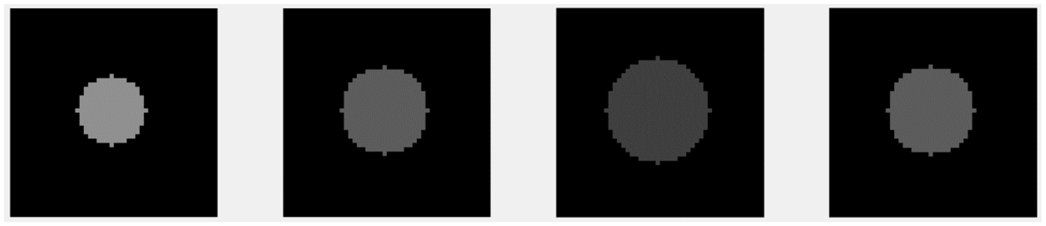
The proposed 4D SSTVD registration framework was first implemented from scratch in Matlab, because of Matlab's fast prototyping capability and ease of debugging. The implemented registration framework is very basic, consisting of 2D+t SSTVD cost function, simple gradient descent optimizer with Armijo line search, linear interpolation moving image resampler, full grid target image sampler, and 2D+t transform parametrized by B-spline. Experiments were conducted on two slightly different 2D+t synthetic images using this basic Matlab implementation of the 2D+t SSTVD algorithm.

2.2.1.1 Experiment 1: Single Disk Time Series

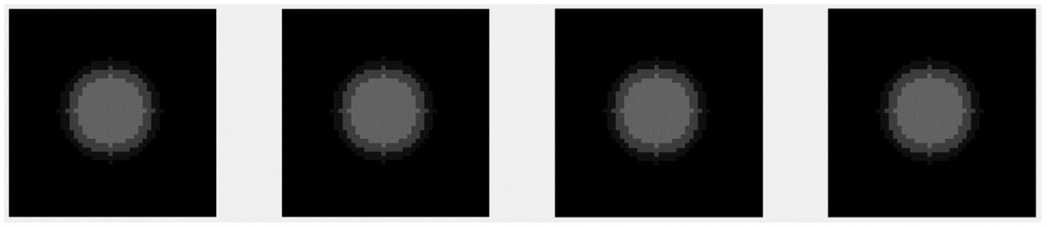
The first image is of dimension 48x48x4, composed of a time series of 4 concentric circles with varying radii and intensities. The 4 circles can be though of as illustrating one period of an infinite cycle of motion, which is intended to abstractly mimic the 4DCT image of lung within the respiratory cycle. The image is assumed to have already been converted into tissue-volume image and the total amount of tissue is preserved. Specifically, the intensity range is $[0, 1]$, representing the fraction of tissue within each pxiel. And for any disk $i \in \{1, 2, 3, 4\}$ within the times series, the area A_i is inversely proportional to the intensity I_i, making $A_i \cdot I_i \equiv Const$. As described in the Methods section, this 2D+t image itself will be treated as the moving image, and the temporal average of 2D+t deformed moving tissue-volume image at each iteration will serve as the target image for that iteration. The algorithm is allowed to run for 210 iterations before it is terminated. The result is visually demonstrated in Fig 2.9. We can see that the initial target image in Sub-figure 2.9a is blurry, being the temporal mean of the original 2D+t image. But the target image in Fig 2.9e, which is used in the final iteration of registration has become a lot sharper, indicating all 2D deformed moving tissue-volume images within the 2D+t time series have been converging toward a common shape (and intensity).

The displacement field in the target image domain, resulting from the final estimated 2D+t forward transform is shown in Fig 2.10. The deformed grid and the Jacobian of the final estimated transform are illustrated in Sub-figure 2.10d and Sub-figure 2.10c, respectively. For such simple synthetic images, we would expect the 2D+t SSTVD cost value and the SSTVD gradient magnitude to decrease monotonically. This is indeed what we observed when plot the cost function value and gradient magnitude with respect to iterations, as shown in Fig 2.11.

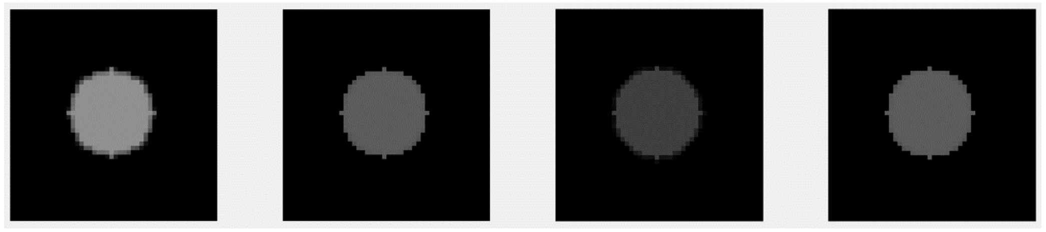
Following the outline of 4D registration framework presented in section 2.1.2,



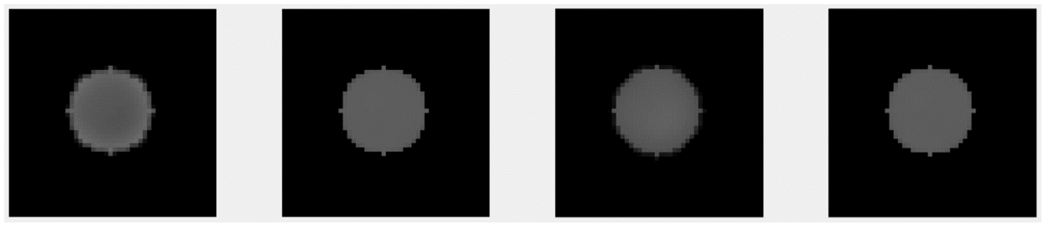
(a) Moving image $I(\mathbf{y})$



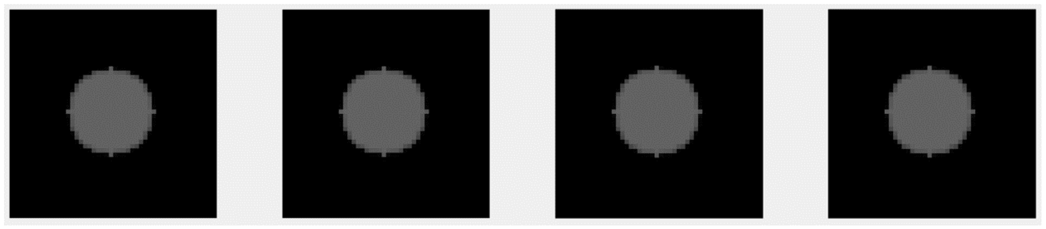
(b) Initial target image $\bar{K}_0(\mathbf{x}_s) = 1/|\Omega_t| \sum_{t\in\Omega_t} I(\mathbf{T}_0(\mathbf{x}_s, t, \mathbf{a})) = \bar{I}(\mathbf{x}_s)$



(c) Final deformed moving image (without Jacobian modification) $I(\mathbf{T}(\mathbf{x}, \mathbf{a}))$

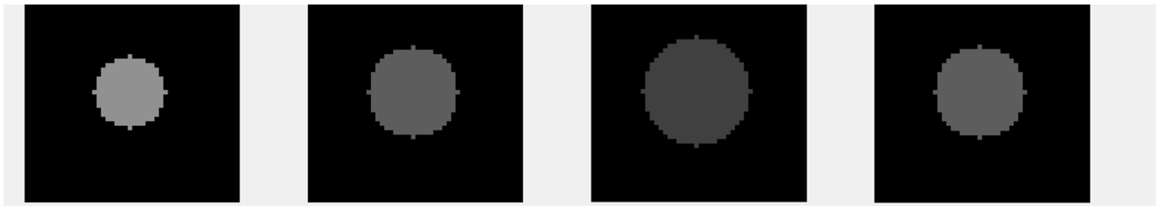


(d) Final deformed moving tissue-volume image $K(\mathbf{x}, \mathbf{a}) = |J_{\mathbf{T}}|(\mathbf{x}, \mathbf{a}) \cdot I(\mathbf{T}(\mathbf{x}, \mathbf{a}))$

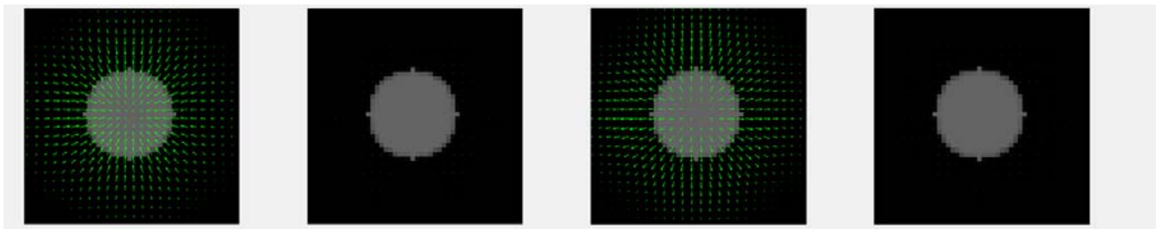


(e) Final target image $\bar{K}(\mathbf{x}_s, \mathbf{a})$

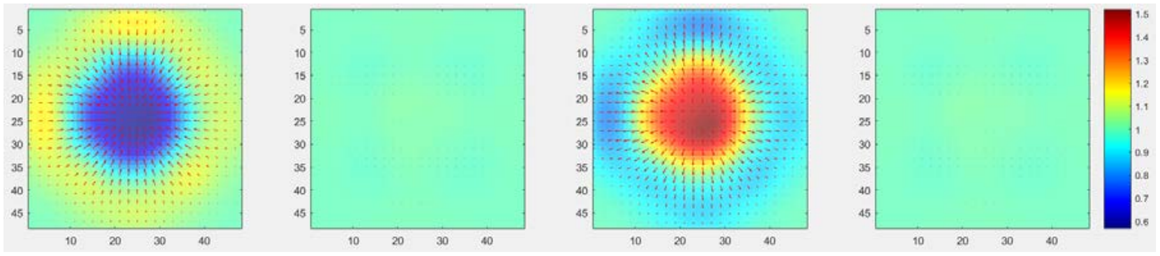
Figure 2.9: Concentric one disk experiment result using Matlab implementation.



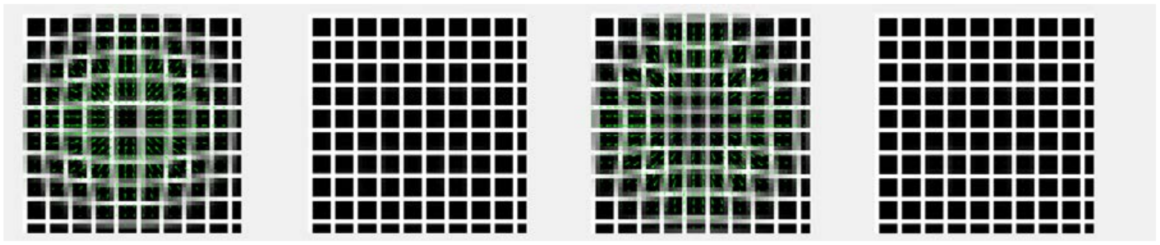
(a) Moving image $I(\mathbf{y})$



(b) Final target image $\bar{K}(\mathbf{x}_s, \mathbf{a})$, overlaid with displacement field of final transform $\mathbf{T}$

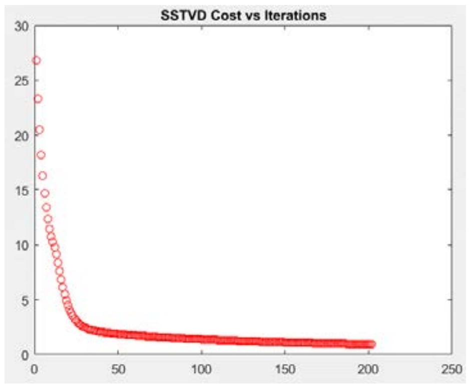


(c) Final Jacobian image $|J_{\mathbf{T}}|(\mathbf{x}, \mathbf{a})$, overlaid with displacement field

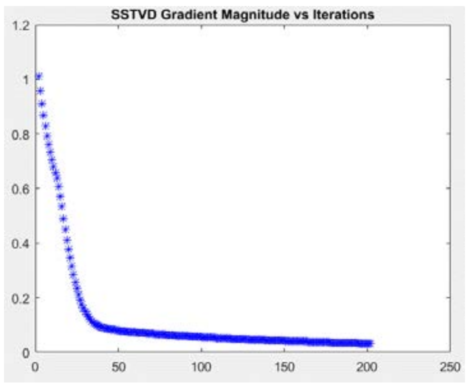


(d) Deformed grid using final forward transform $\mathbf{T}$, overlaid with displacement field

Figure 2.10: 2D+t forward transform qualitative analysis in single disk Matlab experiment.



(a) 2D+t SSTVD cost versus iteration

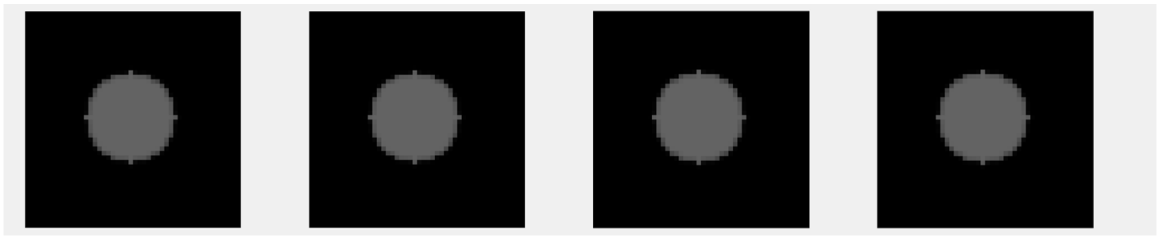


(b) 2D+t SSTVD gradient magnitude versus iteration

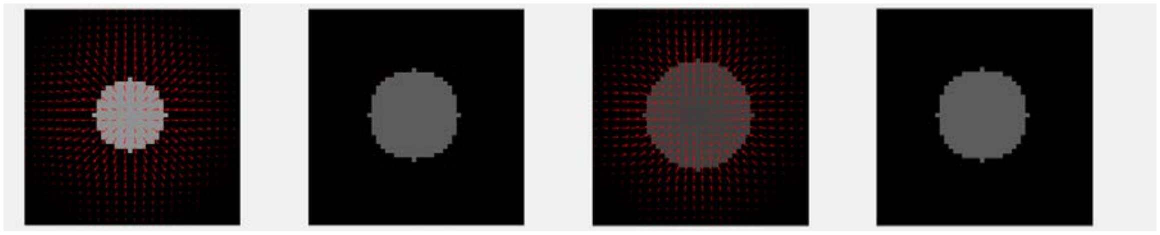
Figure 2.11: Optimization behavior of 2D+t forward transform estimation in single disk Matlab experiment.

we proceed to estimate the 2D+t inverse transform based on the results of the final estimated 2D+t forward transform. In Fig 2.12, the displacement field and the Jacobian in the target image domain (which is the moving image domain for estimating the forward transform), calculated from the final estimated inverse transform are shown.

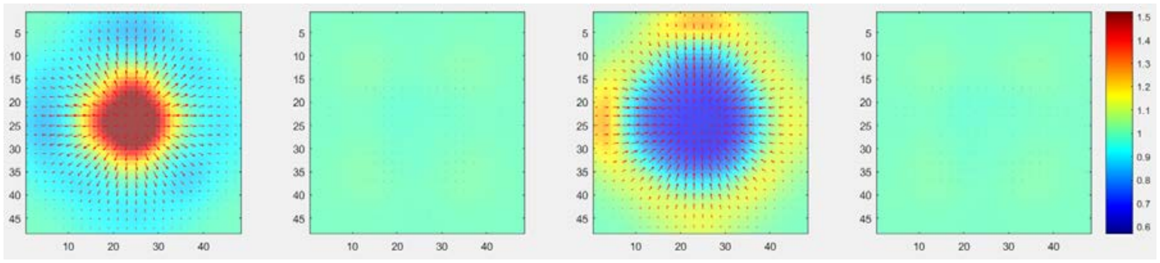
After obtaining both the 2D+t forward and inverse transforms, we could concatenate them to easily acquire 2D spatial transforms between any two time points within the time series. Sub-figure 2.13b illustrates the case where the first time point 2D disk $I(\mathbf{y}_s, 0)$ is the moving image and the third time point 2D disk $I(\mathbf{y}_s, 2)$ is the fixed image. Sub-figure 2.13c shows the case where the two 2D images $I(\mathbf{y}_s, 0)$ and $I(\mathbf{y}_x, 2)$ switch their roles in the registration framework.



(a) Previous final target image $\bar{K}(\mathbf{x}_s, \mathbf{a})$, used as moving image to estimate inverse transform

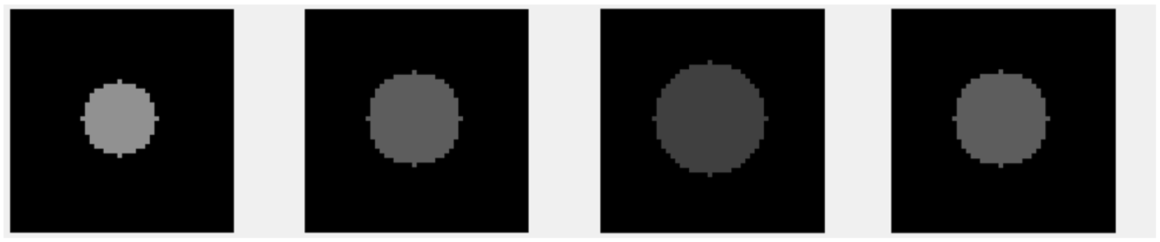


(b) Previous moving image $I(\mathbf{y})$, used as target image to estimate inverse transform

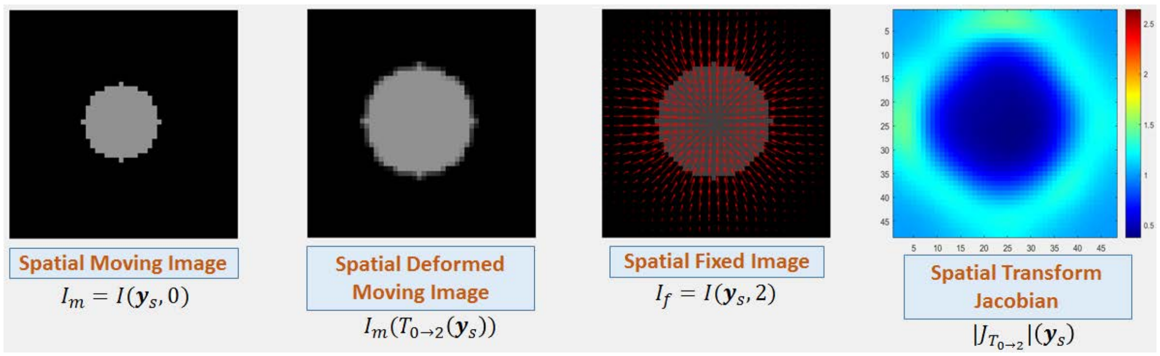


(c) Final Jacobian image $|J_{\mathbf{S}}|(\mathbf{y}, \mathbf{a}')$ for inverse transform, overlaid with displacement field

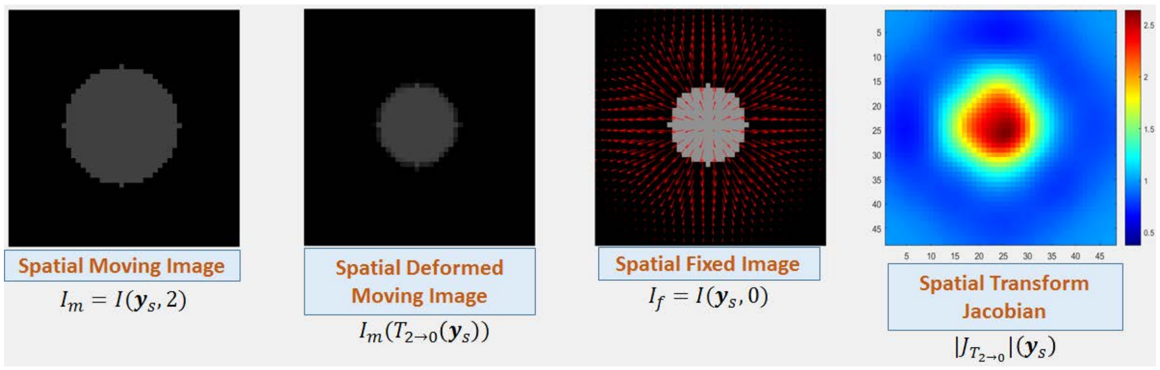
Figure 2.12: 2D+t inverse transform qualitative analysis in single disk Matlab experiment.



(a) 2D+t image $I(\mathbf{y}) = I(\mathbf{y}_s, t), \quad t \in \{0, 1, 2, 3\}$



(b) 2D spatial transform: $I(\mathbf{y}_s, 0)$ moving image, $I(\mathbf{y}_s, 2)$ fixed image



(c) 2D spatial transform: $I(\mathbf{y}_s, 2)$ moving image, $I(\mathbf{y}_s, 0)$ fixed image

Figure 2.13: 2D spatial transforms acquired by composing forward and inverse 2D+t transforms in single disk Matlab experiment.

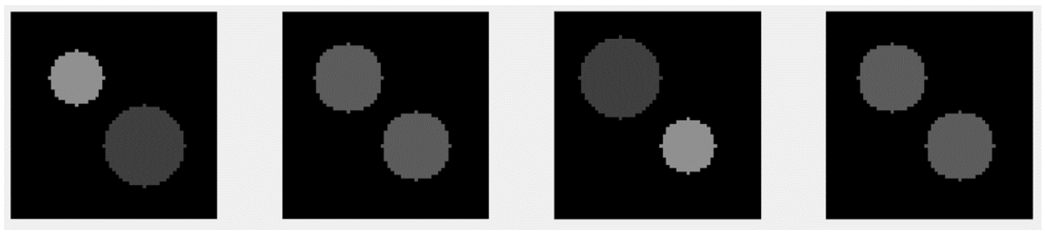
2.2.1.2 Experiment 2: Twin Disks Time Series

The second synthetic 2D+t image on which experiments were conducted using the Matlab implementation is a time series of twin circles of dimension 61x61x4. This synthetic image also complies with the conservation of total amount of tissue across time. The purpose of using twin circles in this 2D+t synthetic image is to test the effectiveness of the proposed registration method when both contraction and expansion are present in the foreground of each 2D time point image during the registration process, as compared to the previous experiment where expansion and contraction do not occur simultaneously in the foreground of any 2D time point image. The registration result is visually illustrated in Fig 2.14.

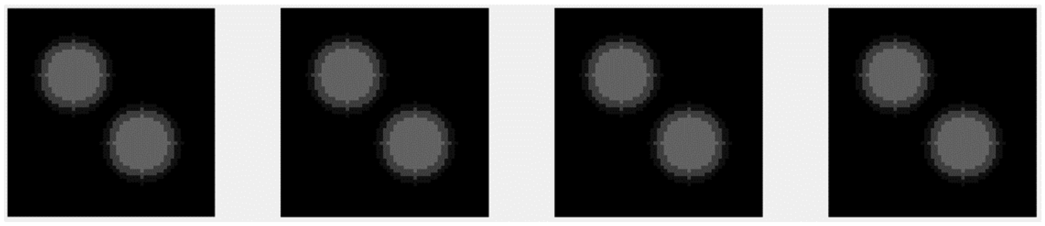
The optimization behavior of the 2D+t forward transform estimation, and the qualitative analysis of the final estimated forward and inverse transforms, including the resulting 2D+t displacement fields and Jacobian images are shown in Fig 2.15, 2.16 and 2.17.

Finally, we could compose the estimated 2D+t forward and inverse transforms to acquire spatial transforms between any pair of 2D time point images. In Sub-figure 2.18b, the first time point 2D image $I(\mathbf{y}_s, 0)$ within the time series is the moving image, while the third time point 2D image $I(\mathbf{y}_s, 2)$ serves as the fixed image. In Sub-figure 2.18c, the first and third 2D images switch their roles in registration framework.

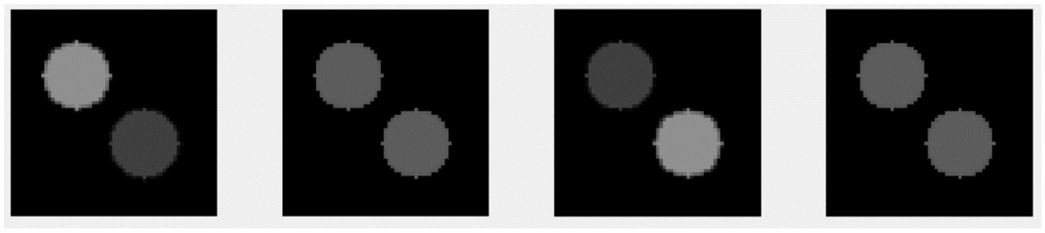
In the above two experiments on 2D+t synthetic images using Matlab implementation, the effectiveness of the proposed 4D SSTVD algorithm has been qualitatively demonstrated. These experiments gave me confidence and motivation to



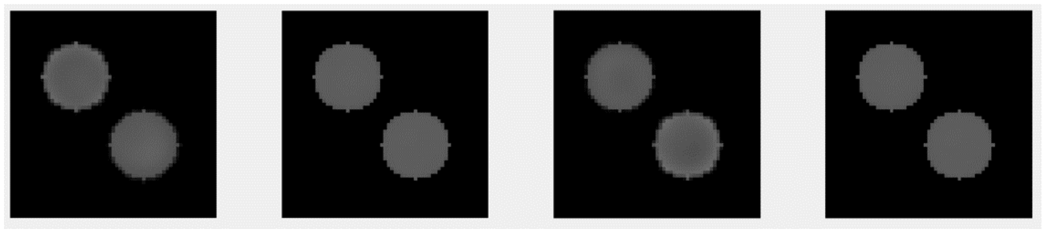
(a) Moving image: $I(\mathbf{y})$



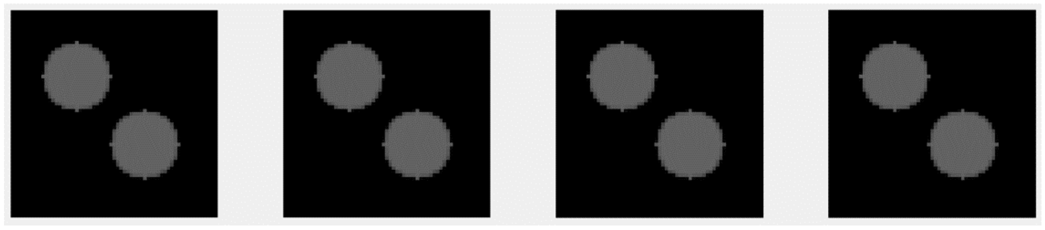
(b) Initial target image $\bar{K}_0(\mathbf{x}_s) = \bar{I}(\mathbf{x}_s)$



(c) Final deformed moving image (without Jacobian modification) $I(T(\mathbf{x}, \mathbf{a}))$

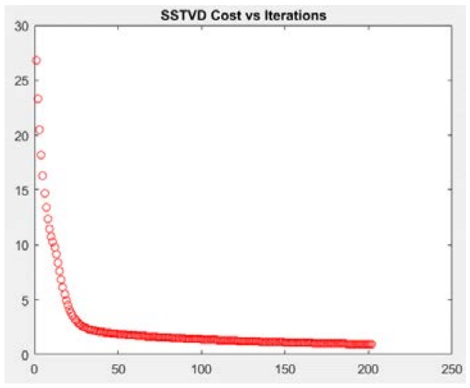


(d) Final deformed moving tissue-volume image $K(\mathbf{x}, \mathbf{a}) = |J_{\mathbf{T}}|(\mathbf{x}, \mathbf{a}) \cdot I(T(\mathbf{x}, \mathbf{a}))$

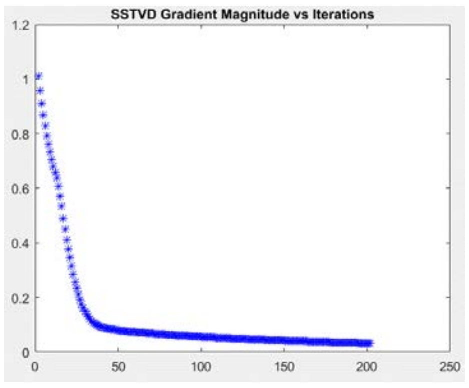


(e) Final target image $\bar{K}(\mathbf{x}_s, \mathbf{a})$

Figure 2.14: Concentric twin circles experiment result using Matlab implementation.



(a) 2D+t SSTVD cost versus iteration



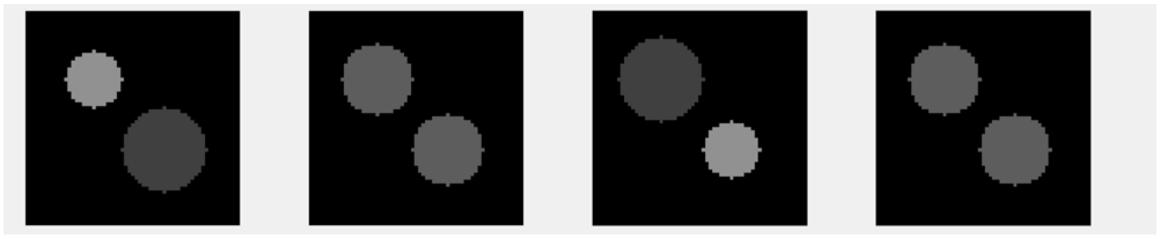
(b) 2D+t SSTVD gradient magnitude versus iteration

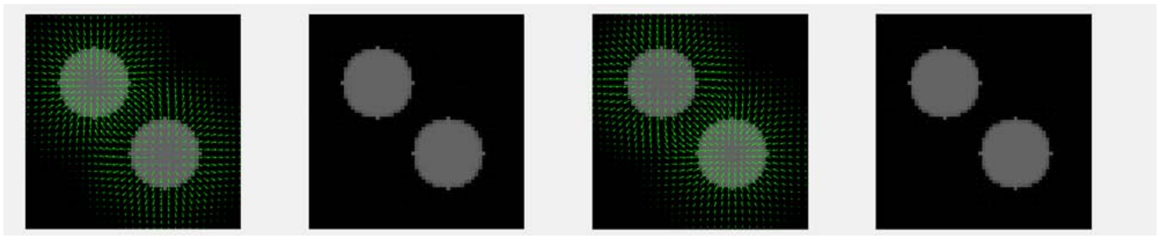
Figure 2.15: Optimization behavior of 2D+t forward transform estimation in twin circles Matlab experiment.

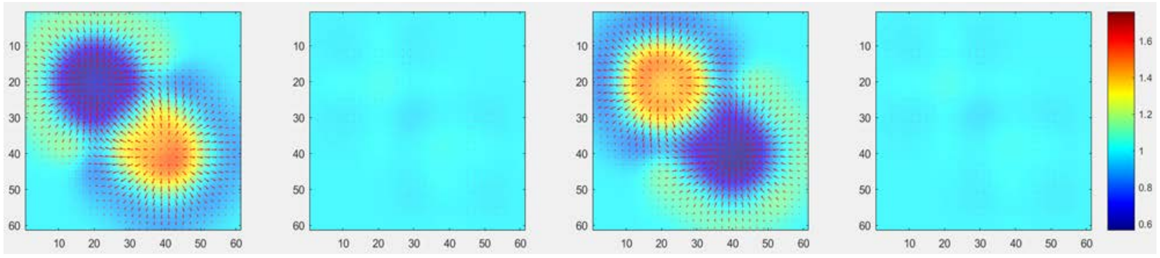
implement the algorithm in C++ using Elastix package based on ITK library, so that it can be used for clinical 4DCT pulmonary image registration.

2.2.2 Synthetic Image Experiments with C++ Implementation

The algorithm was then implemented in C++ using the Elastix package based on ITK. The proposed 4D SSTVD algorithm was tested on a 129x129x129-pixel 2D+t synthetic image as shown in Figure 2.19. The image consists of a time series of disk-shaped regions, whose centers' X coordinates remained the same across time while Y coordinates traced a periodic sinusoid trajectory in time. Conceptually, we can think of the 2D images within the time series as having already been converted into tissue-volume images and their intensity values are within range $[0, 1]$, representing fraction


(a) Moving image $I(\mathbf{y})$


(b) Final target image $\bar{K}(\mathbf{x}_s, \mathbf{a})$, overlaid with displacement field of final transform $\mathbf{T}$



(c) Final Jacobian image $|J_{\mathbf{T}}|(\mathbf{x}, \mathbf{a})$, overlaid with displacement field

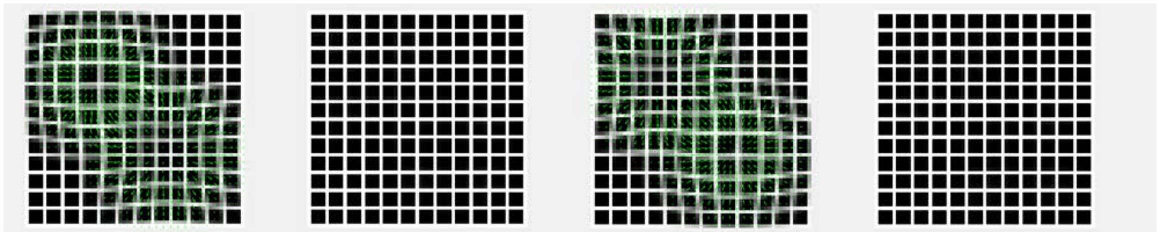
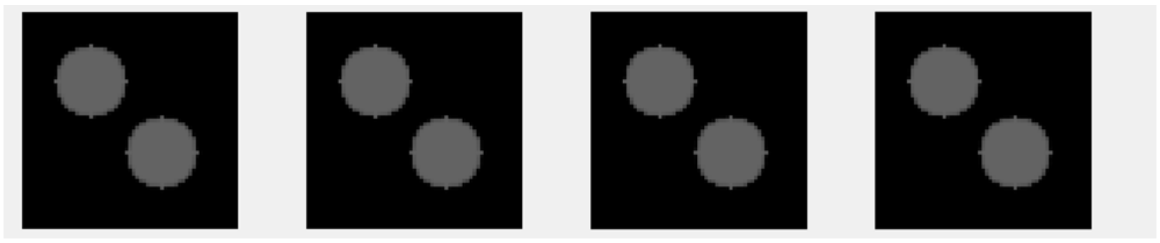
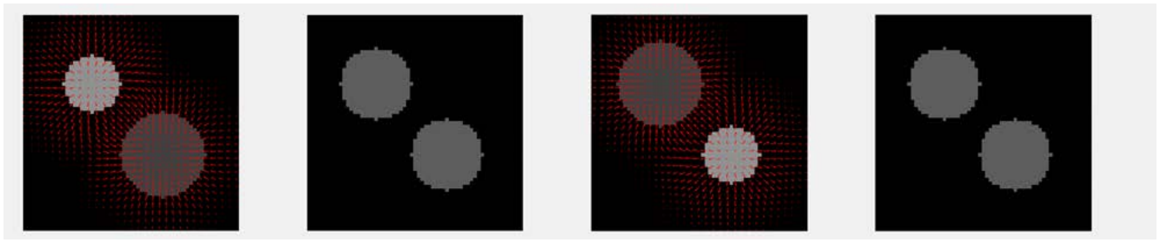

(d) Deformed grid using final forward transform $\mathbf{T}$, overlaid with displacement field

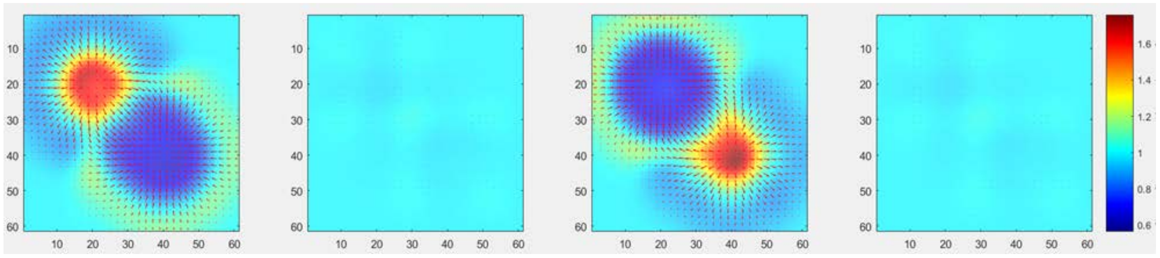
Figure 2.16: 2D+t forward transform qualitative analysis in twin circles Matlab experiment.



(a) Previous final target image $\bar{K}(\mathbf{x}_s, \mathbf{a})$, used as moving image to estimate inverse transform

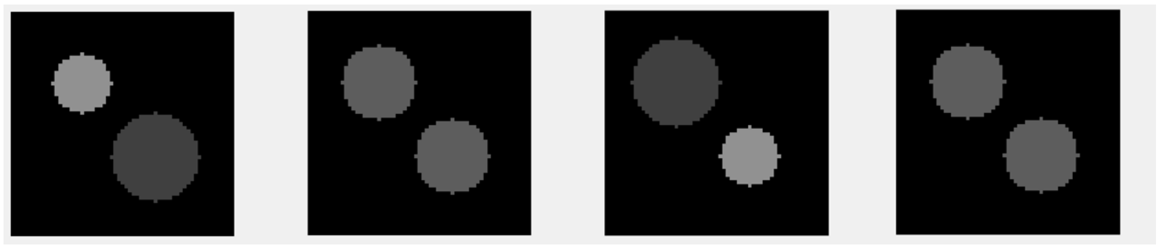


(b) Previous moving image $I(\mathbf{y})$, used as target image to estimate inverse transform

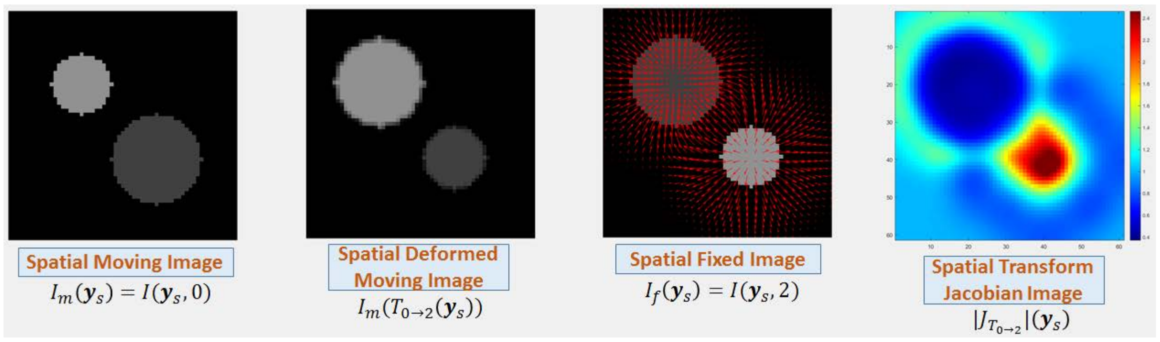


(c) Final Jacobian image $|J_{\mathbf{S}}|(\mathbf{y}, \mathbf{a}')$ for inverse transform, overlaid with displacement field

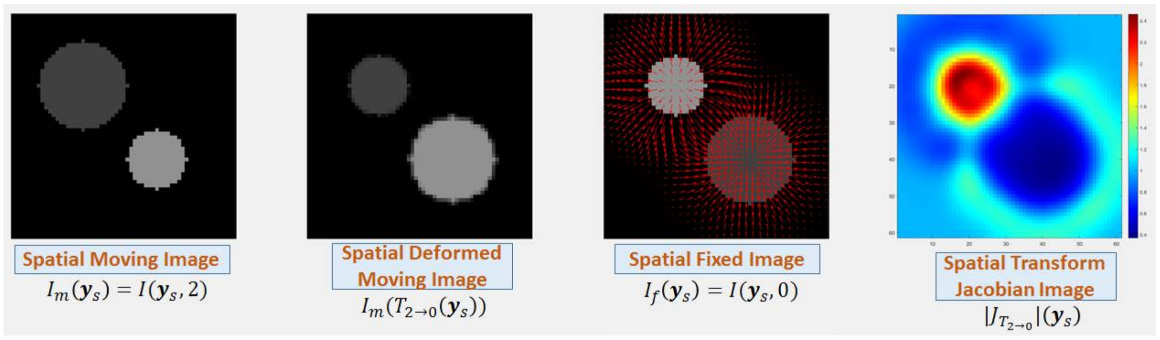
Figure 2.17: 2D+t inverse transform qualitative analysis in twin circles Matlab experiment.



(a) 2D+t image $I(\mathbf{y}) = I(\mathbf{y}_s, t), \quad t \in \{0, 1, 2, 3\}$


Spatial Moving Image
$I_m(\boldsymbol{y}_s) = I(\boldsymbol{y}_s, 0)$
Spatial Deformed Moving Image
$I_m(T_{0\to2}(\boldsymbol{y}_s))$
Spatial Fixed Image
$I_f(\boldsymbol{y}_s) = I(\boldsymbol{y}_s, 2)$
Spatial Transform Jacobian Image
$|J_{T_{0\to2}}|(\boldsymbol{y}_s)$

(b) 2D spatial transform: $I(\mathbf{y}_s, 0)$ moving image, $I(\mathbf{y}_s, 2)$ fixed image

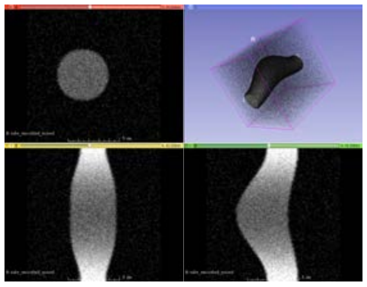

Spatial Moving Image
$I_m(\boldsymbol{y}_s) = I(\boldsymbol{y}_s, 2)$
Spatial Deformed Moving Image
$I_m(T_{2\to0}(\boldsymbol{y}_s))$
Spatial Fixed Image
$I_f(\boldsymbol{y}_s) = I(\boldsymbol{y}_s, 0)$
Spatial Transform Jacobian Image
$|J_{T_{2\to0}}|(\boldsymbol{y}_s)$

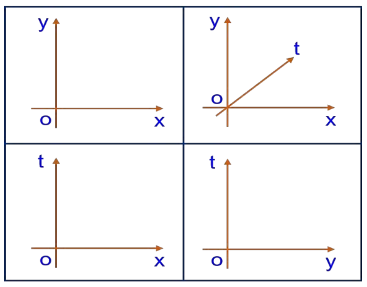
(c) 2D spatial transform: $I(\mathbf{y}_s, 2)$ moving image, $I(\mathbf{y}_s, 0)$ fixed image

Figure 2.18: 2D spatial transforms acquired by composing forward and inverse 2D+t transforms in twin circles Matlab experiment.

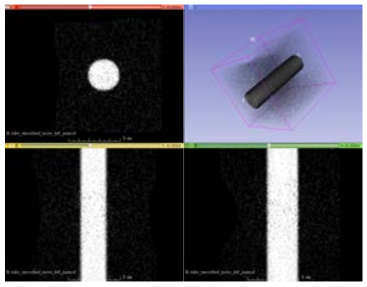
of tissue within voxels. The radii and intensities of the disks vary with respect to time but the total "amount of tissue" remained constant. This means the intensity is inversely proportional to the area of the disk at any time. For example, if the smallest disk at $t = 0$ has intensity I_1 and area A_1, then a disk later in the time series with area A_n will have intensity $I_n = A_1/A_n \cdot I_1$. The synthetic image was blurred by a Gaussian filter with standard deviation of 1 and Guassian noise was applied with 0 mean and 0.005 variance. Each sub-figure in Figure 2.19 consists of four views. The upper-left view is the disk at the middle time point. The lower-left and lower-right views are cross sections of the time series of disks parallel to x-t and y-t plane, respectively. The upper-right view is a 3D (2D+t) rendering of the synthetic image. The orientation of the four views of the synthetic image is as shown in Figure 2.19b.

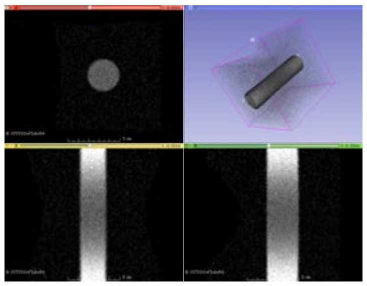
This 2D+t synthetic image is intended to simulate a real 4DCT data set in the sense that the fraction of tissue within voxels varies periodically throughout the respiratory cycle but the total amount of tissue is conserved. Using the proposed 4D SSTVD method, as well as the existing 4D SSD method, we performed registration on this 2D+t synthetic image and chose the 2D image at $t = 0$, whose disk-shaped region has the highest intensity and the smallest radius, as the target image. We used the same optimization parameters for both 4D SSTVD and 4D SSD methods, with a total of 8 resolutions in the multi-grid multi-resolution registration framework. Let's denote the 2D+t synthetic tissue-volume image as I, and the estimated transforms using 4D SSTVD and 4D SSD methods as $\mathbf{T}^r_{SSTVD}$ and $\mathbf{T}^r_{SSD}$, where $r \in \{0, 1, ..., 7\}$ is the resolution index. Then the deformed 2D+t synthetic tissue-volume image (without

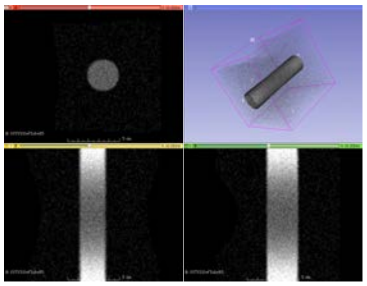

(a) I

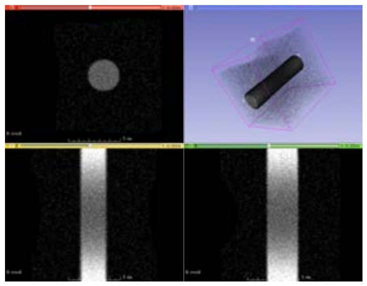


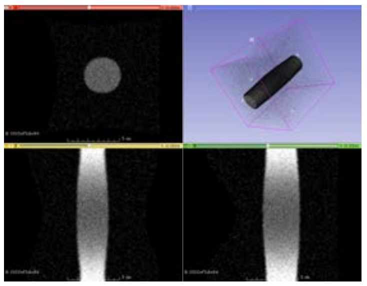
(b) Orientation

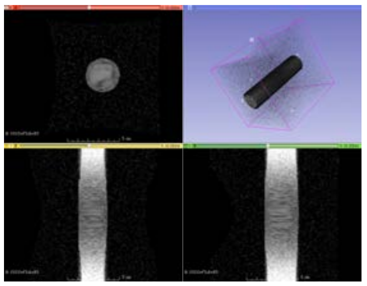

(c) $|J_{\mathbf{T}^7_{SSTVD}}| \cdot I(\mathbf{T}^7_{SSTVD})$


(d) $I(\mathbf{T}^4_{SSTVD})$


(e) $I(\mathbf{T}^5_{SSTVD})$


(f) $I(\mathbf{T}^7_{SSTVD})$


(g) $I(\mathbf{T}^4_{SSD})$


(h) $I(\mathbf{T}^5_{SSD})$

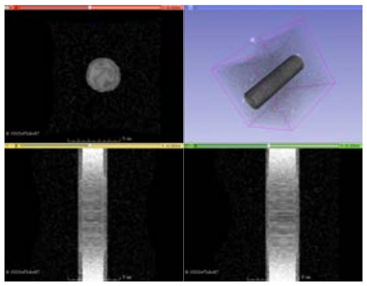

(i) $I(\mathbf{T}^7_{SSD})$

Figure 2.19: Comparison between results from 4D SSTVD and 4D SSD algorithms to register 2D+t synthetic tissue-volume image I.

Jacobian modification) are $I(\mathbf{T}^r_{SSTVD})$ and $I(\mathbf{T}^r_{SSD})$. Through visual observation of Sub-Figure 2.19d-2.19f, we can see that after resolution $r = 4$, the 4D SSTVD method already roughly aligned the time series and at finer resolutions, the results seemed to have stabilized. In Sub-Figure 2.19c, we applied proper Jacobian modification $|J_{\mathbf{T}^7_{SSTVD}}|$ onto the deformed moving image $I(\mathbf{T}^7_{SSTVD})$ to obtain the appropriate 2D+t deformed moving tissue-volume image $|J_{\mathbf{T}^7_{SSTVD}}| \cdot I(\mathbf{T}^7_{SSTVD})$, in which the 2D deformed moving tissue-volume images at other time points carry similar radius and intensity value to the target image at $t = 0$. In comparison, after resolution $r = 4$, the 4D SSD method didn't quite align the time series, as shown in Sub-Figure 2.19d. And when the 4D SSD registration proceeded to finer resolutions, the registration process seemed to have collapsed, resulting in biologically infeasible transformations, as shown in Sub-Figure 2.19h, 2.19i.

2.2.3 Clinical 4DCT Pulmonary Image Experiments

Quantitative experiments were performed on respiratory-gated 4DCT pulmonary data sets. Specifically, we used the publicly available POPI data set containing 4D landmarks (100 landmarks on each phase) for 3 patients, provided by J. Vandemeulebroucke et al.[17]. For each patient, the 4DCT image data consisted of 10 phase images (00, 10, 20, 30, 40, 50, 60, 70, 80, 90) corresponding to 10 breathing phases within the respiratory cycle. We compared the performance of pairwise 3D SSTVD, 4D SSD and the proposed 4D SSTVD tissue preservation algorithm. For all the three algorithms, we obtained the 3D spatial transforms from all phases to the extreme exhale phase. For the pairwise 3D SSTVD registration scheme, the 3D spatial transforms were acquired by using the extreme exhale phase as an explicit target image. For the 4D SSD and the proposed 4D SSTVD algorithms, the 3D spatial transforms to the extreme exhale phase were obtained by first estimating the forward and inverse 4D transforms and then composing them, as discussed in the Method section. Mean landmark error and mean irregularity, as introduced in previous subsection, were computed and the results are shown in Tables 2.2 and 2.3. The last row of Table 2.2 consists of the results reported by Vandemeulebroucke el al.[17]. Figures 2.20 and 2.21 show in bar graph the mean landmark error and mean landmark irregularity for all POPI data sets using 4D SSTVD, 4D SSD and 3D SSTVD algorithms. The height of the bars are the mean values while the full length of the red line segments indicate the standard deviations.

Accuracy (mm)	POPI Patient 1	POPI Patient 2	POPI Patient 3
Before Registration	3.44 ± 3.06	6.41 ± 6.09	3.65 ± 3.89
Proposed 4D SSTVD	0.83 ± 0.63	1.11 ± 0.89	0.87 ± 0.77
4D SSD	0.83 ± 0.65	1.43 ± 1.83	1.02 ± 1.09
3D SSTVD Pairwise	0.80 ± 0.63	1.21 ± 1.29	0.92 ± 0.89
Vandemeule-broucke et al.	0.96 ± 0.66	1.20 ± 0.96	1.11 ± 1.14

Table 2.2: Registration accuracy measured by average landmark error (mean ± standard deviation, smaller is better).

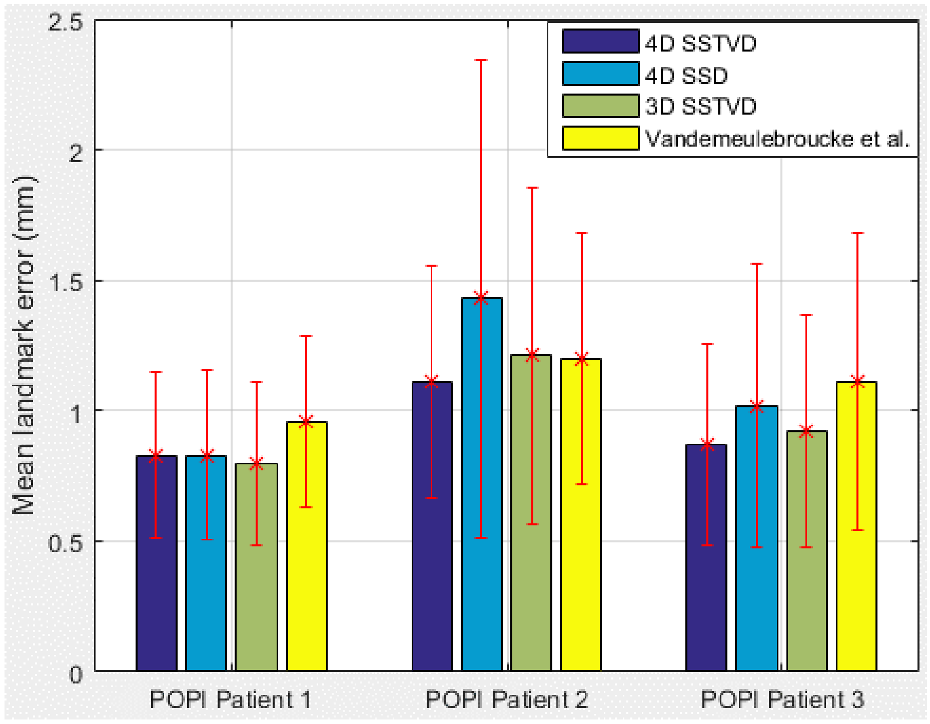


Figure 2.20: Mean landmark error for all POPI data sets using 4D SSTVD, 4D SSD, 3D SSTVD algorithms.

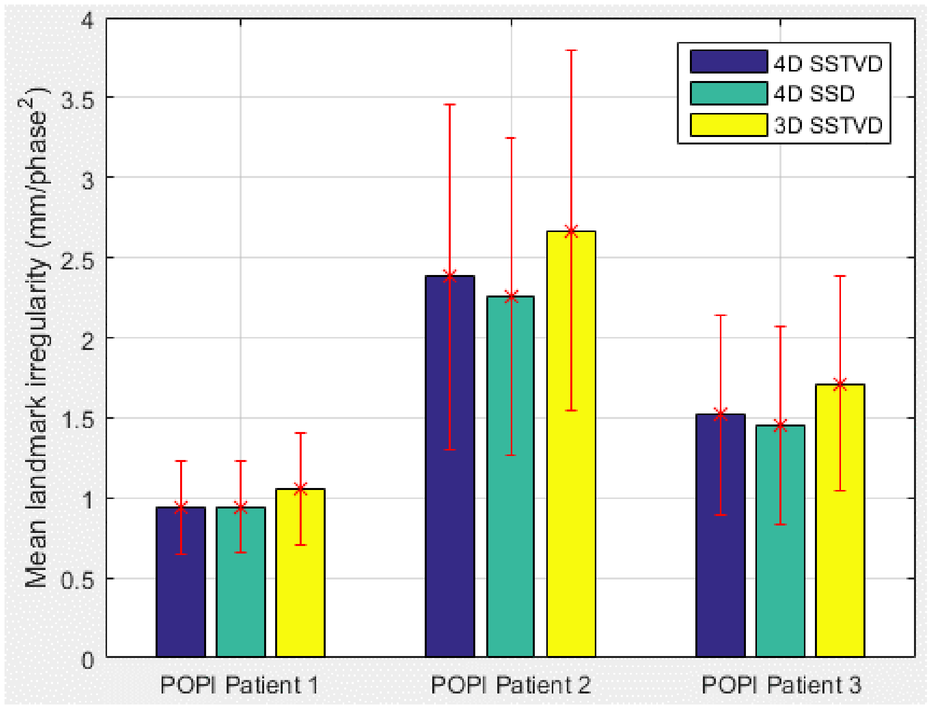


Figure 2.21: Mean landmark irregularity for all POPI data sets using 4D SSTVD, 4D SSD, 3D SSTVD algorithms.

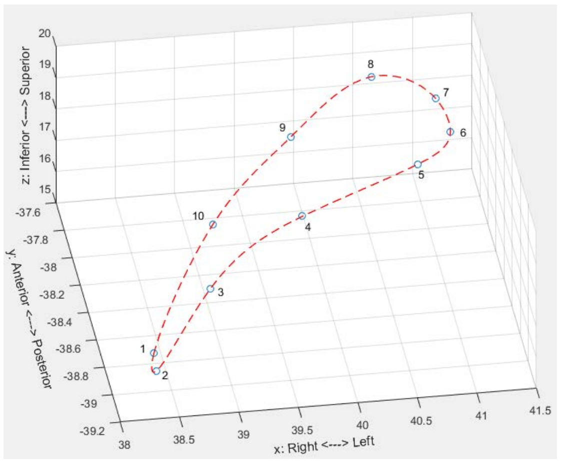
Irregularity ($mm/phase^2$)	POPI Patient 1	POPI Patient 2	POPI Patient 3
Proposed 4D SSTVD	0.94 ± 0.59	2.38 ± 2.15	1.52 ± 1.25
4D SSD	0.94 ± 0.57	2.26 ± 1.98	1.45 ± 1.23
3D SSTVD Pairwise	1.06 ± 0.70	2.67 ± 2.26	1.71 ± 1.34

Table 2.3: Temporal smoothness measured by average landmark trajectory irregularity (mean ± standard deviation, smaller is better).

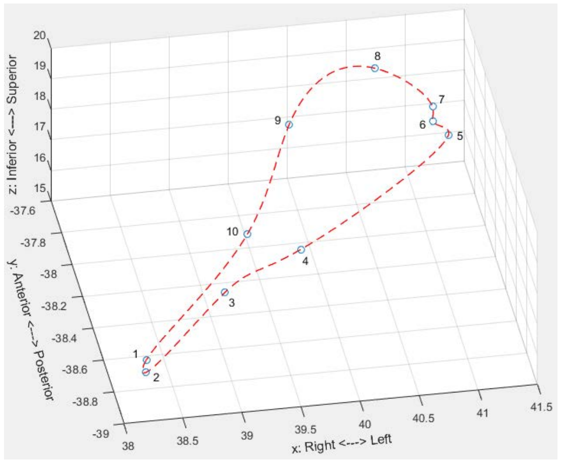
We can see that the proposed 4D SSTVD algorithm achieved better accuracy than 4D SSD method. At the same time, it achieved better temporal smoothness compared to 3D pair-wise SSTVD, which is measured by the average magnitude of acceleration of landmarks when they move along their estimated paths through the respiratory cycle. It should be noted that the relatively big difference among irregularity values for different data sets might stem from the possibility that landmarks for different data sets reside in different regions of the lung. For example, there may be more landmarks located around the diaphragm or other high functioning regions of the lung in POPI Patient 2 than in POPI Patient 1, causing the landmark irregularity to be generally higher for the former.

To further illustrate the advantage of 4D SSTVD algorithm over 3D SSTVD in terms of temporal smoothness, we traced two specific landmarks, i.e., landmark No.47 and No.65 in POPI Patient 1 data set. For 4D and 3D SSTVD algorithms, the mean landmark errors for landmark No.47 are 0.71 mm and 0.61 mm, and the mean landmark irregularity are 0.72 $mm/phase^2$ and 1.00 $mm/phase^2$. So for this landmark, the 4D SSTVD algorithm is slightly worse than the 3D SSTVD in terms of registration accuracy (0.10 mm bigger), but it is better in terms of temporal smoothness (0.28 $mm/phase^2$ smaller). The difference in temporal smoothness can be best seen in Fig.2.22, where the estimated trajectories of landmark No.47 using 4D and pairwise 3D SSTVD algorithms are shown. The blue circles indicate estimated landmark locations on all phases resulting from the 3D spatial transforms deforming all the phases to match the extreme exhale phase. The red dashed curves are obtained by cubic spline interpolation using built-in functions of Matlab. The landmark coordinates are given as in physical/world coordinate system, with units of mm. For landmark No.65, the mean landmark error for 4D and 3D SSTVD algorithms are 0.64 mm and 0.68 mm, respectively. And the mean landmark irregularity for 4D and 3D SSTVD are 0.88 $mm/phase^2$ and 0.96 $mm/phase^2$, respectively. 4D SSTVD achieves both better accuracy and better temporal smoothness for this landmark, and the difference in temporal smoothness between the 4D and 3D algorithms are shown in Fig.2.23.

Visual inspection of the 4D SSTVD registration results on all three POPI data sets are illustrated in Figure 2.24 through 2.26. Before registration, we can see that for all data sets, the 3D temporal mean of the 4D tissue-volume images were blurry, especially in regions of vessels and the diaphragm. Accordingly, the 3D temporal variance images all had high values in those blurry areas. After registration using

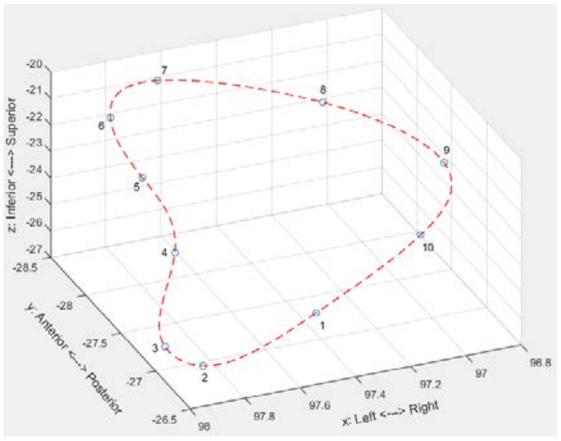


(a) Estimated trajectory of landmark No.47 using 4D SSTVD.

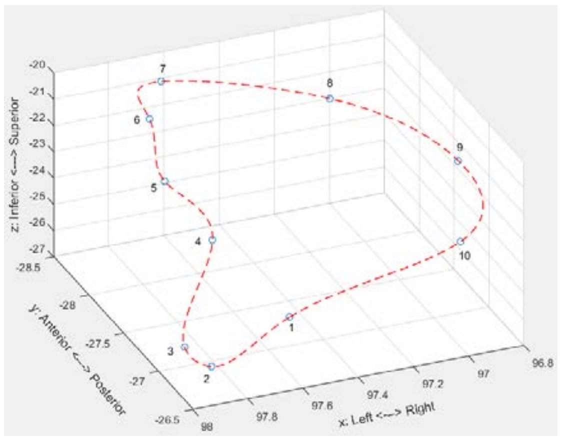


(b) Estimated trajectory of landmark No.47 using 3D SSTVD.

Figure 2.22: Estimated trajectories of landmark No.47 in POPI Patient 1 data set using 4D and 3D SSTVD algorithms. 4D SSTVD gives better temporal smoothness even though its registration accuracy may be slightly worse than 3D SSTVD.



(a) Estimated trajectory of landmark No.65 using 4D SSTVD.



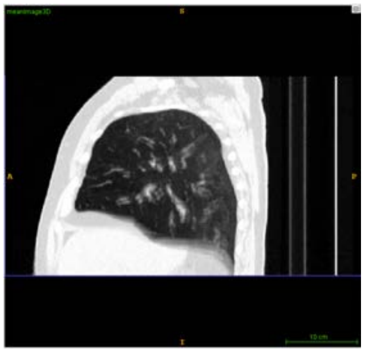
(b) Estimated trajectory of landmark No.65 using 3D SSTVD.

Figure 2.23: Estimated trajectories of landmark No.65 in POPI Patient 1 data set using 4D and 3D SSTVD algorithms. 4D SSTVD gives better temporal smoothness even though its registration accuracy may be slightly worse than 3D SSTVD.

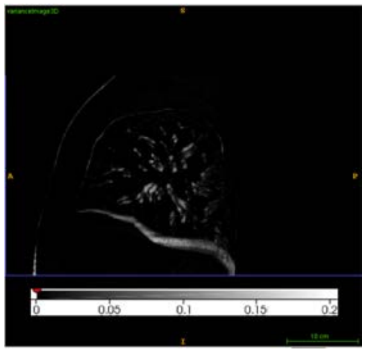
the proposed 4D SSTVD algorithm, the 3D mean images became a lot sharper and the 3D variance images grew a lot darker within the lung region, indicating that the registration indeed made the phase images better aligned. It is also worth noting that each of the after registration 3D variance images all has a bright stripe near the bottom of the image, outside the lung region. These areas of high variances exist because all phase images have been deformed to match the extreme exhale phase, and their bottom regions were “moved up” by different amounts, resulting in intensity differences at these regions across time dimension in the deformed moving tissue-volume image.

4D Jacobian image was also computed from the estimated 4D transform that deformed all phase images within the original 4D tissue-volume image to match the extreme exhale phase. 4D masks for the 4DCT data sets were generated using a 4D optimal surface finding (OSF) algorithm proposed by Gerard et al. [18]. The masks were applied onto the Jacobian images to mask out regions outside the lung, and the Jacobian values within the lung region would provide relevant information about pulmonary ventilation behavior. For our experiments on clinical 4DCT data sets using the proposed 4D SSTVD method, all Jacobian values obtained from the estimated 4D transform were positive, indicating no folding or collapsing of space was introduced by the transform and thus the predicted lung motion would be biologically feasible. In Sub-Figure 2.26c, the temporal mean of 4D Jacobian image was overlaid on the extreme exhale phase image to qualitatively illustrate the average ventilation behavior (mostly expansion) of the lung starting from the extreme exhale phase. In

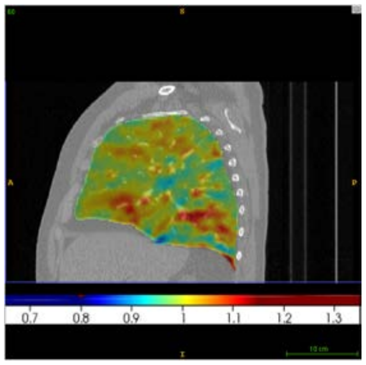
Sub-Figure 2.26f, the temporal variance of 4D Jacobian image was overlaid on the extreme exhale phase. Larger values of Jacobian variance indicate bigger differences among the predicted expansions/contractions from the extreme exhale phase to each of the other phases. High Jacobian variance regions roughly correspond to the most blurry regions in Sub-Figure 2.26a.



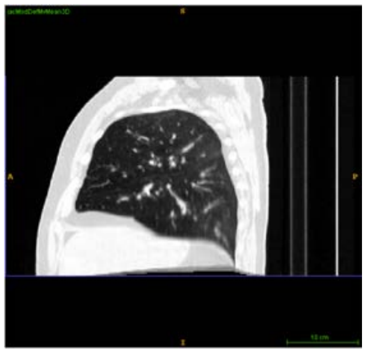
(a) Mean before



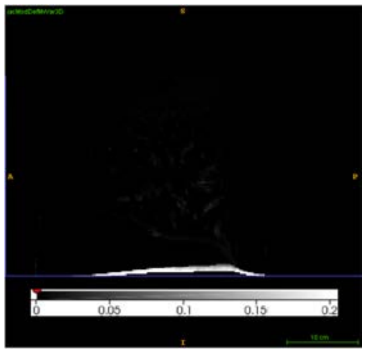
(b) Variance before



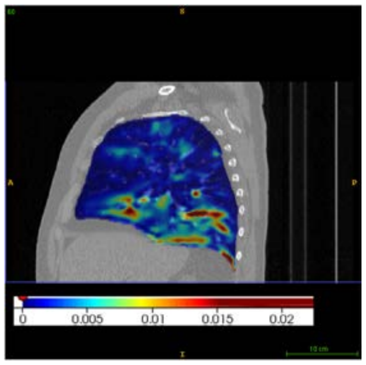
(c) Jacobian mean



(d) Mean after

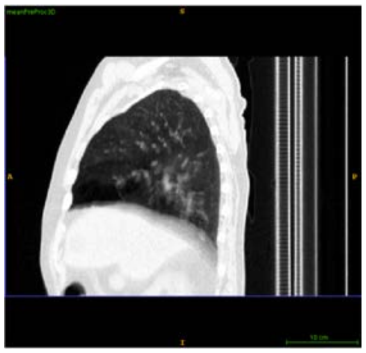


(e) Variance after

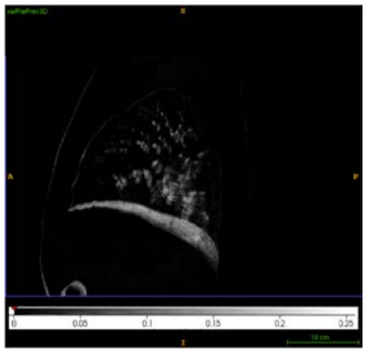


(f) Jacobian variance

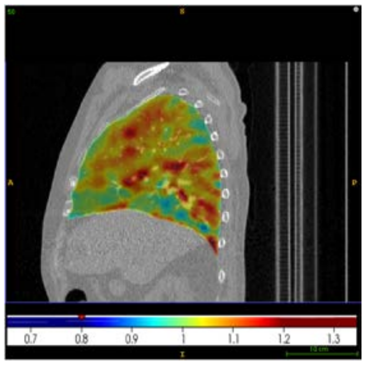
Figure 2.24: Results of 4D SSTVD for POPI Patient 1, sagittal view. (a) Temporal mean of 4D tissue-volume image before registration. (b) Temporal variance of 4D tissue-volume image before registration. (c) Temporal mean of 4D Jacobian image overlaid on the extreme exhale phase. (d) Temporal variance of 4D tissue-volume image after registration. (e) Temporal variance of 4D tissue-volume image before registration. (f) Temporal variance of 4D Jacobian image overlaid on the extreme exhale phase.



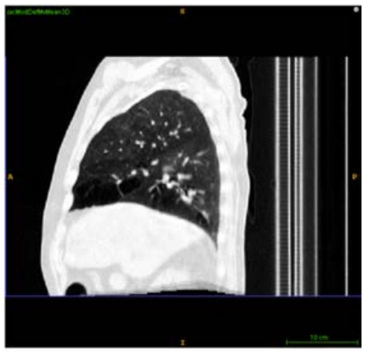
(a) Mean before



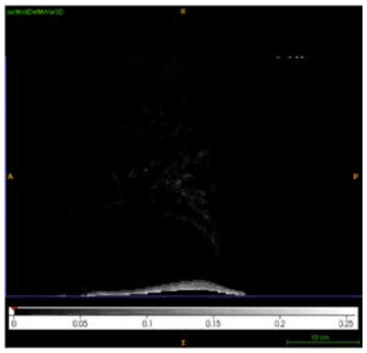
(b) Variance before



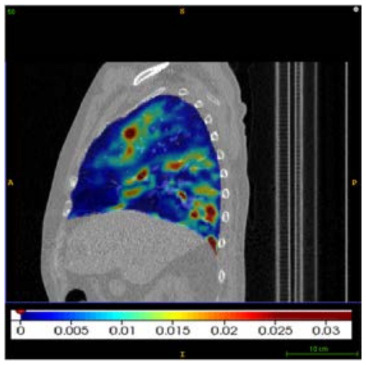
(c) Jacobian mean



(d) Mean after

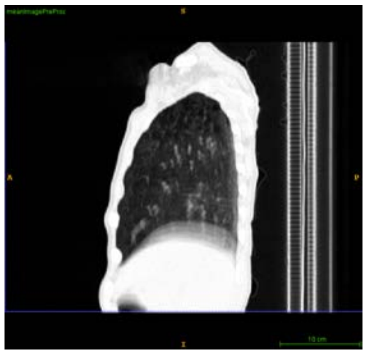


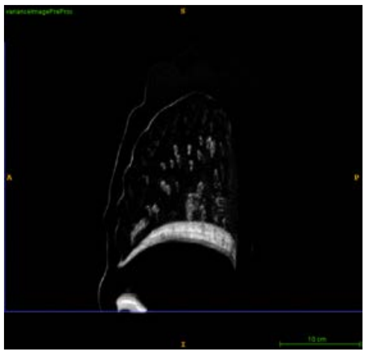
(e) Variance after

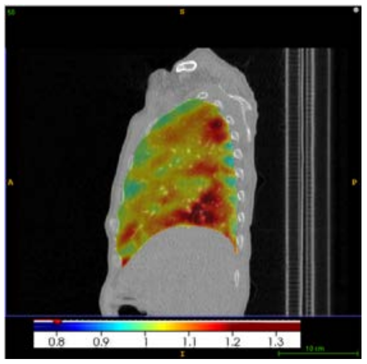


(f) Jacobian variance

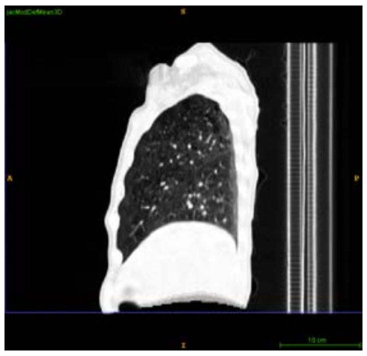
Figure 2.25: Results of 4D SSTVD for POPI Patient 2, sagittal view. (a) Temporal mean of 4D tissue-volume image before registration. (b) Temporal variance of 4D tissue-volume image before registration. (c) Temporal mean of 4D Jacobian image overlaid on the extreme exhale phase. (d) Temporal variance of 4D tissue-volume image after registration. (e) Temporal variance of 4D tissue-volume image before registration. (f) Temporal variance of 4D Jacobian image overlaid on the extreme exhale phase.

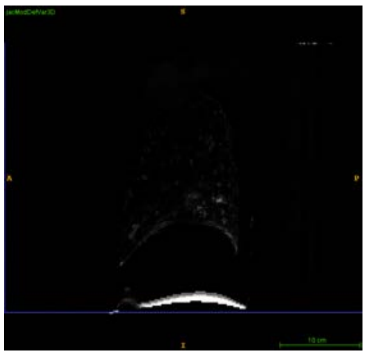

(a) Mean before

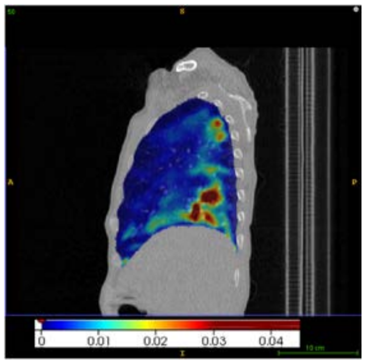

(b) Variance before



(c) Jacobian mean


(d) Mean after


(e) Variance after



(f) Jacobian variance

Figure 2.26: Results of 4D SSTVD for POPI Patient 3, sagittal view. (a) Temporal mean of 4D tissue-volume image before registration. (b) Temporal variance of 4D tissue-volume image before registration. (c) Temporal mean of 4D Jacobian image overlaid on the extreme exhale phase. (d) Temporal variance of 4D tissue-volume image after registration. (e) Temporal variance of 4D tissue-volume image before registration. (f) Temporal variance of 4D Jacobian image overlaid on the extreme exhale phase.

CHAPTER 3
4D DATA SETS LANDMARKING SOFTWARE

3.1 Purpose

Identifying point-wise correspondence in 4DCT pulmonary data sets plays an indispensable role for diagnosis and evaluation of pulmonary cancer and is crucial for assessing the accuracy and temporal smoothness of registration results, which in turn provide valuable information to guide radiotherapy. Extending the approach pioneered by Murphy et al.[12], a GUI landmarking software for 4D data sets was designed and implemented in Java as an ImageJ plug-in to aid the user to accurately and efficiently label corresponding landmark points in 4D data sets.

3.2 Usage

The general procedure for using this landmarking software tool can be divided into three major steps. The first step is to acquire a set of landmarks on one specific phase image within the 4D data set (preferably on one of the two extreme phases for pulmonary 4D data sets). This can be achieved by either manually labeling the set of desired landmarks on that specific phase using this landmarking software, or by using the automatic landmark detection feature of some other existing software tools, like the one presented in the work of Murphy et al.[12]. Then, the second step is to use this landmarking software to label corresponding landmarks on another phase image within the 4D data set (preferably on the other extreme phase for 4D pulmonary data sets). During this step, one of the core features of this landmarking software will assist the user to take full advantage of already labeled landmark positions and make the task of selecting further point-wise correspondence progressively easier. The software will provide more and more accurate suggested locations of the remaining landmarks for the user to consider as initial guesses. Finally, after all corresponding landmarks have been labeled on two phases within the 4D data set (preferably on the two extreme phases for pulmonary 4D data sets), another feature of the software tool will provide the user with suggested locations for all landmarks on all other phase images. And the user can then efficiently fine-tune the landmark locations on those other phases using the convenient 4D volume traversal capabilities of the software.

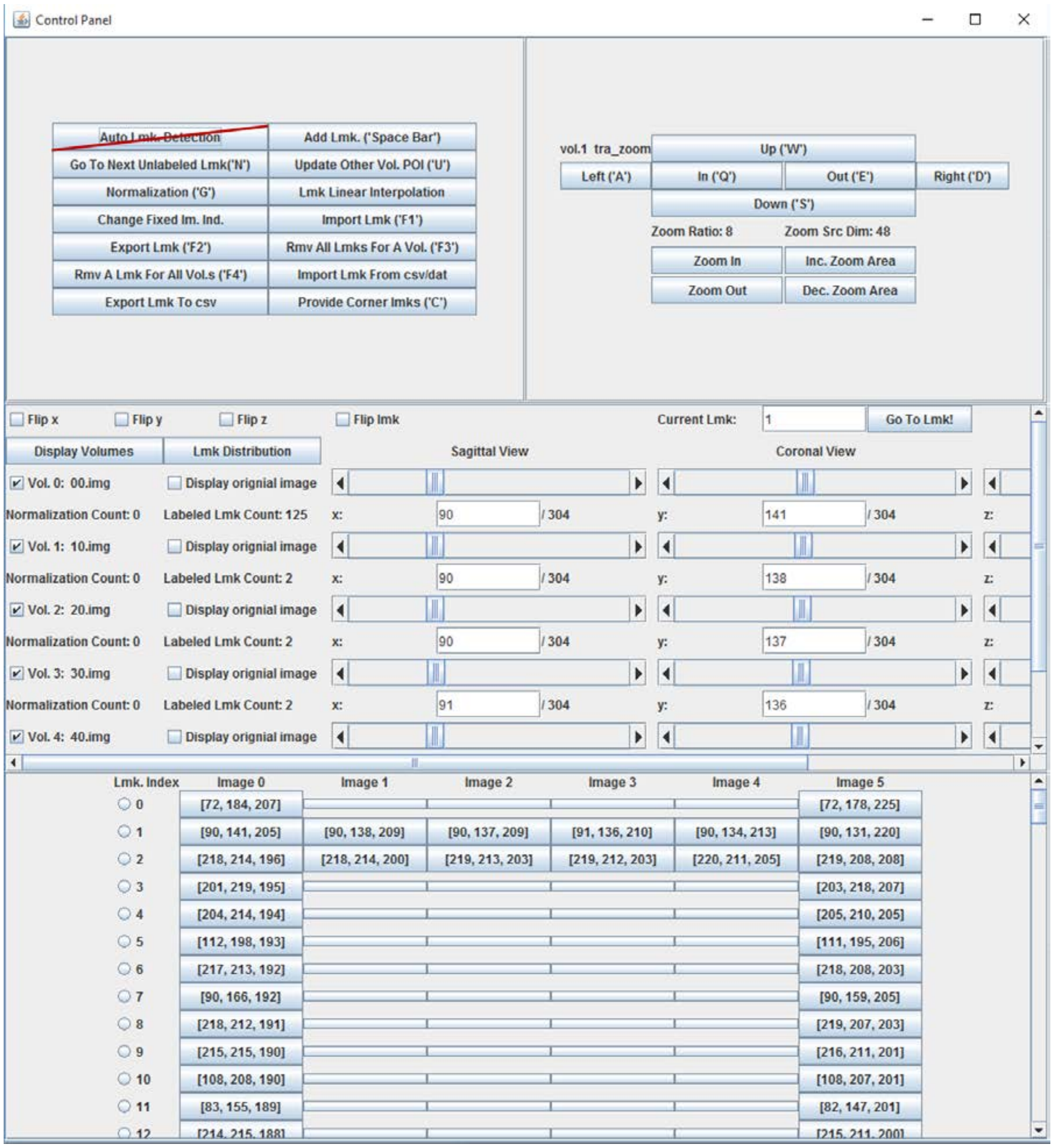
The main control GUI for the software is as shown in Fig 3.1. The user can perform actions such as “add landmark”, “remove landmark”, “import landmark from file”, “export landmark to file”, “perform normalization (registration)”, “perform landmark location linear interpolation” or “synchronize views in all 3D images” by clicking the corresponding buttons in the upper left section of the control panel. The user is able to set focus on any anatomical view (transverse, sagittal or coronal) of any displayed 3D phase images by either clicking the mouse on the view in the image display GUI or pressing the user-friendly keyboard shortcuts: “J” and “L” change focus to different anatomical views within the same 3D volume, “I” and “K” to traverse among different 3D volumes within the 4D data set in the same anatomical view. The current anatomical view that is focused on will be indicated by a red box around it. Within any chosen anatomical view of any 3D volume, the user can also effortlessly navigate through the 3D volume by either pressing the navigation buttons in the upper right section of the control GUI or, better yet, by pressing user-friendly keyboard shortcuts: “W” to move one voxel up, “S” to move one voxel down, “A” to move one voxel left, “D” to move one voxel right, “Q” to move one voxel in, “E” to move one voxel out. Other views of the same 3D volume will be updated correspondingly. In the middle part of the control panel, the user can choose which phase images to be shown in the image display GUI and flip the X, Y or Z orientation of images if necessary. And in the bottom part of the control GUI, all labeled landmarks are displayed. The user can easily visit any previously labeled landmark location by clicking the radio button to the left of the landmark. If the location of some landmark is not yet selected on a certain phase image, the user will be taken to its suggested location on that phase image. The image display GUI in which the standard and zoomed anatomical views of selected phase images are displayed is as shown in Fig 3.2.

During the landmark labeling process, the user can also easily check the distribution of previously labeled landmark locations, as shown in Figure 3.3, so that he/she adjust the locations of landmarks to make the overall distribution more uniform throughout the 4D data set.

3.3 Features and Mechanisms

Two essential mechanisms corresponding to the second and the third major steps of usage are at work within the landmarking tool.

The first mechanism kicks in after the user acquires the set of desired land-



Current Lmk: 1

✓ Vol. 0: 00.img	Display orignial image			
Normalization Count: 0	Labeled Lmk Count: 125	x: 90 / 304	y: 141 / 304	z:
✓ Vol. 1: 10.img	Display orignial image			
Normalization Count: 0	Labeled Lmk Count: 2	x: 90 / 304	y: 138 / 304	z:
✓ Vol. 2: 20.img	Display orignial image			
Normalization Count: 0	Labeled Lmk Count: 2	x: 90 / 304	y: 137 / 304	z:
✓ Vol. 3: 30.img	Display orignial image			
Normalization Count: 0	Labeled Lmk Count: 2	x: 91 / 304	y: 136 / 304	z:
✓ Vol. 4: 40.img	Display orignial image			

Lmk. Index	Image 0	Image 1	Image 2	Image 3	Image 4	Image 5
0	[72, 184, 207]					[72, 178, 225]
1	[90, 141, 205]	[90, 138, 209]	[90, 137, 209]	[91, 136, 210]	[90, 134, 213]	[90, 131, 220]
2	[218, 214, 196]	[218, 214, 200]	[219, 213, 203]	[219, 212, 203]	[220, 211, 205]	[219, 208, 208]
3	[201, 219, 195]					[203, 218, 207]
4	[204, 214, 194]					[205, 210, 205]
5	[112, 198, 193]					[111, 195, 206]
6	[217, 213, 192]					[218, 208, 203]
7	[90, 166, 192]					[90, 159, 205]
8	[218, 212, 191]					[219, 207, 203]
9	[215, 215, 190]					[216, 211, 201]
10	[108, 208, 190]					[108, 207, 201]
11	[83, 155, 189]					[82, 147, 201]
12	[214, 215, 188]					[215, 211, 200]

Figure 3.1: 4D landmarking software control panel. The first and last row are the extreme exhale (0%) and extreme inhale (100%) phase images. The rows in between are 20%, 40%, 60%, 80% inhalation phase images. The landmarks on 0% and 100% phase images have been labeled. Suggested landmark locations on the phases in between are presented through linear interpolation (the red markers at the centers of view for intermediate phases are the suggested landmark locations).

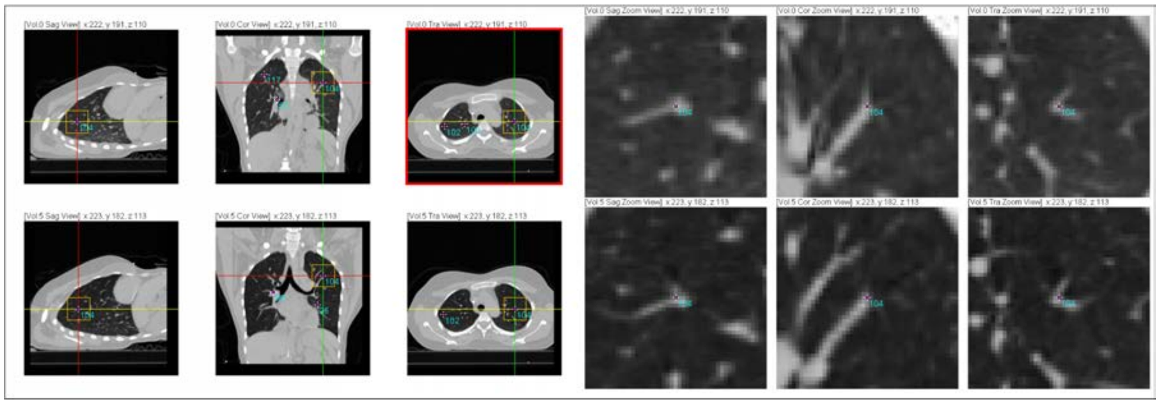


Figure 3.2: 4D landmarking software image display panel. The two rows of images illustrate the standard and zoomed versions of the three anatomical views of the extreme exhale and the extreme inhale phase images. The centers of view for both phase images are the corresponding locations of landmark No.104.

marks on one phase and starts to label corresponding landmarks on another phase (step 2 of usage). At the user's request, this feature will perform image registration between the pair of images based on corresponding landmarks that have recently been labeled by the user. The registration procedure will produce a deformed moving image that looks more similar to the fixed image, and the suggested landmark locations provided on the deformed moving image will also be more accurate thanks to the registration process. Keep repeating this process will make the registration and the suggested landmark locations increasingly more accurate, and thus drastically expedites the labeling process. The user will end up taking almost no effort to select further landmarks.

The specific type of image registration technique being used here is a vari-

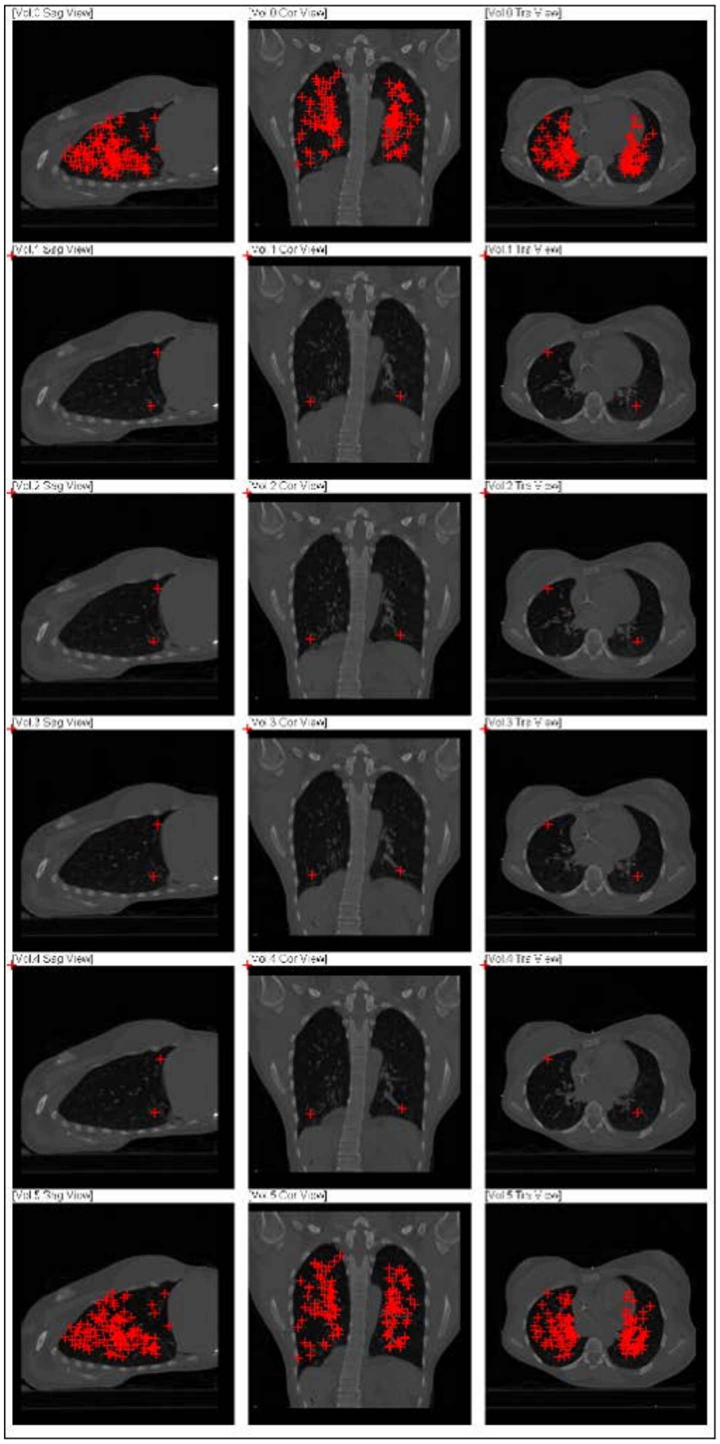


Figure 3.3: 4D landmarking software landmark distribution feature. The user can easily check the distribution of the previously labeled landmarks.

ant version of the thin plate spline registration[2], as discussed in the work of Joshi et al.[7]. Specifically, suppose the user has acquired the full set of landmarks $\mathbf{P}$ on the fixed image $I_f(\mathbf{x})$, and has labeled landmarks $\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_N$ on the moving image that correspond to the subset of landmarks $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\} \subset \mathbf{P}$ on the fixed image. Also denote $\mathbf{T} : \Omega_f \to \Omega_m, \mathbf{x} \mapsto \mathbf{T}(\mathbf{x})$ as the spatial transform deforming the moving image to the fixed image, which is to be estimated based on the corresponding landmarks. The estimated transform $\mathbf{T}$ will consist of two components: one affine transform $\mathbf{T}_{affine}$ and another transform $\mathbf{T}_{spline}$ induced by spline interpolation from the displacements of the landmarks $\{\mathbf{p}_i\}_{i=1}^{N}$. Then, $\forall \mathbf{x} \in \Omega_f$, $\mathbf{T}(\mathbf{x}) \in \Omega_m$ would have expression as shown in Equ. 3.1.

$$\begin{aligned}
\mathbf{T}(\mathbf{x}) = \mathbf{T}_{affine}(\mathbf{x}) + \mathbf{T}_{spline}(\mathbf{x}) &= \mathbf{a}_1 x_1 + \mathbf{a}_2 x_2 + \mathbf{a}_3 x_3 + \mathbf{a}_4 + \sum_{i=1}^{N} \mathbf{b}_i \; r_i(\mathbf{x}) \\
\begin{bmatrix} T_1(\mathbf{x}) \\ T_2(\mathbf{x}) \\ T_3(\mathbf{x}) \end{bmatrix} &= \begin{bmatrix} a_{11}x_1 + a_{21}x_2 + a_{31}x_3 + a_{41} + b_{11}r_1(\mathbf{x}) + ... + b_{N1}r_N(\mathbf{x}) \\ a_{12}x_1 + a_{22}x_2 + a_{32}x_3 + a_{42} + b_{12}r_1(\mathbf{x}) + ... + b_{N2}r_N(\mathbf{x}) \\ a_{13}x_1 + a_{23}x_2 + a_{33}x_3 + a_{43} + b_{13}r_1(\mathbf{x}) + ... + b_{N3}r_N(\mathbf{x}) \end{bmatrix} \\
&= \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} & b_{11} & ... & b_{N1} \\ a_{12} & a_{22} & a_{32} & a_{42} & b_{12} & ... & b_{N2} \\ a_{13} & a_{23} & a_{33} & a_{43} & b_{13} & ... & b_{N3} \end{bmatrix} \cdot (x_1, x_2, x_3, 1, r_1(\mathbf{x}), ..., r_N(\mathbf{x}))^T \\
&= \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} \\ a_{12} & a_{22} & a_{32} & a_{42} \\ a_{13} & a_{23} & a_{33} & a_{43} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} + \begin{bmatrix} b_{11} & ... & b_{N1} \\ b_{12} & ... & b_{N2} \\ b_{13} & ... & b_{N3} \end{bmatrix} \cdot \begin{bmatrix} r_1(\mathbf{x}) \\ \vdots \\ r_N(\mathbf{x}) \end{bmatrix} \qquad (3.1)
\end{aligned}$$

where $r : [0, +\infty) \to \mathbb{R}, ||\mathbf{x} - \mathbf{p}_i|| \mapsto r(||\mathbf{x} - \mathbf{p}_i||) \triangleq r_i(\mathbf{x})$ is the radial basis function that depends on the distance between any point $\mathbf{x}$ and the ith landmark $\mathbf{p}_i$ in the fixed image domain. The functional form of r indicates how the displacement at $\mathbf{x}$ is influenced by the displacement at $\mathbf{p}_i$ through spline interpolation. This function will determine the interpolation behavior of the displacements of points around a certain landmark. $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4 \in \mathbb{R}^3$ are vectors containing the affine transform parameters, while $\mathbf{b}_1$, $\mathbf{b}_2$, ..., $\mathbf{b}_N \in \mathbb{R}^3$ are the weights of the spline interpolation transform. Altogether, the transform $\mathbf{T}$ is parametrized by $12 + 3N$ parameters, and we can estimate these parameters using the coordinates of the corresponding landmarks $\{\mathbf{p}_i\}_{i=1}^N$ and $\{\mathbf{q}_i\}_{i=1}^N$ as shown in Equ. 3.2 and 3.3.

$$\begin{bmatrix} Q \\ 0_{4\times3} \end{bmatrix} = \begin{bmatrix} \bar{P} & R \\ 0_{4\times4} & \bar{P} \end{bmatrix} \cdot \begin{bmatrix} A \\ B \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_N^T \\ 0_{4\times3} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{p}}_1^T & r_{1,1} & r_{1,2} & \cdots & r_{1,N} \\ \bar{\mathbf{p}}_2^T & r_{2,1} & r_{2,2} & \cdots & r_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{\mathbf{p}}_N^T & r_{N,1} & r_{N,2} & \cdots & r_{N,N} \\ 0_{4\times4} & \bar{\mathbf{p}}_1 & \bar{\mathbf{p}}_2 & \dots & \bar{\mathbf{p}}_N \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \mathbf{a}_3^T \\ \mathbf{a}_4^T \\ \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_N^T \end{bmatrix}$$

$$\begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,3} \\ q_{2,1} & q_{2,2} & q_{2,3} \\ \vdots & \vdots & \vdots \\ q_{N,1} & q_{N,2} & q_{N,3} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & 1 & r_{1,1} & r_{1,2} & \cdots & r_{1,N} \\ p_{2,1} & p_{2,2} & p_{2,3} & 1 & r_{2,1} & r_{2,2} & \cdots & r_{2,N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{N,1} & p_{N,2} & p_{N,3} & 1 & r_{N,1} & r_{N,2} & \cdots & r_{N,N} \\ 0 & 0 & 0 & 0 & p_{1,1} & p_{2,1} & \dots & p_{N,1} \\ 0 & 0 & 0 & 0 & p_{1,2} & p_{2,2} & \dots & p_{N,2} \\ 0 & 0 & 0 & 0 & p_{1,3} & p_{2,3} & \dots & p_{N,3} \\ 0 & 0 & 0 & 0 & 1 & 1 & \dots & 1 \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \\ b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ \vdots & \vdots & \vdots \\ b_{N1} & b_{N2} & b_{N3} \end{bmatrix} \tag{3.2}$$

Taking the inverse of Equ. 3.2, we have:

$$\begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \bar{P} & R \\ 0_{4\times4} & \bar{P} \end{bmatrix}^{-1} \cdot \begin{bmatrix} Q \\ 0_{4\times3} \end{bmatrix} \tag{3.3}$$

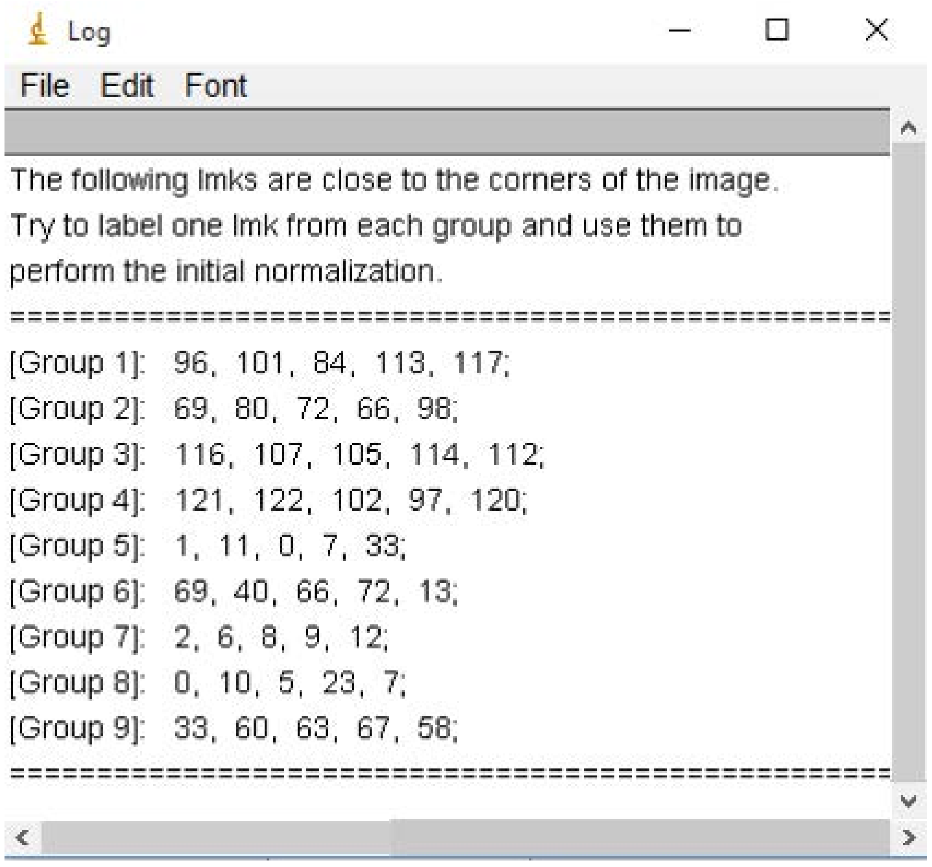
where $\bar{\mathbf{p}}_i = (p_{i,1}, p_{i,2}, p_{i,3}, 1)^T$ is the augmented coordinates of landmark i in the fixed image. $||\mathbf{p}_i - \mathbf{p}_j||$ is the distance between landmarks i and j in the fixed image, and $r_{i,j} = r(||\mathbf{p}_i - \mathbf{p}_j||)$ is the radial basis function centered at landmark $\mathbf{p}_j$. Indeed, we would want the matrix inversion in Equ. 3.3 to be valid, and what's more, we want the initial normalization to roughly align the moving and the fixed image globally. Therefore, we prefer that the user begins by labeling corresponding landmarks on the (deformed) moving image that are scattered and close to the corners/boundaries of the foreground of the image, so that the estimated transform obtained from the initial normalization would not only exist but also make the moving image to match the fixed image globally, which will help the user to label more landmarks with less effort. Under such consideration, we implemented a small functionality in the software tool that provides the user with recommended indices of landmarks to label, such that these landmarks are scattered and close to the corners/boundaries of the foreground of the image. This functionality is shown in Figure 3.4, where the user can pick one or more landmark indices from each group to label, and the estimated transform will exist and be global.

The radial basis function we used is the one proposed by Joshi et al.[7] and is given as in Equ. 3.4.

$$r(||\mathbf{x}-\mathbf{p}_i||) = \sqrt{\frac{2}{\pi c}}e^{-\sqrt{c}||\mathbf{x}-\mathbf{p}_i||} \tag{3.4}$$

where c is a positive constant. In the implementation of this software, we took $c = \frac{\pi}{2} \approx 0.64$ so that $r(||\mathbf{x}-\mathbf{p}_i||) \approx e^{-0.8||\mathbf{x}-\mathbf{p}_i||}$.

The reason of choosing this radial basis instead of the common thin plate spline radial basis in 3D, $r(||\mathbf{x}-\mathbf{p}_i||) = ||\mathbf{x}-\mathbf{p}_i||$, is because this radial basis is decaying exponentially. Therefore, the displacements of points outside of certain distance from $\mathbf{p}_i$ will not be influenced by the displacement of $\mathbf{p}_i$ due to spline interpolation. This



```
The following lmks are close to the corners of the image.
Try to label one lmk from each group and use them to
perform the initial normalization.
=====================================================
[Group 1]:  96, 101, 84, 113, 117;
[Group 2]:  69, 80, 72, 66, 98;
[Group 3]:  116, 107, 105, 114, 112;
[Group 4]:  121, 122, 102, 97, 120;
[Group 5]:  1, 11, 0, 7, 33;
[Group 6]:  69, 40, 66, 72, 13;
[Group 7]:  2, 6, 8, 9, 12;
[Group 8]:  0, 10, 5, 23, 7;
[Group 9]:  33, 60, 63, 67, 58;
=====================================================
```

Figure 3.4: 4D landmarking software corner landmark detection feature. The software tool provides the user with scattered and close to boundary landmark indices to label. And the user can in turn perform initial normalization based on these landmarks.

choice of radial basis will tremendously shorten the computation time spent in the normalization process, especially when the number of labeled landmarks is large. If we used the thin plate spline radial basis function $r(||\mathbf{x}-\mathbf{p}_i||) = ||\mathbf{x}-\mathbf{p}_i||$, then during computing the displacement of any point $\mathbf{x}$ in the fixed image, we need to calculate every $r(||\mathbf{x}-\mathbf{p}_i||)$ term in Equ.3.2, i.e., take into account the influence of every labeled landmark $\mathbf{p}_i$ through spline interpolation. By using this variant version of the thin plate spline with radial basis function as given in Equ. 3.4, we only computed the terms $r(||\mathbf{x}-\mathbf{p}_i||) = \sqrt{\frac{2}{\pi c}}e^{-\sqrt{c}||\mathbf{x}-\mathbf{p}_i||}$ for which the landmark $\mathbf{p}_i$ is close enough to the point $\mathbf{x}$. Specifically, we can stipulate that once the radial basis function value is less than 10^{-3}, i.e., $e^{-0.8||\mathbf{x}-\mathbf{p}_i||} \leq 10^{-3} \iff ||\mathbf{x}-\mathbf{p}_i|| \geq 8.6$, then we would not consider the influence of $\mathbf{p}_i$ on the displacement of $\mathbf{x}$ through spline interpolation.

The second important mechanism of this software tool is the landmark location linear interpolation feature. After the user acquires the full set of landmarks on one phase and finishes labeling the corresponding landmarks on another phase, he/she can use this mechanism to generate meaningful suggested locations for all landmarks on all the other phases. The user does not have to perform any more normalization procedures for any of the other phase images. The effectiveness of this feature is best illustrated in Figure 3.6 as compared to Figure 3.5. Without the linear interpolation feature to provide the user with meaningful suggested locations of all landmarks on all intermediate phases, the user will take some extra endeavor to label the corresponding landmark locations on these intermediate phases, as seen in Figure 3.5. However, after using the landmark location linear interpolation feature, the user will only need very small amount of effort to label the landmarks on these intermediate phases based on the initial guesses, as shown in Figure 3.6. Note in Figure 3.5 and 3.6, the first and last rows represent the two extreme phases, 0% inhale and 100% inhale. The second through fifth rows show the 20% inhale, 40% inhale, 60% inhale, and 80% inhale phase images. This feature works in the same way for intermediate exhale phase images.

Combining the above two mechanisms, the user can label landmarks on 4DCT pulmonary data sets a lot more efficiently compared to using other existing tools for landmarking. The recommended procedure would be that the user first obtain the full set of landmarks on the extreme exhale/inhale phase image, then use normalization mechanism to label corresponding landmarks on the extreme inhale/exhale phase image, and finally use the landmark location interpolation mechanism to generate meaningful suggested landmark locations on other intermediate phases and slightly adjust the landmark locations based on the initial guesses. Figures 3.7 - 3.12 illustrated the mechanisms behind this software in great detail.

3.4 Experiments and Results

Landmark labeling is an intrinsically subjective task. Since the image is a discrete representation of the actual continuous object being imaged, truly corresponding points may not even be present in the image. Observer variability is thus unavoidable and there is no “absolute truth” to compare to.

Therefore, we carried out independent inter-observer landmark labeling experiments to test the landmarking software’s reliability and repeatability. Specifically,

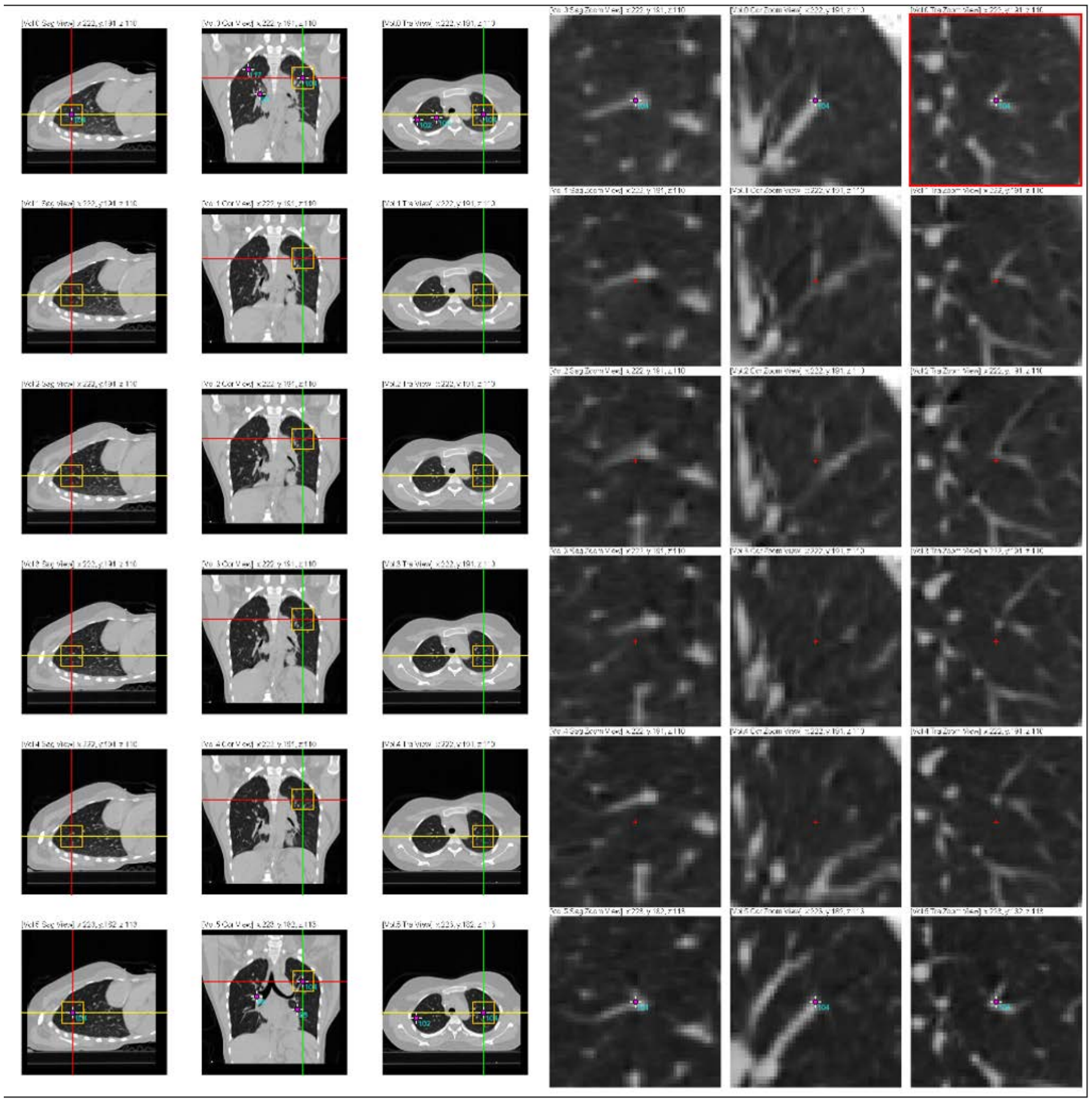


Figure 3.5: Suggested landmark locations on intermediate phases without linear interpolation feature. After the user has labeled corresponding landmarks on two extreme phases, if the user does not use landmark location interpolation feature, the suggested landmark locations for the other intermediate phases will all be the same as the coordinates of landmarks on one of the two already labeled extreme phases. And thus, the user will take some extra endeavor to find proper positions of landmarks on these phases.

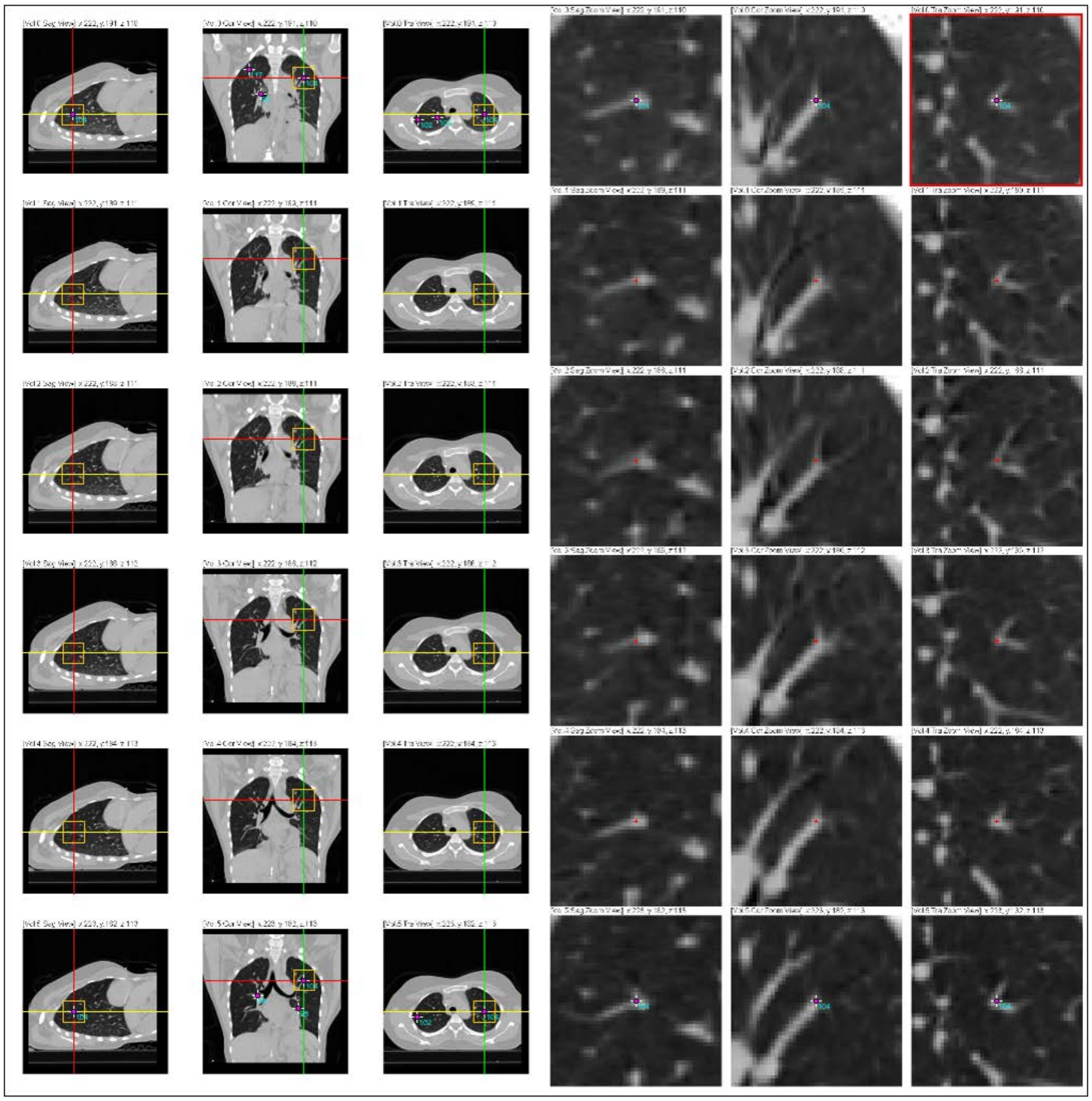
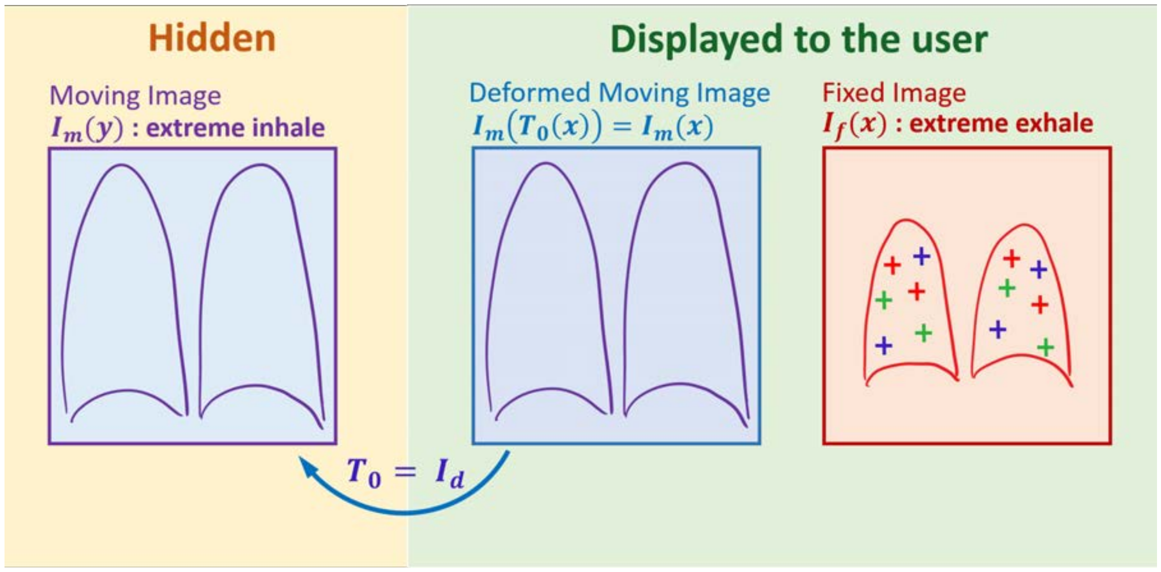
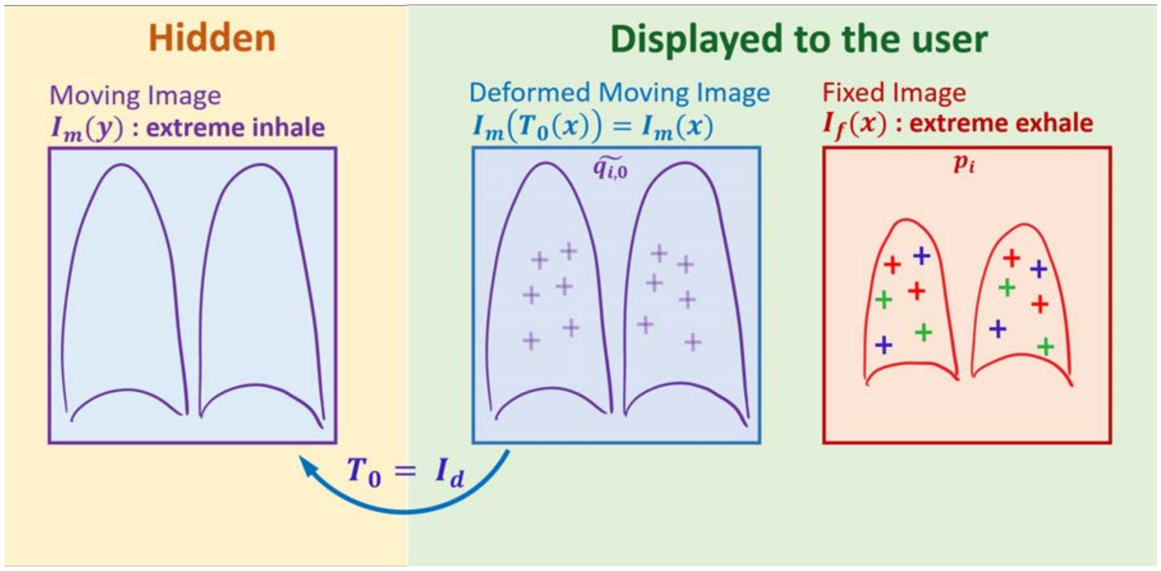


Figure 3.6: Suggested landmark locations on intermediate phases with linear interpolation feature. After the user has labeled corresponding landmarks on two extreme phases, if the user chooses to use the landmark location interpolation feature, the software tool will provide much more meaningful suggested locations for all landmarks on the other intermediate phases. And thus, the user can take very small amount of effort to pin-point the proper locations of landmarks on these phases.

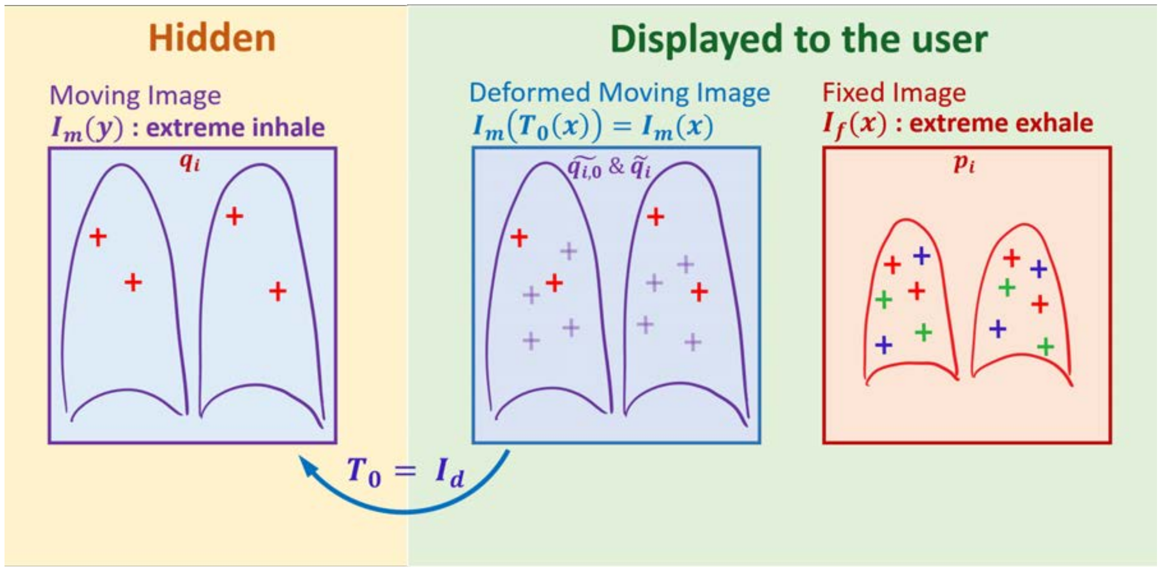


(a) The user acquires a set of desired landmarks $\mathbf{p}_i$ on the fixed image $I_f(\mathbf{x})$. The software uses transform $\mathbf{T}_0 = I_d$ to generate deformed moving image $I_m(\mathbf{T}_0(\mathbf{x})) = I_m(\mathbf{x})$.

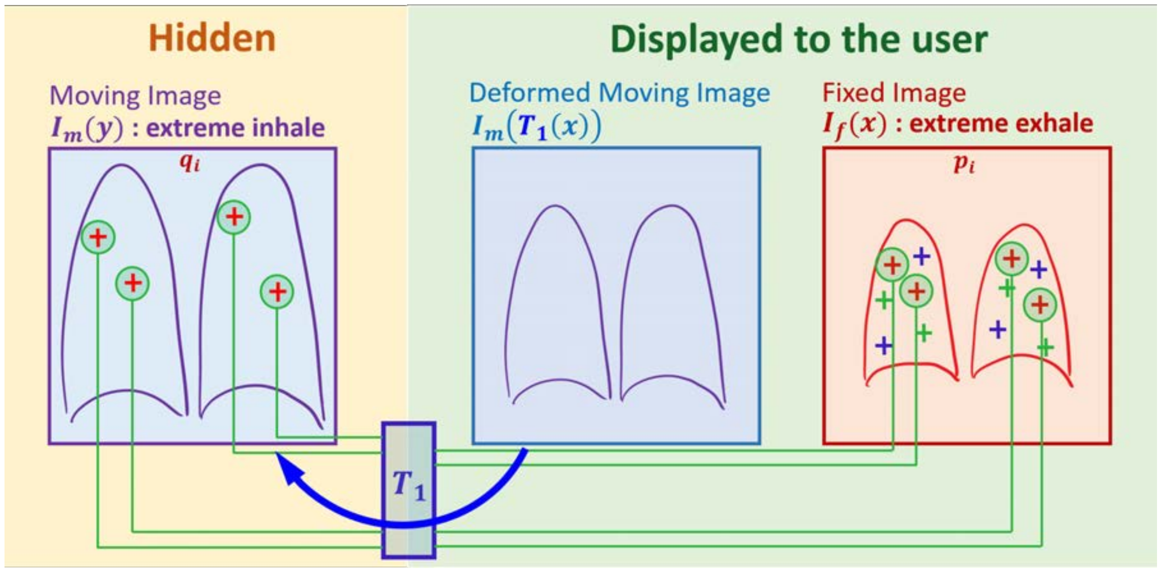


(b) The software provides suggested landmark locations $\tilde{\mathbf{q}}_{i,0} = \mathbf{p}_i$ on the deformed moving image, as initial guesses for the user.

Figure 3.7: Illustration of the mechanism of the 4D landmarking software: 1 - 2.

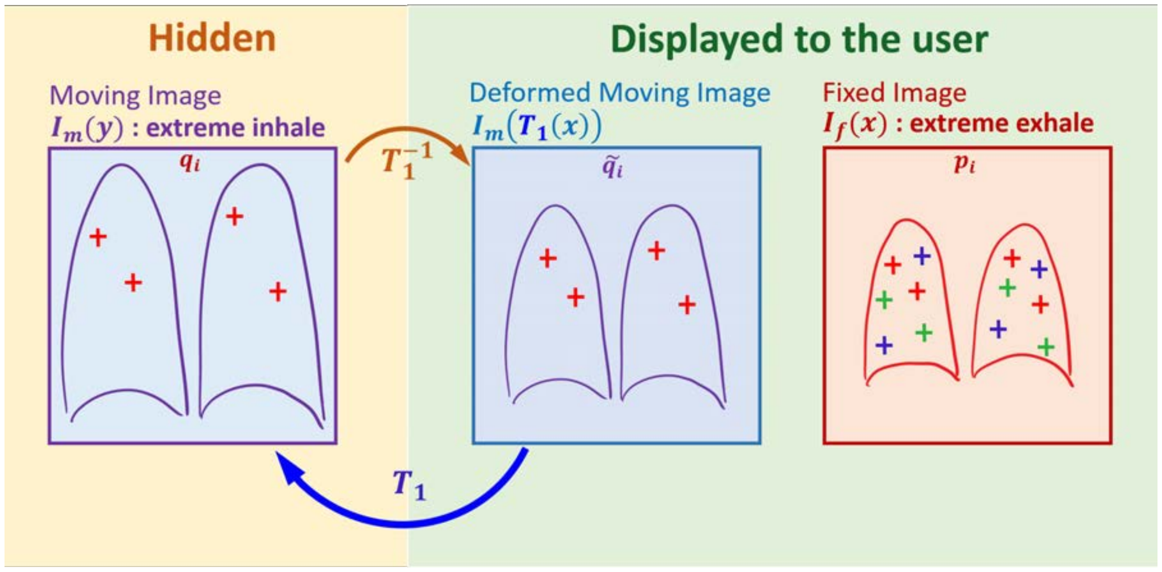


(a) The user labels landmarks $\tilde{\mathbf{q}}_i$ the deformed moving image, based on the initial guesses $\tilde{\mathbf{q}}_{i,0}$. The software pushes the user-labeled landmarks $\tilde{\mathbf{q}}_i$ from the deformed moving image to $\mathbf{q}_i = \mathbf{T}_0(\tilde{\mathbf{q}}_i)$ on the moving image.

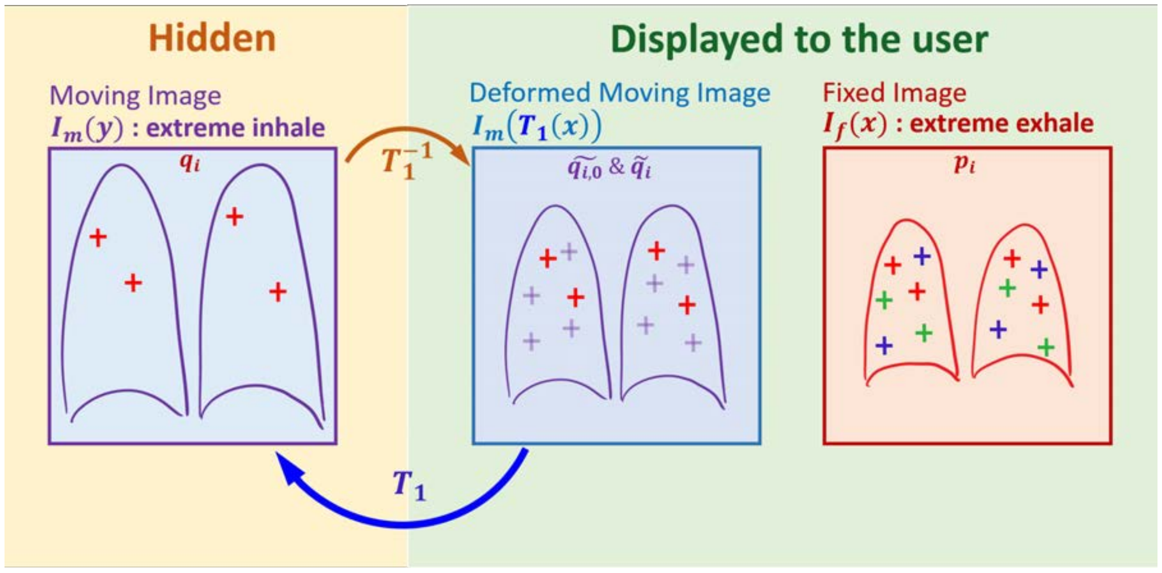


(b) The software estimates transform $\mathbf{T}_1$ based on corresponding landmarks $\mathbf{q}_i$ and $\mathbf{p}_i$ on the moving and the fixed images. The software generates the deformed moving image $I_m(\mathbf{T}_1(\mathbf{x}))$ resulting from transform $\mathbf{T}_1$.

Figure 3.8: Illustration of the mechanism of the 4D landmarking software: 3 - 4.

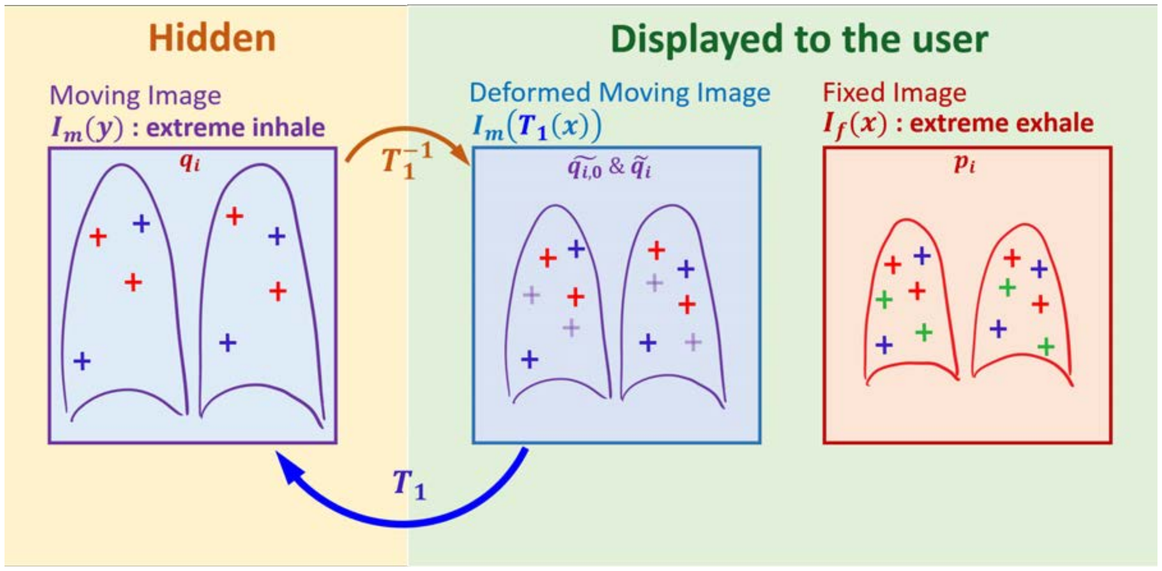


(a) The software estimates the inverse transform $\mathbf{T}_1^{-1}$ from $\mathbf{T}_1$. The software pushes previously labeled landmarks $\mathbf{q}_i$ from the moving image to $\tilde{\mathbf{q}}_i = \mathbf{T}_1^{-1}(\mathbf{q}_i)$ on the deformed moving image.

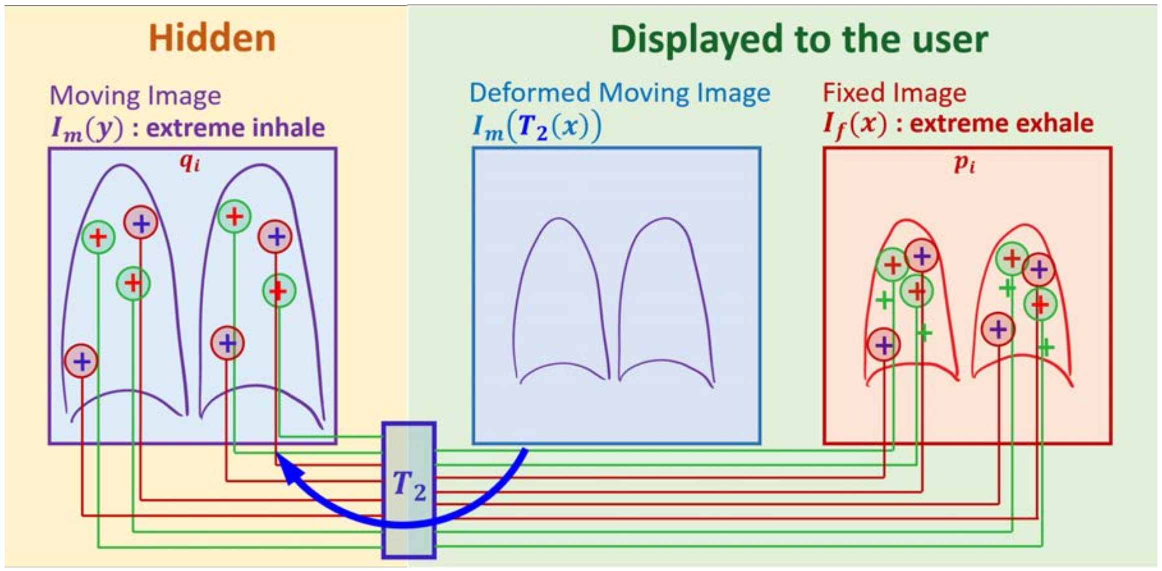


(b) The software provides suggested landmark locations $\tilde{\mathbf{q}}_{i,0} = \mathbf{p}_i$ for the remaining landmarks on the deformed moving image.

Figure 3.9: Illustration of the mechanism of the 4D landmarking software: 5 - 6.

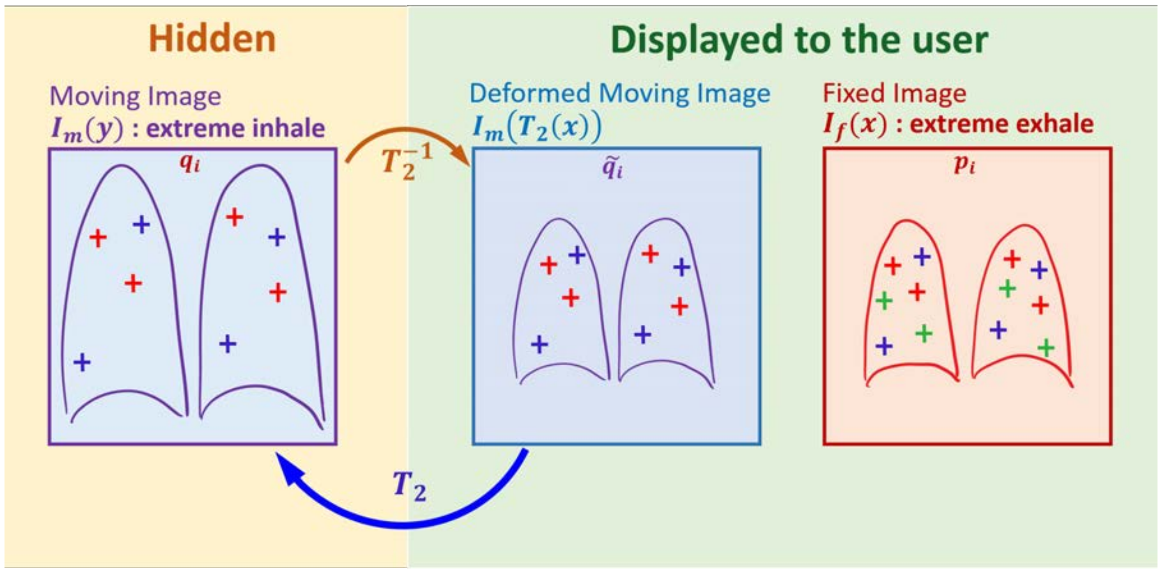


(a) The user labels more landmarks $\tilde{\mathbf{q}}_i$ on the deformed moving image, based on the initial guesses $\tilde{\mathbf{q}}_{i,0}$. The software pushes newly labeled landmarks $\tilde{\mathbf{q}}_i$ from the deformed moving image to $\mathbf{q}_i = \mathbf{T}_1(\tilde{\mathbf{q}}_i)$ on the moving image.

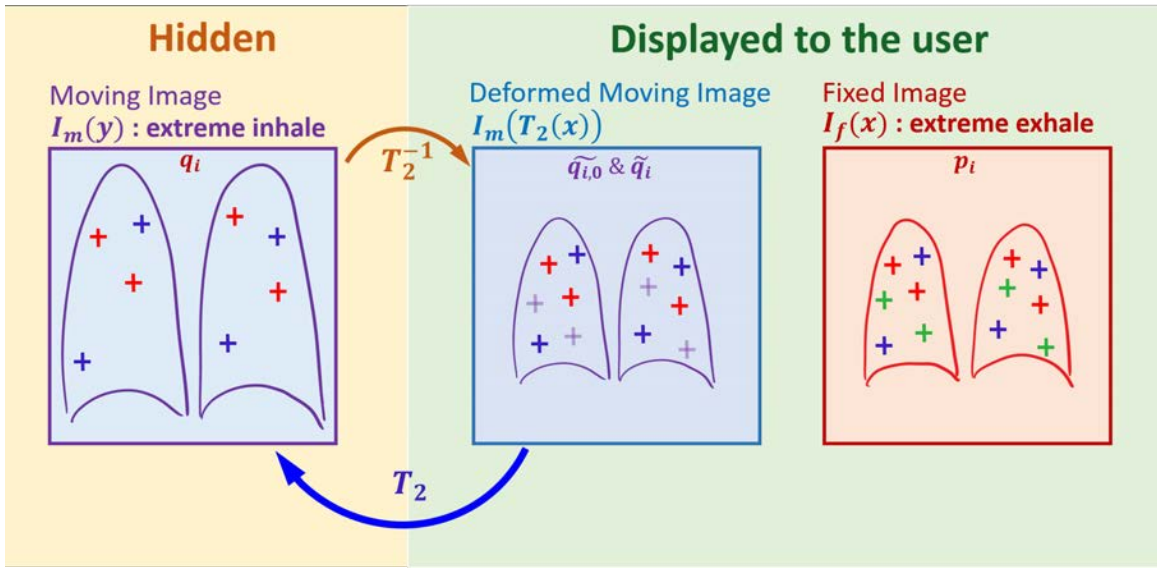


(b) The software estimates transform $\mathbf{T}_2$ based on corresponding landmarks $\mathbf{q}_i$ and $\mathbf{p}_i$ on the moving and the fixed images. The software generates the deformed moving image $I_m(\mathbf{T}_2(\mathbf{x}))$ resulting from transform $\mathbf{T}_2$.

Figure 3.10: Illustration of the mechanism of the 4D landmarking software: 7 - 8.

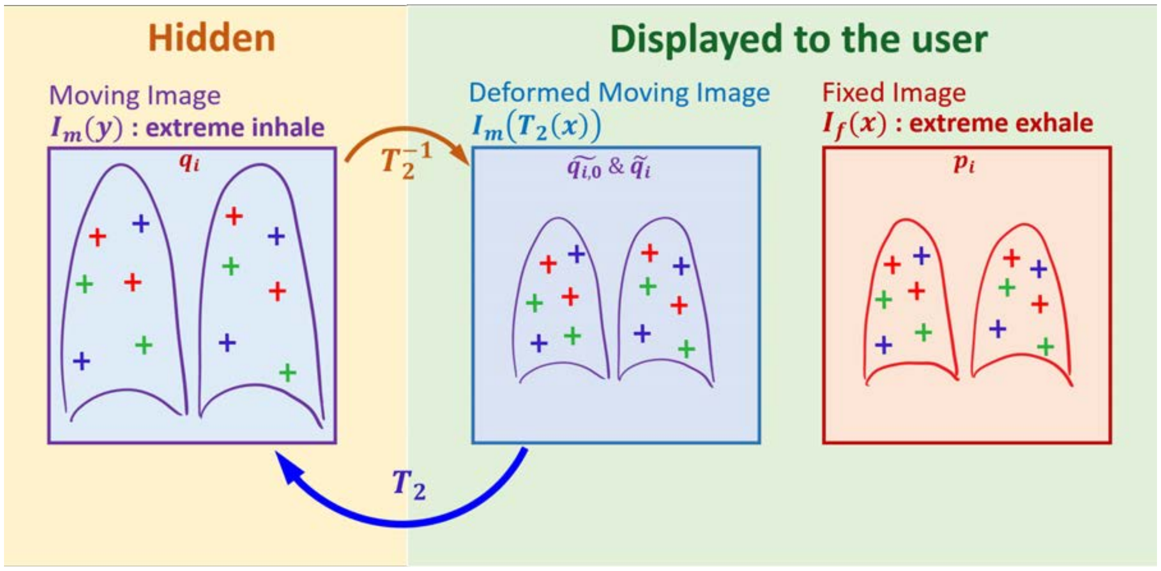


(a) The software estimates the inverse transform $\mathbf{T}_2^{-1}$ from $\mathbf{T}_2$. The software pushes previously labeled landmarks $\mathbf{q}_i$ from the moving image to $\tilde{\mathbf{q}}_i = \mathbf{T}_2^{-1}(\mathbf{q}_i)$ on the deformed moving image.

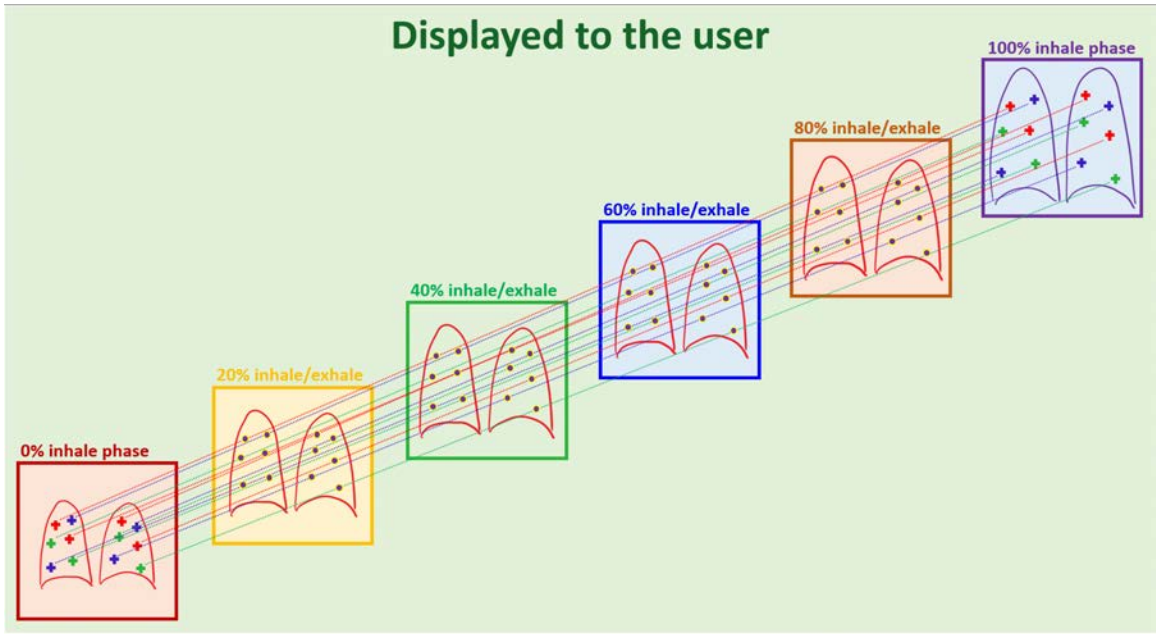


(b) The software provides suggested landmark locations $\tilde{\mathbf{q}}_{i,0} = \mathbf{p}_i$ for the remaining landmarks on the deformed moving image.

Figure 3.11: Illustration of the mechanism of the 4D landmarking software: 9 - 10.



(a) The user labels more landmarks $\tilde{\mathbf{q}}_i$ on the deformed moving image, based on the initial guesses $\tilde{\mathbf{q}}_{i,0}$. The software pushes newly labeled landmarks $\tilde{\mathbf{q}}_i$ from the deformed moving image to $\mathbf{q}_i = \mathbf{T}_1(\tilde{\mathbf{q}}_i)$ on the moving image.



(b) All corresponding landmarks on the moving and the fixed images (two extreme phases) have been labeled. The software uses linear interpolation to provide suggested landmark locations for all landmarks on all intermediate inhalation/exhalation phases.

Figure 3.12: Illustration of the mechanism of the 4D landmarking software: 11 - 12.

we used a total of five re-sampled 4DCT pulmonary data sets: PFS-001, PFS-002, PFS-004, PFS-007, PFS-008, all with image dimension 304 x 304 x 320 x 10 and voxel physical size 1mm x 1mm x 1mm. Five observers in the project group were trained on how to use the 4D landmarking software and asked them to label 123, 109, 123, 117, and 110 numbers of landmarks on the five 4DCT data sets based on the locations of automatically detected landmarks on the extreme inhale phase using iX software [12]. Due to the time consuming nature of the landmarking task, only one observer used the full functionalities of the 4D landmarking software and labeled 4D landmarks on every breathing phase for two out of the five 4DCT data sets. But all the five observers finished labeling corresponding landmarks on the extreme exhale phase within the 4DCT data sets using the normalization feature. Therefore the analysis of the landmarking results is based on the locations of landmarks on the extreme exhale phases labeled by the 5 observers.

For each landmark in the extreme exhale phase of every data set, we have a constellation of 5 landmark locations selected by the 5 observers, from which a mean landmark location can be computed. This mean landmark location will be regarded as the "true" location for this landmark. Then, for each observer, we can calculate the Euclidean distance from each of his landmark location to the corresponding mean location, call this distance the "landmark variation" and treat these distances as a set of independent identically distributed random variables from that observer's perspective. Thus, for each data set and for each observer, we can compute the sample mean and standard deviation of his/her landmark variation, and use these values as a qualitative measure of the repeatability and reliability of the 4D landmarking software.

Clearly, for each landmark within each data set, the locations labeled by every observer will have non-trivial influence on the overall analysis. For example, if 1 observer was inexperienced with the lung anatomy at first or grew tired, impatient or careless during the tedious landmark labeling process, he/she might end up labeling a few, if not many, inaccurate landmark locations within a certain data set. Consequently, the calculated mean landmark locations in that data set (which we shall use as the “true” landmark locations) will inevitably be “contaminated” by this observer’s results and the mean and standard deviation of the landmark variation of the other 4 observers will become larger even if the 4 of them reached relatively good consensus for most of the landmark locations. This phenomenon does not occur too often, but whenever the landmark variation of more than 40 landmarks labeled by a certain observer in a data set has landmark variation of more than 2.5 mm, we would exclude that observer’s landmark results for that data set and only use the results from the remaining observers for the analysis.

Figures 3.13 - 3.17 illustrate the projected locations onto the coronal view of landmarks labeled by the observers. The mean landmarks locations are represented by red crosses, while the individual observer-labeled landmarks are shown as small circles of different colors. There is a disk surrounding each mean landmark location, whose radius is the maximum value of landmark variation among all observers for that landmark. Figures 3.18 - 3.22 show the histograms of landmark variation for

Landmark Variation	User 1	User 2	User 3	User 4	User 5
PFS-001	0.70±0.31	0.72±0.36	**Excluded**	0.73±0.32	0.78±0.42
PFS-002	0.63±0.34	0.75±0.39	0.70 ±0.31	0.69±0.34	0.82±0.39
PFS-004	0.73±0.46	0.82±0.45	0.90±0.51	0.87±0.49	0.96±0.52
PFS-007	0.65±0.42	0.73±0.36	0.65±0.34	**Excluded**	0.68±0.37
PFS-008	0.64±0.34	0.87±0.47	0.82±0.47	0.68±0.34	0.78±0.40

Table 3.1: Sample mean and standard deviation of landmark variation for all observers on all data sets. The format is Mean ± Standard Deviation. The unit for all numbers in the table is mm. The landmark results of User 3 and User 4 were excluded from the analysis of data set PFS-001 and PFS-007, respectively. This is because if they were included, then more than 40 of their landmark variation values would be bigger than 2.5 mm.

each observer on all data sets. Table 3.1 demonstrates the sample mean, standard deviation and maximum sample value of landmark variation for each observer on all data sets.

In Fig.3.13-3.17, the small circles tend to concentrate relatively tightly around the red cross for most landmarks, especially for data sets PFS-002 and PFS-007. This indicates that the independent observers reached consensus on the location of most landmarks. From Table 3.1, we can see that, on average, each individual observer (excluding the outliers) achieves sub-voxel accuracy with respect to the “true” landmark locations when using the 4D landmarking software. We also need to keep in

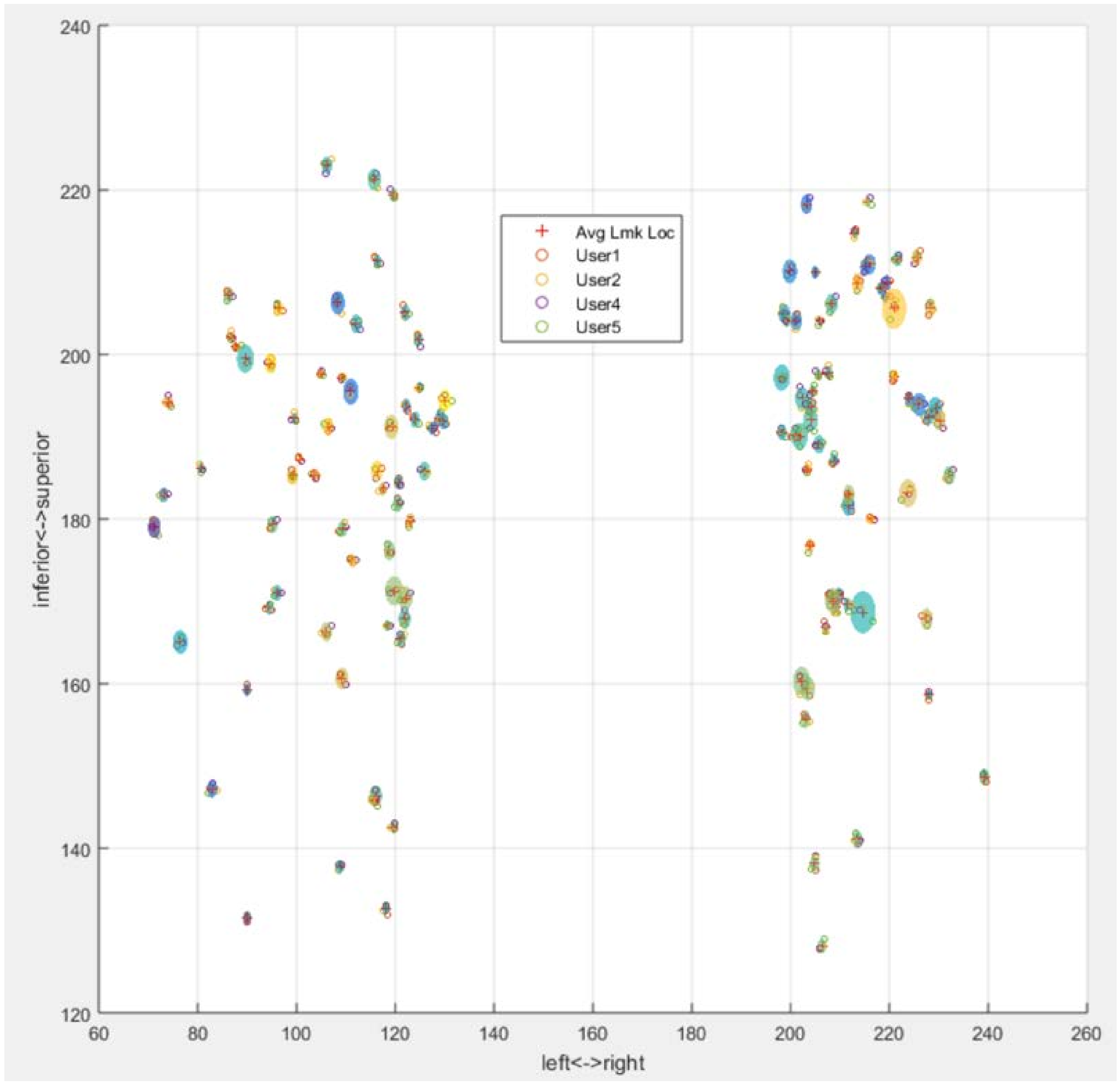


Figure 3.13: Landmarks labeled by 4 observers on the extreme inhale phase of PFS-001. Landmarks were projected onto coronal plane. The mean landmarks locations are represented by red crosses, while the individual observer-labeled landmarks are shown as small circles of different colors. There is a disk surrounding each mean landmark location, whose radius is the maximum landmark variation among all observers for that landmark.

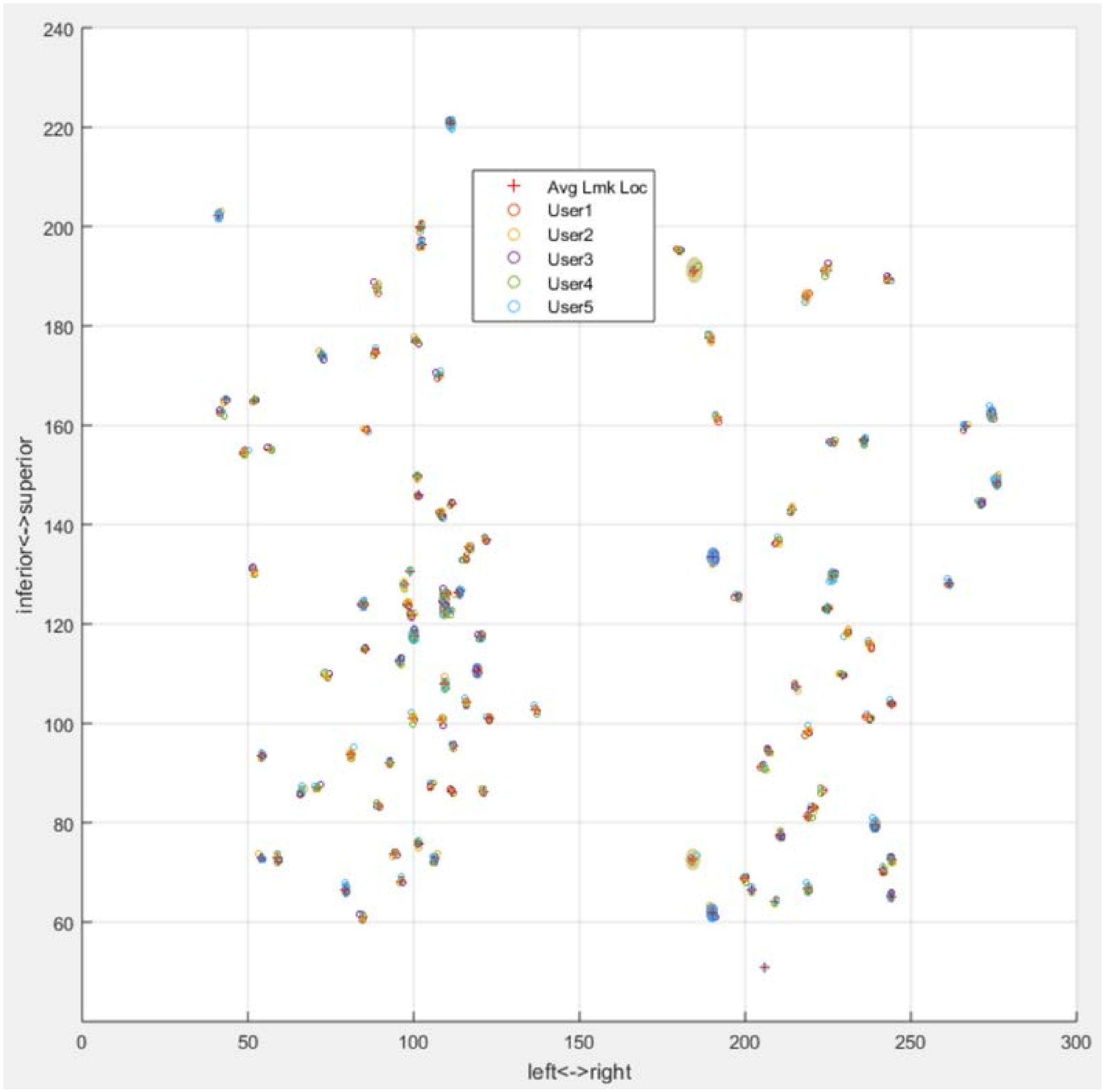


Figure 3.14: Landmarks labeled by 5 observers on the extreme inhale phase of PFS-002. Landmarks were projected onto coronal plane. The mean landmarks locations are represented by red crosses, while the individual observer-labeled landmarks are shown as small circles of different colors. There is a disk surrounding each mean landmark location, whose radius is the maximum landmark variation among all observers for that landmark.

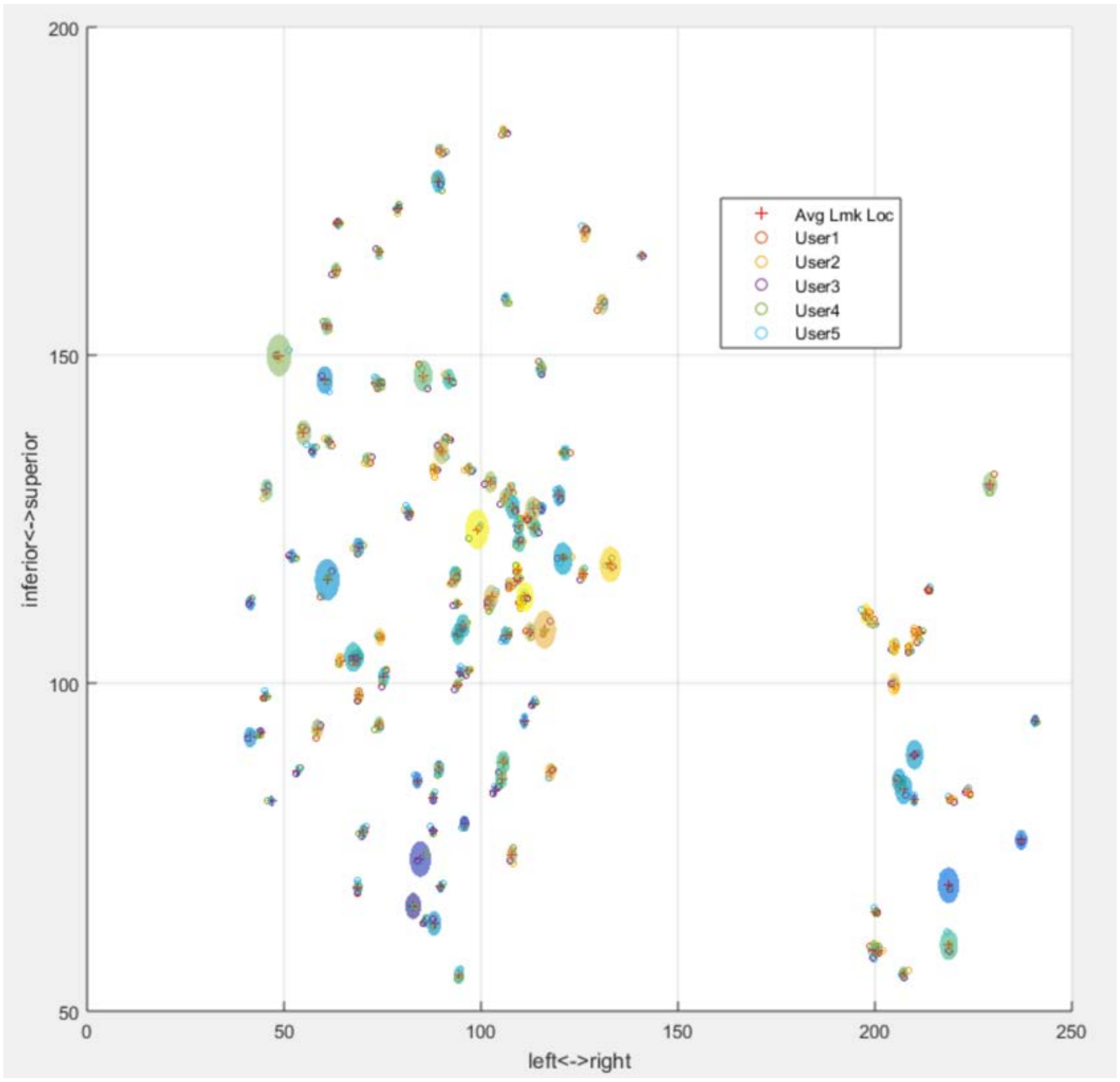


Figure 3.15: Landmarks labeled by 5 observers on the extreme inhale phase of PFS-004. Landmarks were projected onto coronal plane. The mean landmarks locations are represented by red crosses, while the individual observer-labeled landmarks are shown as small circles of different colors. There is a disk surrounding each mean landmark location, whose radius is the maximum landmark variation among all observers for that landmark.

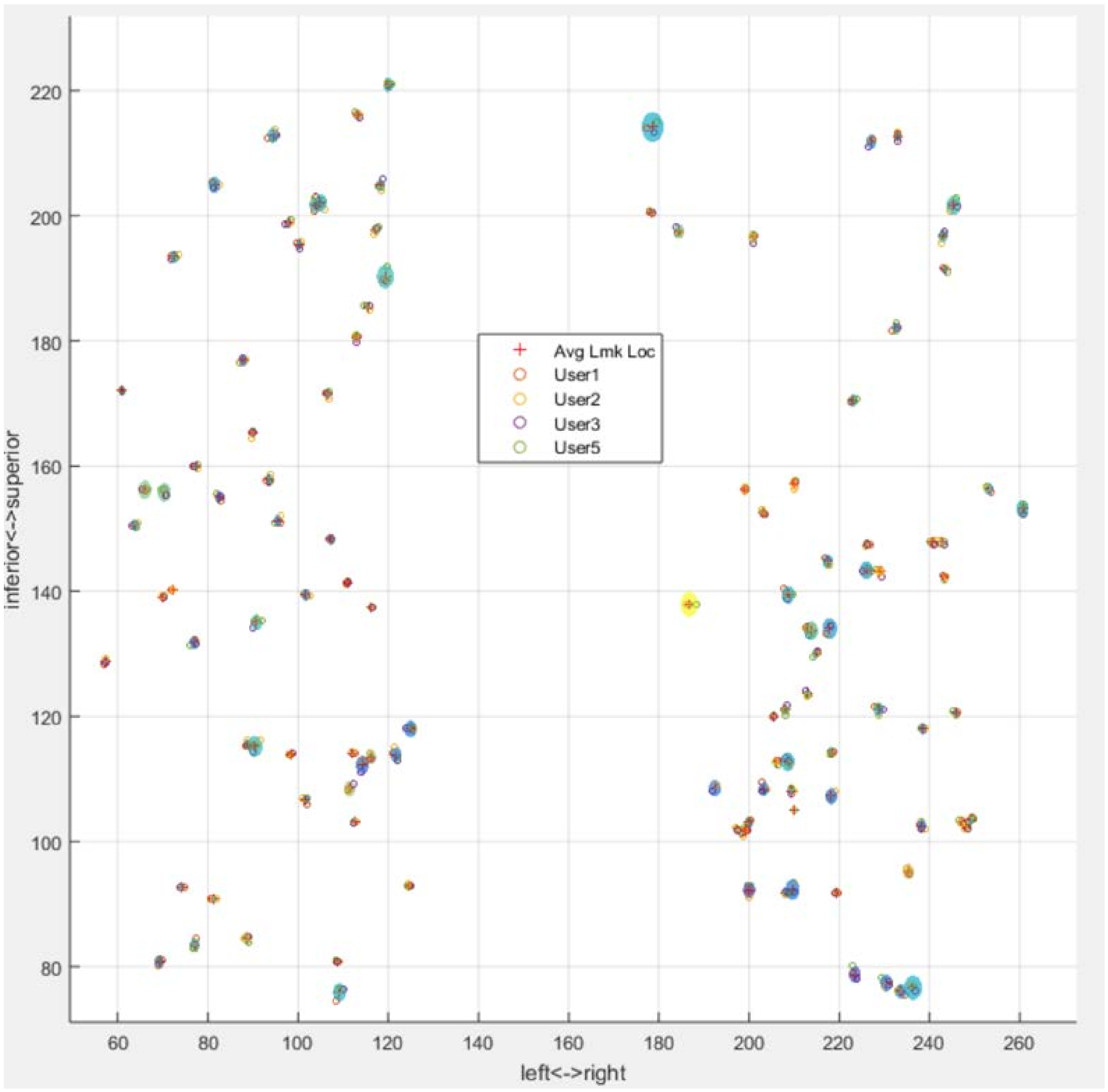


Figure 3.16: Landmarks labeled by 5 observers on the extreme inhale phase of PFS-007. Landmarks were projected onto coronal plane. The mean landmarks locations are represented by red crosses, while the individual observer-labeled landmarks are shown as small circles of different colors. There is a disk surrounding each mean landmark location, whose radius is the maximum landmark variation among all observers for that landmark.

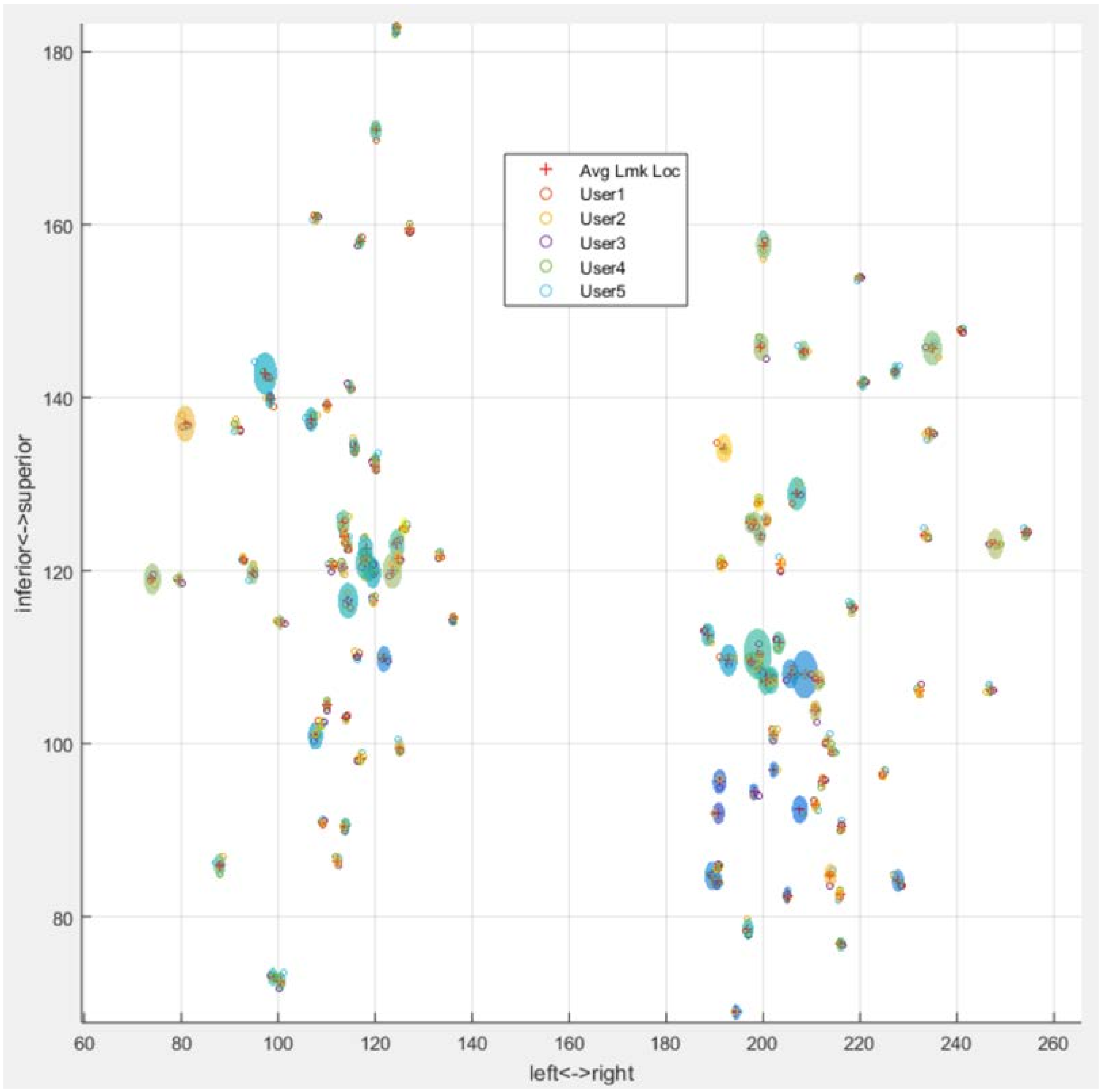


Figure 3.17: Landmarks labeled by 4 observers on the extreme inhale phase of PFS-008. Landmarks were projected onto coronal plane. The mean landmarks locations are represented by red crosses, while the individual observer-labeled landmarks are shown as small circles of different colors. There is a disk surrounding each mean landmark location, whose radius is the maximum landmark variation among all observers for that landmark.

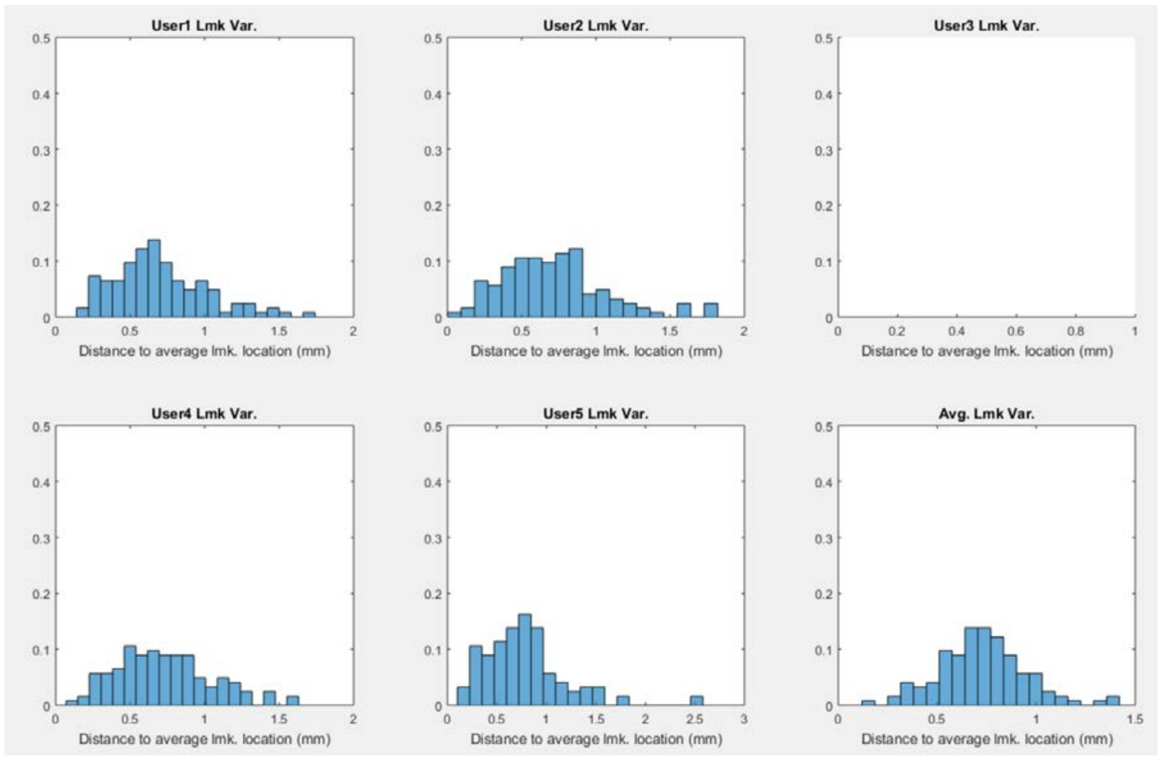


Figure 3.18: Histograms of landmark variation for each involved observer on PFS-001.

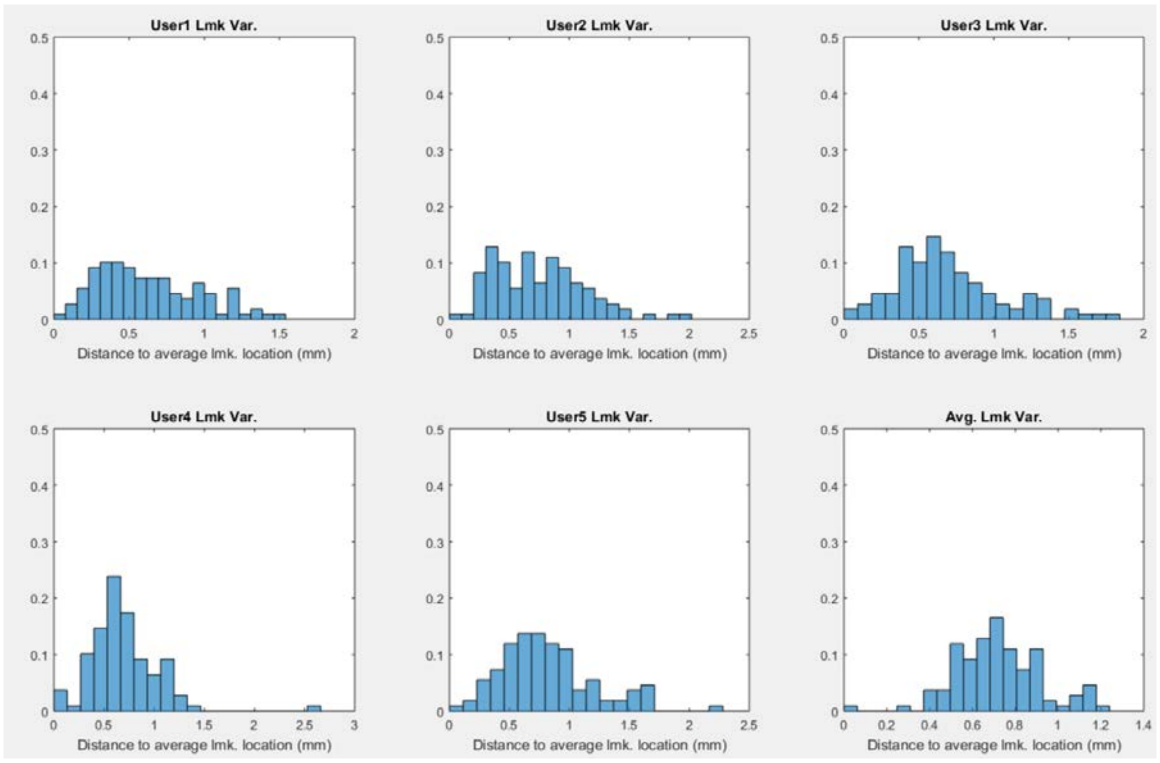


Figure 3.19: Histograms of landmark variation for each involved observer on PFS-002.

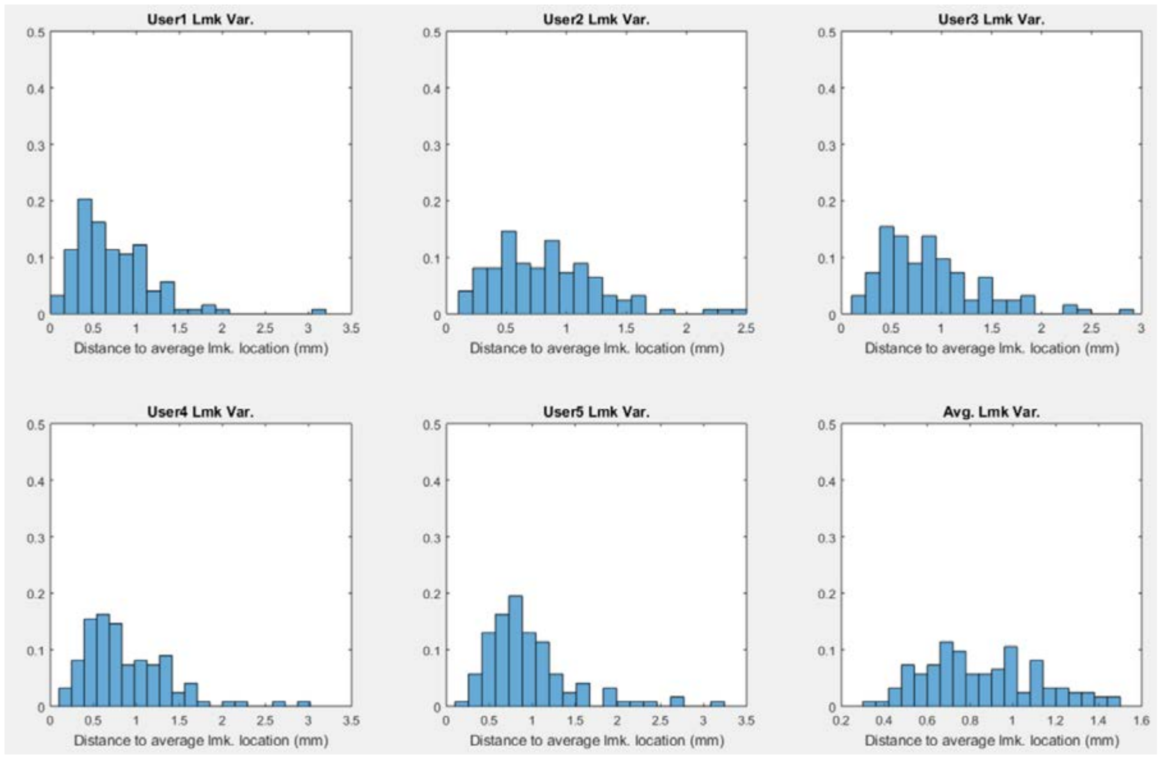


Figure 3.20: Histograms of landmark variation for each involved observer on PFS-004.

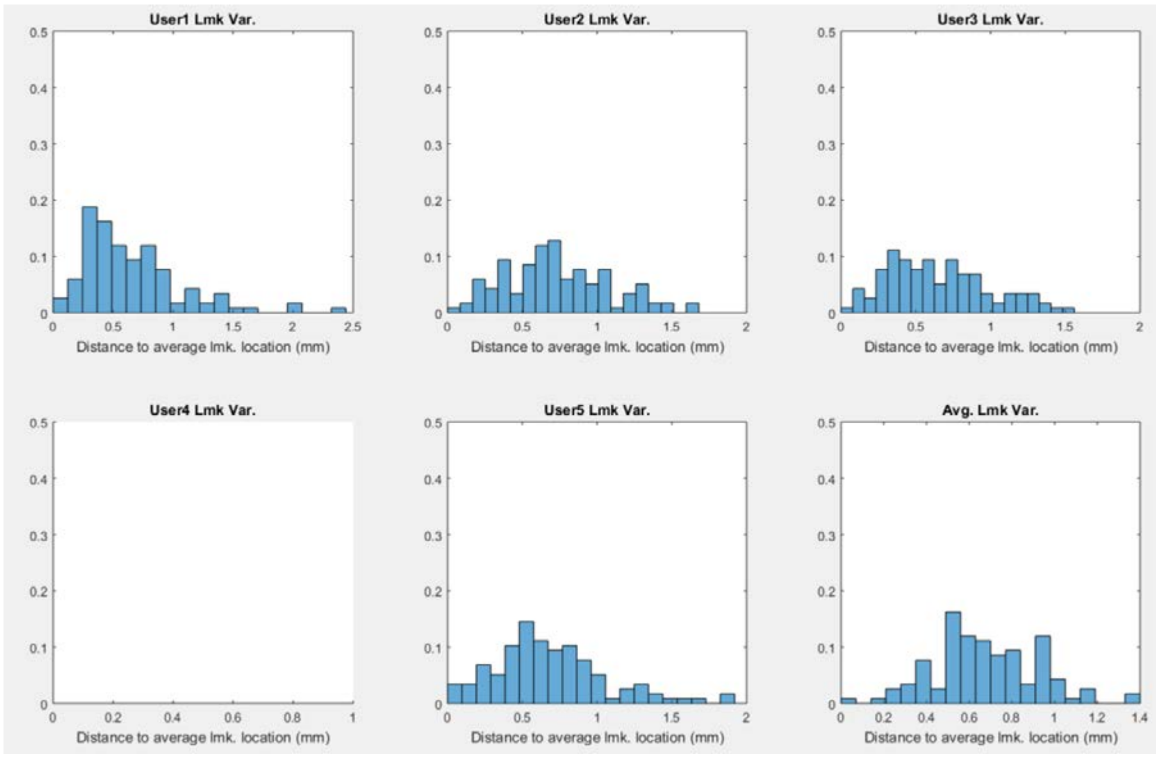


Figure 3.21: Histograms of landmark variation for each involved observer on PFS-007.

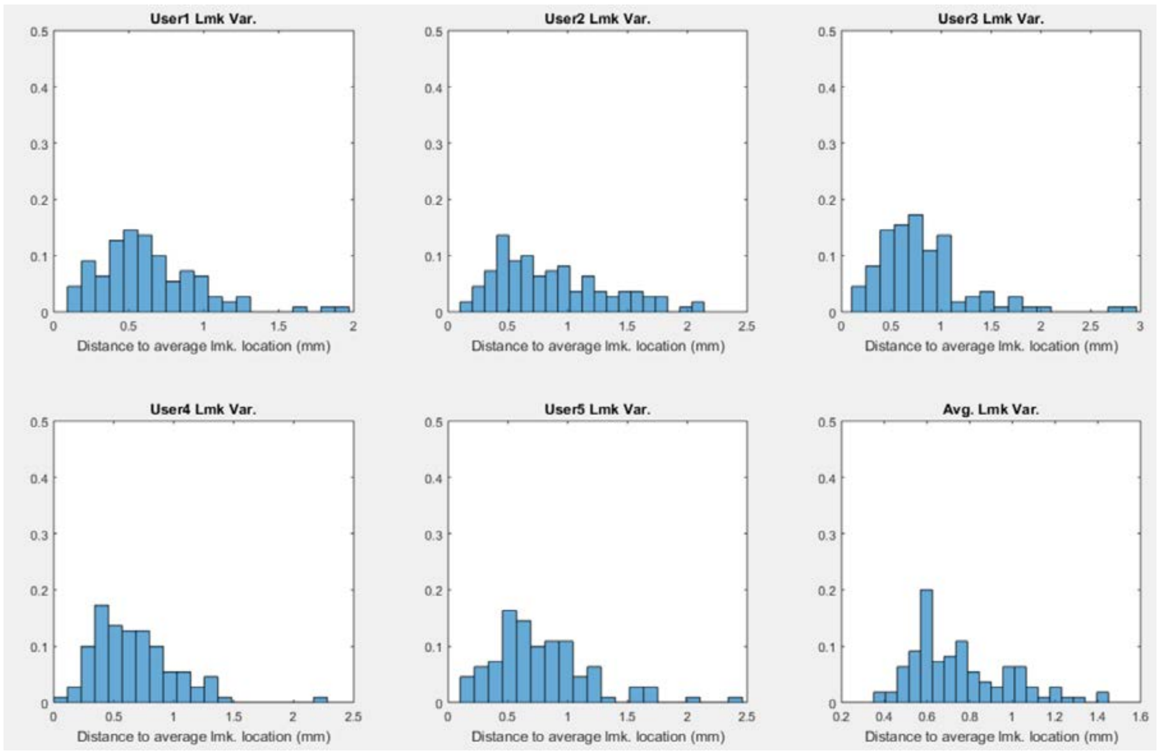


Figure 3.22: Histograms of landmark variation for each involved observer on PFS-008.

mind that the experiment is the first time for all but one of the 5 observers to label landmarks on pulmonary CT images. Given more time to practice, they would develop more familiarity with the anatomy of the lung, as well as the landmarking software itself and may thus very well achieve higher accuracy using this software.

CHAPTER 4
CONCLUSION AND DISCUSSION

The 4D tissue preservation algorithm inherits the advantage of 3D SSTVD to handle registration scenarios where spatially corresponding voxels have varying CT numbers due to changes in fraction of tissue within voxels while the total tissue volume is preserved. Meanwhile, the 4D cubic B-spline transformation model and temporally extended linear elasticity ensure the temporal smoothness of the deformation field. Comparison results on 4DCT data sets indicate the proposed 4D SSTVD algorithm strikes a good balance between accuracy and temporal regularity. Without an explicit target image in the 4D registration framework, all information of the dynamic data set is considered simultaneously, avoiding bias toward any specific reference image and increasing robustness against potential outliers caused by artifacts or noise[19]. By incorporating temporal information within 4DCT data sets, the proposed method can provide more relevant information for motion tracking and ventilation estimation and thus aid in radiotherapy treatment planning [5],[19]. Another benefit of considering temporal information is that temporal interpolation can be used to provide an estimate of lung motion at certain intermediate time points that are not present in the initial 4DCT data set.

One limitation of this work is the accuracy of estimated 4D inverse transform. Since B-spline parameterization of the geometric transform does not have closed form inverse expression, the 4D inverse transform should have been estimated point by point, without using another B-spline parametric model. But in our implementation with the Elastix package, the 4D inverse transform was estimated using a set of finer B-spline grids than were used to estimate the 4D forward transform. This way, the inverse transform can have higher degrees of freedom, but this may still lead to inaccurate estimation of inverse transforms for some 4DCT data sets. Figure 4.1 shows a 3D Jacobian image for PFS-002 4DCT data set. This Jacobian image corresponds to the estimated 3D spatial transform $\mathbf{T}_{0\to0}$ (from the extreme exhale phase to itself) that is obtained by composing the forward and inverse 4D transforms. Ideally, we would expect this spatial transform $\mathbf{T}_{0\to0}$ to be the identity map, and the Jacobian value should be 1 across the Jacobian image. But we can vaguely observe some patterns in Fig.4.1, indicating that the estimated spatial transform $\mathbf{T}_{0\to0}$ is not identity, which in turn suggests that the inverse map is not extremely accurate because of the B-spline parameterization.

The landmarking software demonstrated its effectiveness and reliability to aid users to efficiently and accurately label corresponding landmarks on 4D data sets through the inter-observer experiments. As the users grow more familiar with the lung anatomy as well as the software itself, we would expect the labeled landmarks to be even more accurate.

During the implementation of the 4D landmarking software, a design decision was made to always present the deformed moving image to the user every time after the normalization feature is used. This would make the extreme inhale phase, on which the user needs to label corresponding landmarks, to become more and more aligned with the extreme exhale phase, and provide more accurate suggested locations

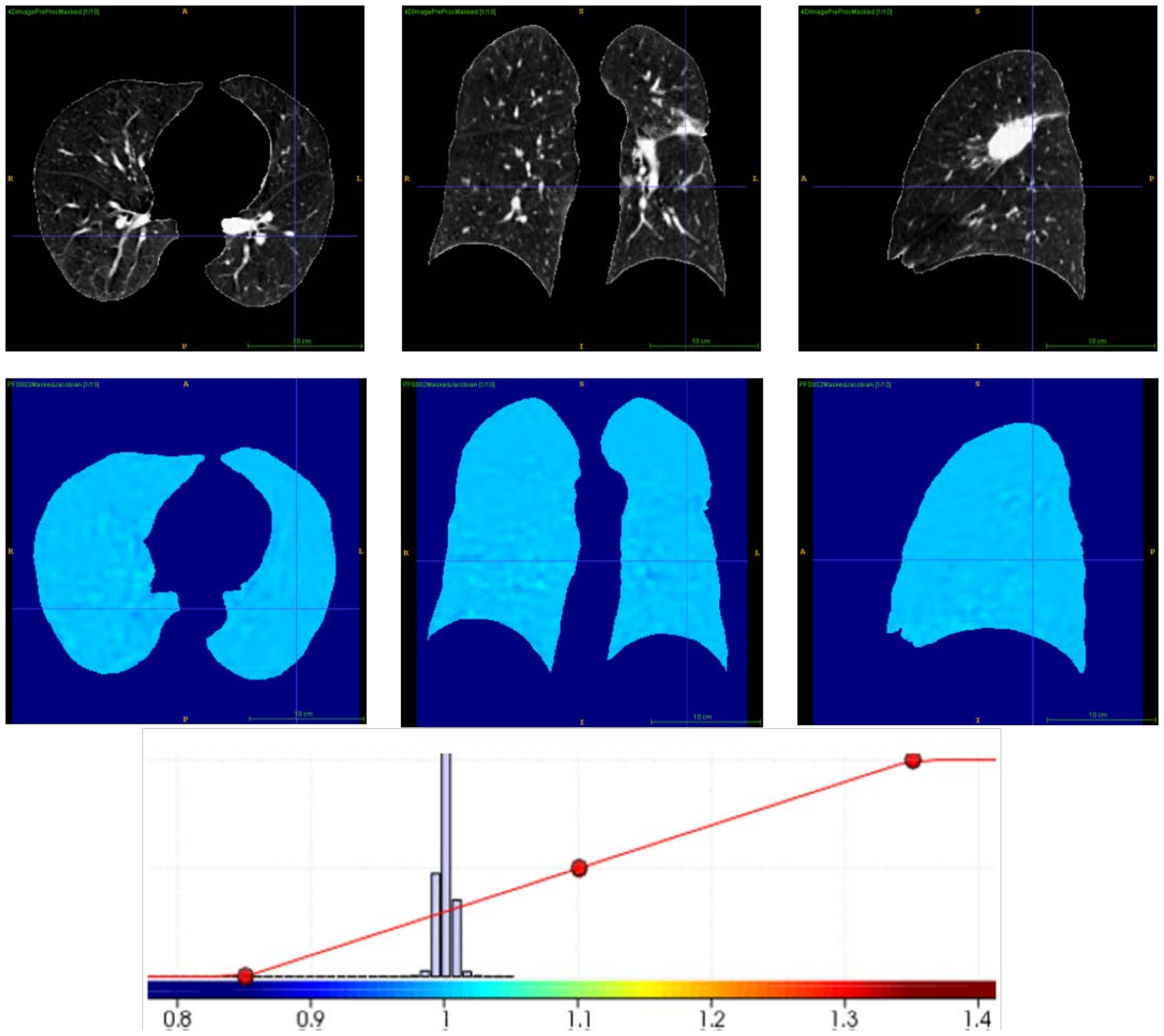


Figure 4.1: Illustration of 4D SSTVD inverse inaccuracy. The figure illustrates 3D Jacobian image corresponding to 3D spatial transform $\mathbf{T}_{0\to 0}$, obtained by composition of 4D forward and inverse transforms, for data set PFS-002. We can see that the Jacobian image is not 1 uniformly, indicating the spatial transform is not identity everywhere.

for the remaining landmarks. However, the deformed moving image is inevitably less sharp compared to the original image due to linear interpolation. This blurring effect may potentially render the user unable to pick the exact corresponding voxel on the deformed moving image. In the future, after the software is extensively used, this design decision may need to be revoked based on user feedback. Displaying the original un-deformed extreme inhale phase image may turn out to be more beneficial to the user, and we may only need the normalization feature to generate more accurate suggested landmark locations on the extreme inhale phase image, without actually deforming it.

Another feature that will bring great convenience to the user is automatic landmark detection. For now, the user will have to rely on other software tools or manual labor to obtain landmarks on the extreme exhale phase to initiate the whole landmarking process. In the future, the automatic landmark detection feature may be developed to be incorporated into this software, making it a self-contained 4D landmarking tool by itself.

APPENDIX A
4D SSTVD ELASTIX/ITK C++ IMPLEMENTATION USER GUIDE

Step 0: Preparation

1) Download Elastix 4.8 source code and ITK 4.8.2 source code (not the newest ITK 4.9.0 because Elastix does not support it yet).

2) Put the three source code folders "**SumSquaredTissueVolumeDifferenceMetric4D**", "**LinearElasticityPenalty4D**", "**DisplacementMagnitudePenalty**" into Elastix Metrics folder, e.g.: XXXXXX/elastix_sources_v4.8/src/Components/Metrics/. Replace existing "DisplacementMagnitudePenalty" if prompted.

3) In Linux server like c-reg, the system clock is sometimes not synced with the internet. So you need to input the following command to prevent warnings like "Clock skew detected" or "XXX has modification xxx seconds in the future":

   ```
   find XXX/elastix_sources_v4.8/src/Components/Metrics/* -type f -exec touch {} +
   ```

   where XXX specifies the path leading to "elastix_sourcec_v4.8" from your current location.

4) Ccmake/make/build/compile (whatever the process is called) ITK (in 64 bit version) in Release mode with "Module_ITKReview" option selected in Cmake and then ccmake/make/build/compile Elastix in Release mode with "USE_ALL_COMPONENTS" selected and add ";float" after "short" in "ELASTIX_IMAGE_4D_PIXELTYPES" option.

5) Ccmake/make/build/compile the "PreProcessing" folder in Release mode using Cmake to generate an executable named "PreProcessingModule".

Step 1: Data Preprocessing

[Description]:
This step combines 3D images into 4D, performs intensity preprocessing to transform voxel HU intensity into "tissue volume" (fraction of tissue volume within a voxel volume).
Then this step will combine 3D masks into 4D mask and apply the 4D mask onto the 4D image.

[Usage:]
In command line interface, type in command like the following:

"PreProcessingModule 4DImage.nii 4DImagePreProc.nii 4DMask.nii 4DImagePreProcMasked.nii 3DImage1.nii … 3DImageN.nii 3DMask1.nii … 3DMaskN.nii"

where you need to specify the paths of the 3D image files and the 3D masks.
A real example input using SCAN1 data from PFS-002 as well as Sarah's 4D segmented masks is like the following:

PreProcessingModule 4DImage.nii 4DImagePreProc.nii 4DMask.nii 4DImagePreProcMasked.nii original_data/BeforeRT_4DCT_0EX_SCAN1.hdr

**original_data/BeforeRT_4DCT_20IN_SCAN1.hdr
original_data/BeforeRT_4DCT_40IN_SCAN1.hdr
original_data/BeforeRT_4DCT_60IN_SCAN1.hdr
original_data/BeforeRT_4DCT_80IN_SCAN1.hdr
original_data/BeforeRT_4DCT_100IN_SCAN1.hdr
original_data/BeforeRT_4DCT_80EX_SCAN1.hdr
original_data/BeforeRT_4DCT_60EX_SCAN1.hdr
original_data/BeforeRT_4DCT_40EX_SCAN1.hdr
original_data/BeforeRT_4DCT_20EX_SCAN1.hdr original_data/mask0EX.hdr
original_data/mask20IN.hdr original_data/mask40IN.hdr original_data/mask60IN.hdr
original_data/mask80IN.hdr original_data/mask100IN.hdr original_data/mask80EX.hdr
original_data/mask60EX.hdr original_data/mask40EX.hdr original_data/mask20EX.hdr**

The first 4 files starting with "4D" are the outputs. Specifically, we shall use "4DImagePreProc.nii" or "4DImagePreProcMaksed.nii" for registration depending on whether we want to use masked image or not. And the "4DImage.nii" can be used to generate deformed 4D image using the final transform parameters obtained from registration process.

Note: Do not use "4DImage.nii" for 4D SSTVD registration.

Step 2: Registration

[Description]:

This step performs 4D SSTVD (with or without linear elasticity regularization) registration on the preprocessed 4D image dataset using the parameters provided in files "param.forward.txt" and "param.inverse.txt". Just use common Elastix registration framework.
Note: You can modify the parameter file and choose to only use 4D SSTVD intensity metric without linear elasticity regularization and witness a further improvement in running speed. From the results of all my experiments, I haven't encountered negative Jacobians even without regularization.

After making the algorithm support multi-threading, I changed the default optimizer from adaptive stochastic gradient descent (ASGD) to quasi-newton LBFGS. This is because the ASGD optimizer seems to be non-multi-threaded, while the LBFGS optimizer is, which makes ASGD a lot slower than LBFGS despite it being a first order optimizer while LBFGS is 2nd order!

You can still choose to use adaptive stochastic gradient descent optimizer, but the running time will be more.

[Usage]:

1) Perform registration that deforms the 4D image dataset to an implicit temporal average image that is previously unknown and constantly updated throughout the registration process. Type in command like:

   ```
   XXX/bin/elastix –f XXX/4DImagePreProcMasked.nii –m XXX/4DImagePreProcMasked.nii
   –p XXX/param.forward.txt –out elastix_output_forward
   ```

   where elastix is the compiled executable in your elastix build folder.

   Note that fixed and moving images are the same 4D image. You need to create the output folder in advance.

   You can use options like "-priority high" and/or "-threads XXX" to specify the priority and number of threads you want to dedicate to the registration process.

2) Perform another registration to approximate the inverse transform. Type in command like:

   ```
   XXX/bin/elastix –f XXX/4DImagePreProcMasked.nii –m XXX/4DImagePreProcMasked.nii
   –p XXX/param.inverse.txt –t0 elastix_output_forward/TransformParameters.0.txt –out
   elastix_output_inverse
   ```

   Note that you need to provide transform parameters from the forward transform as initial transform using the "-t0" option as shown above.

3) Combine forward and inverse transform to get spatial transforms that deform all phases to a specific phase you designated. Type in command like:

   ```
   XXX/combine.py point elastix_output_forward/TransformParameters.0.txt
   elastix_output_inverse/TransformParameters.0.txt combined_transform.0.txt
   combined_transform.1.txt 0
   ```

   Note that the second word "point" is necessary and the very last 0 indicates we want to obtain transforms that deform other phases to phase 0. You can change this number to 0,1,2,…,9 to acquire transforms that deform all phases to another specific phase.

   You can also combine steps 1) and 2) in one command:

   ```
   XXX/bin/elastix –f XXX/4DImagePreProcMasked.nii –m XXX/4DImagePreProcMasked.nii
   –p XXX/param.forward.txt XXX/param.inverse.txt –out elastix_output
   ```

   and then combine forward and inverse transform using:

```
XXX/combine.py point elastix_output/TransformParameters.0.txt
elastix_output/TransformParameters.1.txt combined_transform.0.txt
combined_transform.1.txt 0
```

Step 3: Transformation/Deformation

[Description]:
Actually deform all phases within the 4D image to a certain phase. Generate other information resulting from the transforms if required.

[Usage]:
Type in command like:

```
XXX/transformix –in XXX/4DImagePreProc.nii –tp combined_transform.1.txt –out
transformix_output
```

Note that you need to use combined_transform.1.txt instead of combined_transform.0.txt. You can specify options like "-def all", "-jac all" or "-jacmat all" to get displacement vectors, Jacobian values or Jacobian matrices at each voxel for analysis.

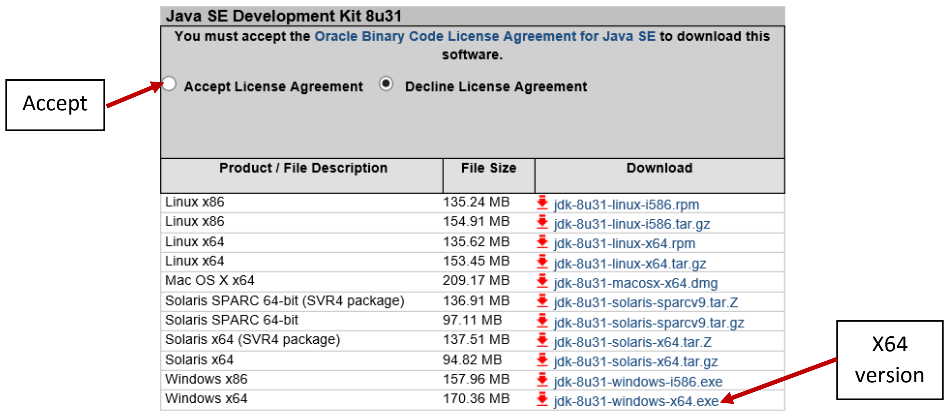
APPENDIX B
4D LANDMARKING SOFTWARE USER GUIDE

Multi-Volume Landmarking Tool Tutorial

[Installation of software]:

[1] Download and install the latest 64bit version of Java JDK in its default path:

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.



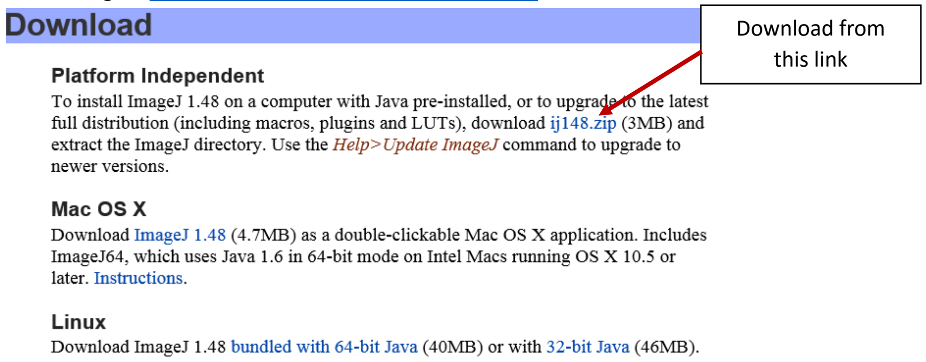
Java SE Development Kit 8u31

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

○ **Accept License Agreement** ● **Decline License Agreement**

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u31-linux-i586.rpm
Linux x86	154.91 MB	jdk-8u31-linux-i586.tar.gz
Linux x64	135.62 MB	jdk-8u31-linux-x64.rpm
Linux x64	153.45 MB	jdk-8u31-linux-x64.tar.gz
Mac OS X x64	209.17 MB	jdk-8u31-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	136.91 MB	jdk-8u31-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.11 MB	jdk-8u31-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.51 MB	jdk-8u31-solaris-x64.tar.Z
Solaris x64	94.82 MB	jdk-8u31-solaris-x64.tar.gz
Windows x86	157.96 MB	jdk-8u31-windows-i586.exe
Windows x64	170.36 MB	jdk-8u31-windows-x64.exe

[2] Download ImageJ: http://rsb.info.nih.gov/ij/download.html.



Download

Platform Independent

To install ImageJ 1.48 on a computer with Java pre-installed, or to upgrade to the latest full distribution (including macros, plugins and LUTs), download ij148.zip (3MB) and extract the ImageJ directory. Use the *Help>Update ImageJ* command to upgrade to newer versions.

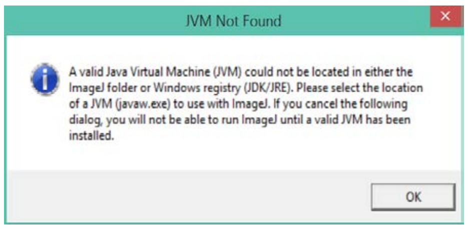
Mac OS X

Download ImageJ 1.48 (4.7MB) as a double-clickable Mac OS X application. Includes ImageJ64, which uses Java 1.6 in 64-bit mode on Intel Macs running OS X 10.5 or later. Instructions.

Linux

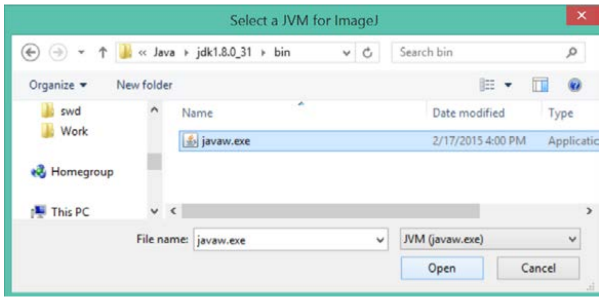
Download ImageJ 1.48 bundled with 64-bit Java (40MB) or with 32-bit Java (46MB).

[3.1] Install ImageJ. Unzip the downloaded ImageJ folder and put it wherever you like. When you first run the executable "ImageJ.exe", the program will try to do some configuration and you may be prompted to specify the path of valid Java Virtual Machine on your computer.



JVM Not Found

A valid Java Virtual Machine (JVM) could not be located in either the ImageJ folder or Windows registry (JDK/JRE). Please select the location of a JVM (javaw.exe) to use with ImageJ. If you cancel the following dialog, you will not be able to run ImageJ until a valid JVM has been installed.



Select a JVM for ImageJ

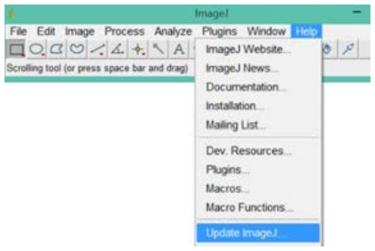
« Java › jdk1.8.0_31 › bin

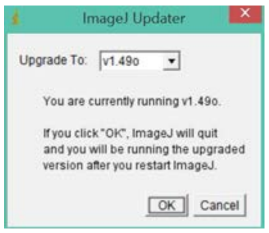
Name	Date modified	Type
javaw.exe	2/17/2015 4:00 PM	Applicatio

File name: javaw.exe — JVM (javaw.exe)

Navigate through path “C:\Program Files\Java\jdk1.8.0_31\bin\javaw.exe” to select the JVM executable and you will be notified that a configuration file has been generated for ImageJ. It is also possible that the location of Java Virtual Machine has been automatically detected and the above prompt window does not show up. Note: do not use the javaw.exe in “Program Files (x86)” folder.

[3.2] Update ImageJ to the newest version by clicking “Help”→”Update ImageJ”. Then click “OK” in the pop-up dialogue. ImageJ will download and install updates and then close itself. You need to reopen it manually.

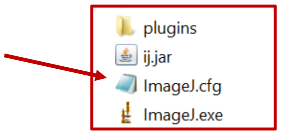




You are currently running v1.49o.

If you click "OK", ImageJ will quit and you will be running the upgraded version after you restart ImageJ.

[4] Open the file “ImageJ.cfg” with Notepad in the ImageJ main folder.



Change the second line in “ImageJ.cfg” from whatever it is to the following:

“C:\Program Files\Java\jdk1.8.0_31\bin\javaw.exe”.

At the beginning of the third line, change “-Xmx640m” into “-Xmx6400m” (we are actually changing the amount of available memory for Java) if you have 8GB of memory and leave the rest of the third line unchanged. After editing, the file “ImageJ.cfg” will look similar to this:

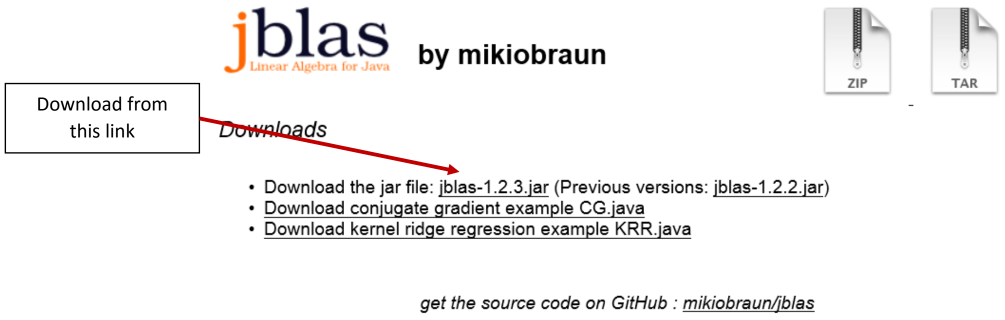
```
.
C:\Program Files\Java\jdk1.8.0_31\bin\javaw.exe
-Xmx6400m -cp C:\PROGRA~1\Java\JDK18~1.0_3\lib\tools.jar;ij.jar ij.ImageJ
```

It doesn’t matter if this part is different for you.

Then save the changes and close the file.

[5] Download jblas package (for fast matrix calculation):

http://mikiobraun.github.io/jblas/download.html.



Downloads

- Download the jar file: jblas-1.2.3.jar (Previous versions: jblas-1.2.2.jar)
- Download conjugate gradient example CG.java
- Download kernel ridge regression example KRR.java

get the source code on GitHub : mikiobraun/jblas

Put the downloaded “jblas-1.2.3.jar” file in directory “…\ImageJ\plugins\jars”.

[6] Create a new folder in directory “…\ImageJ\plugins\”. Let’s call this new folder “MVL”, for example. Then copy the source file “MultiVolLmk.java” into this folder (“…\ImageJ\plugins\MVL\”).

[7] Copy the file “RemoveShortcuts.txt” into directory “…\ImageJ\macros\”

[Data Preparation]:

[1] We want to move all 4D-CT image files, mask files and landmark files into folder “…\ImageJ\plugins\MVL\”. For our current inter-observer experiment, we need to grab our image files and landmark files from “lungmech” server. Specifically,

(a) Create subfolders “PFS-001”, “PFS-002”, “PFS-004”, “PFS-007”, “PFS-008”, “PFS-009”, “PFS-010”, “PFS-011” and “PFS-012” inside folder “…\ImageJ\plugins\MVL\”.

(b) Go to “lungmech”. In the folder “Du_4DCT_Database”, copy 20 image files:

“BeforeRT_4DCT_0EX_SCAN1.img”, “BeforeRT_4DCT_0EX_SCAN1.hdr”,
“BeforeRT_4DCT_20IN_SCAN1.img”, “BeforeRT_4DCT_20IN_SCAN1.hdr”,
“BeforeRT_4DCT_20EX_SCAN1.img”, “BeforeRT_4DCT_20EX_SCAN1.hdr”,

…

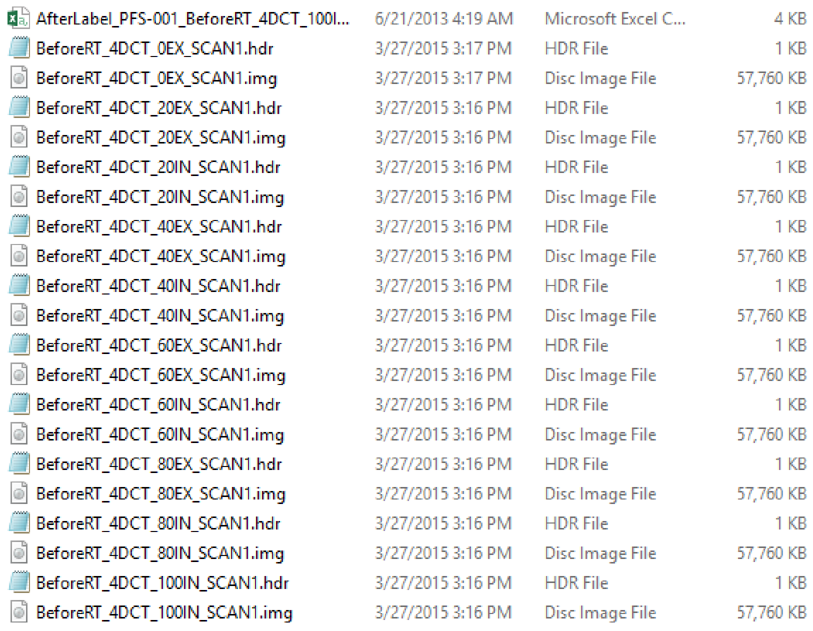
“BeforeRT_4DCT_100IN_SCAN1.img”, “BeforeRT_4DCT_100IN_SCAN1.hdr”

from the “resample” subfolder of each of the 9 patients “PFS-001”, “PFS-002”, “PFS-004”, “PFS-007”, “PFS-008”, “PFS-009”, “PFS-010”, “PFS-011” and “PFS-012” into our local folders of these patients.

(c) Copy the landmark files for the 9 patients from “…\Du_4DCT_Database\lmk” folder into our local folders of the 9 patients.

[Note]: The landmark data contain corresponding landmark locations on 0EX and 100IN phase images for each patient. The landmarks on 0EX phase image were obtained using iX software’s automatic landmark detection feature, and the corresponding landmarks on 100IN phase image were manually labeled by Kaifang. We have to bear with the fact that some automatically detected landmarks are not as qualified as we expect them to be.

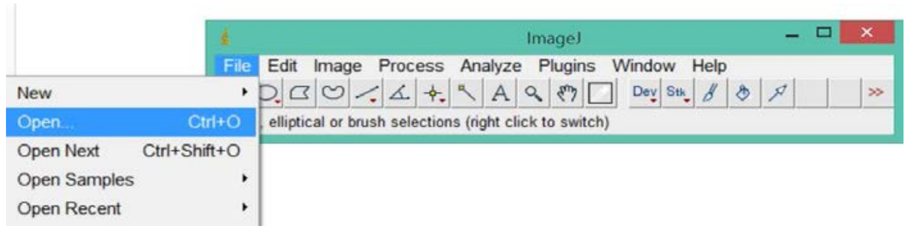
For illustrative purpose in this tutorial, we shall focus on patient "PFS-002". The folder "...\ImageJ\plugins\MVL\PFS-002" will look like the following after copying these files from "lungmech":



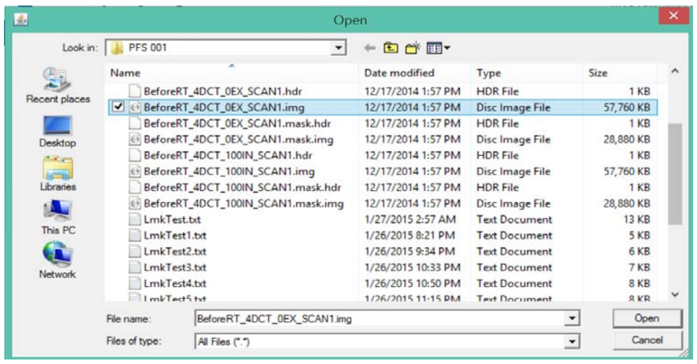
AfterLabel_PFS-001_BeforeRT_4DCT_100I...	6/21/2013 4:19 AM	Microsoft Excel C...	4 KB
BeforeRT_4DCT_0EX_SCAN1.hdr	3/27/2015 3:17 PM	HDR File	1 KB
BeforeRT_4DCT_0EX_SCAN1.img	3/27/2015 3:17 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_20EX_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_20EX_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_20IN_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_20IN_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_40EX_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_40EX_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_40IN_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_40IN_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_60EX_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_60EX_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_60IN_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_60IN_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_80EX_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_80EX_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_80IN_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_80IN_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_100IN_SCAN1.hdr	3/27/2015 3:16 PM	HDR File	1 KB
BeforeRT_4DCT_100IN_SCAN1.img	3/27/2015 3:16 PM	Disc Image File	57,760 KB

[2] To save memory, we convert 16-bit images into 8-bit (through numerous experiments, I found that adjusting brightness and contrast would compensate for the seeming loss of information to the extent that no visual difference can be noticed between 8-bit and 16-bit images). So now, we convert the 4DCT images into 8-bit.

Run the ImageJ.exe application and the main panel of ImageJ will show up. Click "File" → "Open" and then select, for example, the image "BeforeRT_4DCT_0EX_SCAN1.img" in folder "PFS-002".



elliptical or brush selections (right click to switch)



Open

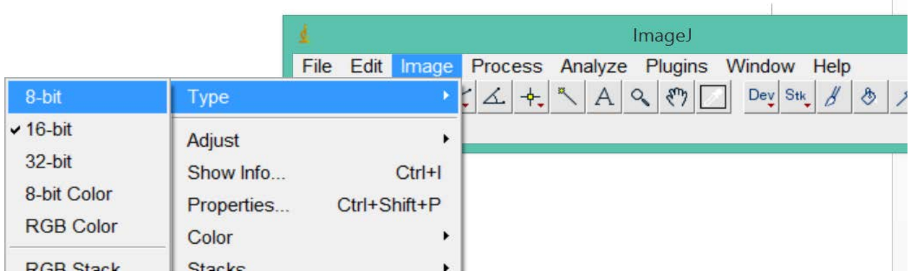
Look in: PFS 001

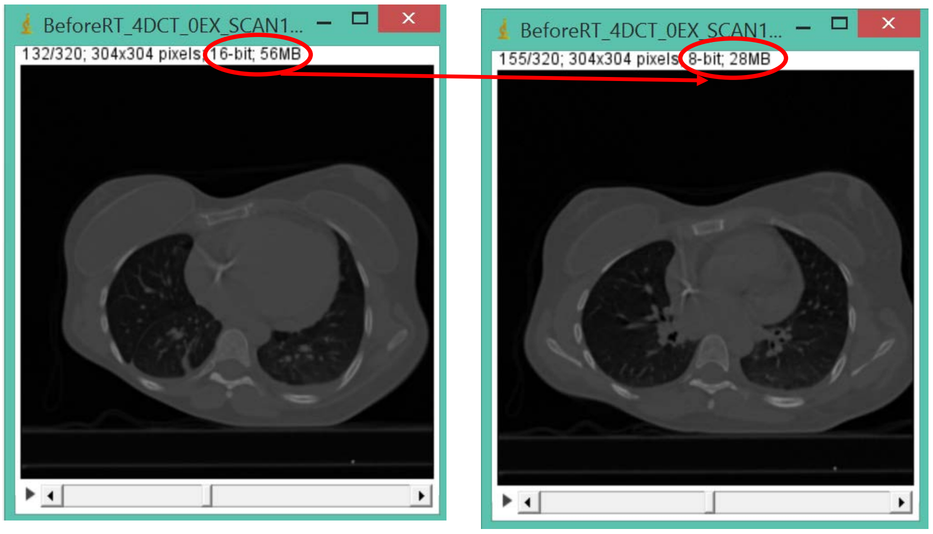
Name	Date modified	Type	Size
BeforeRT_4DCT_0EX_SCAN1.hdr	12/17/2014 1:57 PM	HDR File	1 KB
BeforeRT_4DCT_0EX_SCAN1.img	12/17/2014 1:57 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_0EX_SCAN1.mask.hdr	12/17/2014 1:57 PM	HDR File	1 KB
BeforeRT_4DCT_0EX_SCAN1.mask.img	12/17/2014 1:57 PM	Disc Image File	28,880 KB
BeforeRT_4DCT_100IN_SCAN1.hdr	12/17/2014 1:57 PM	HDR File	1 KB
BeforeRT_4DCT_100IN_SCAN1.img	12/17/2014 1:57 PM	Disc Image File	57,760 KB
BeforeRT_4DCT_100IN_SCAN1.mask.hdr	12/17/2014 1:57 PM	HDR File	1 KB
BeforeRT_4DCT_100IN_SCAN1.mask.img	12/17/2014 1:57 PM	Disc Image File	28,880 KB
LmkTest.txt	1/27/2015 2:57 AM	Text Document	13 KB
LmkTest1.txt	1/26/2015 8:21 PM	Text Document	5 KB
LmkTest2.txt	1/26/2015 9:34 PM	Text Document	6 KB
LmkTest3.txt	1/26/2015 10:33 PM	Text Document	7 KB
LmkTest4.txt	1/26/2015 10:50 PM	Text Document	8 KB
LmkTest5.txt	1/26/2015 11:15 PM	Text Document	8 KB

File name: BeforeRT_4DCT_0EX_SCAN1.img

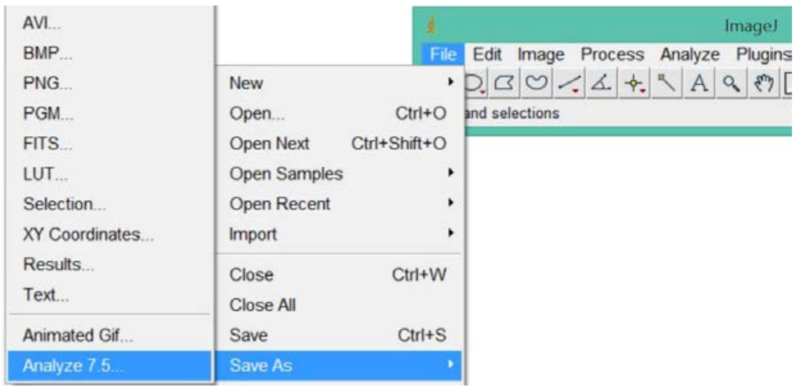
Files of type: All Files (*.*)

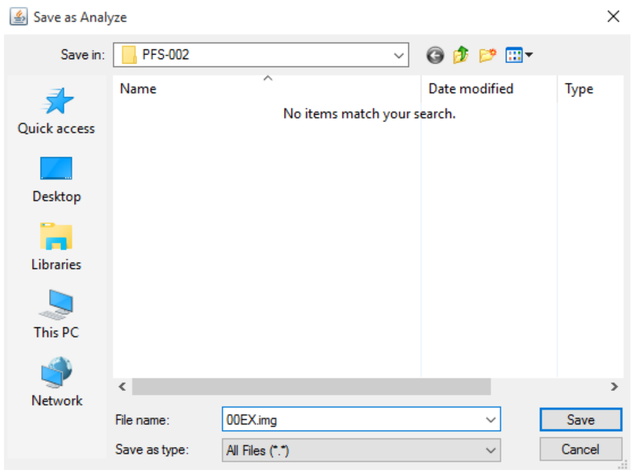
The image will be loaded. Then, in ImageJ panel, click "Image"→"Type"→"8-bit" and the previously opened image volume will be converted into 8-bit.




BeforeRT_4DCT_0EX_SCAN1...
132/320; 304x304 pixels; 16-bit; 56MB
BeforeRT_4DCT_0EX_SCAN1...
155/320; 304x304 pixels; 8-bit; 28MB

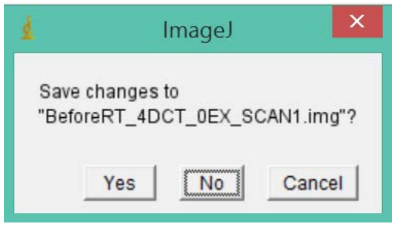
Next, save this 8-bit image into folder "...\ImageJ\plugins\MVL\PFS-002" by clicking "File"→"Save As"→"Analyze 7.5" and give it a new (and simple) name. For example, we call it "00EX.img".





No items match your search.
File name: 00EX.img
Save as type: All Files (*.*)

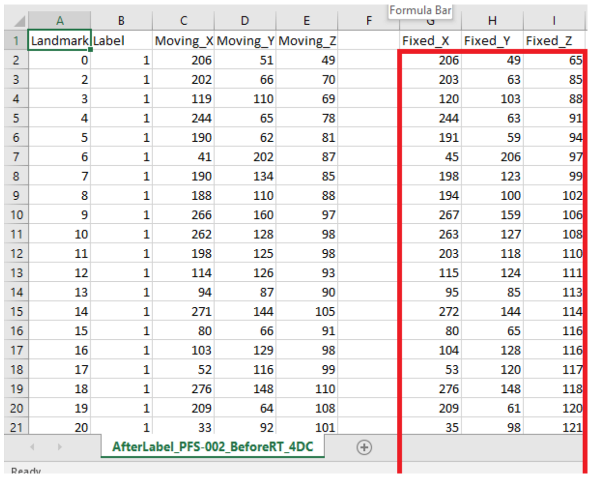
Finally, close the currently opened image "BeforeRT_4DCT_0EX_SCAN1.img". When you are prompted with the following message, choose "No" so that the original 16-bit image will not be overwritten.



Save changes to "BeforeRT_4DCT_0EX_SCAN1.img"?

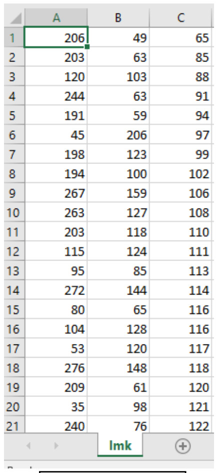
Repeat the exact same procedure for the other images “BeforeRT_4DCT_20IN_SCAN1.img”, …, “BeforeRT_4DCT_100IN_SCAN1.img”, …, “BeforeRT_4DCT_20EX_SCAN1.img” in folder “PFS-002” and name the resulting 8-bit image “20IN.img”, “40IN.img”, “60IN.img”, “80IN.img”, “100IN.img”, “80EX.img”, “60EX.img”, “40EX.img”, “20EX.img”.

[2] Since we only need the automatically detected landmarks on 00EX phase, we shall modify the “csv” landmark file into a simpler format. Open the original landmark file with Excel. Copy the three columns corresponding to coordinates of landmarks on the “fixed image”, i.e., 00EX phase. Paste these three columns into an Excel spread sheet and save as “lmk.csv”.



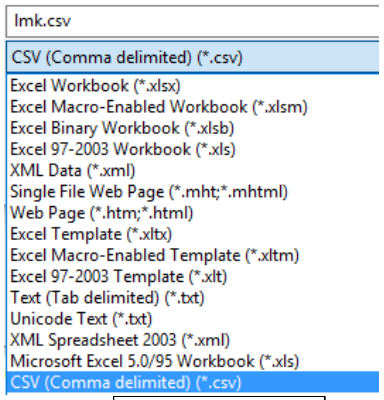
	A	B	C	D	E	F	G	H	I
1	Landmark	Label	Moving_X	Moving_Y	Moving_Z		Fixed_X	Fixed_Y	Fixed_Z
2	0	1	206	51	49		206	49	65
3	2	1	202	66	70		203	63	85
4	3	1	119	110	69		120	103	88
5	4	1	244	65	78		244	63	91
6	5	1	190	62	81		191	59	94
7	6	1	41	202	87		45	206	97
8	7	1	190	134	85		198	123	99
9	8	1	188	110	88		194	100	102
10	9	1	266	160	97		267	159	106
11	10	1	262	128	98		263	127	108
12	11	1	198	125	98		203	118	110
13	12	1	114	126	93		115	124	111
14	13	1	94	87	90		95	85	113
15	14	1	271	144	105		272	144	114
16	15	1	80	66	91		80	65	116
17	16	1	103	129	98		104	128	116
18	17	1	52	116	99		53	120	117
19	18	1	276	148	110		276	148	118
20	19	1	209	64	108		209	61	120
21	20	1	33	92	101		35	98	121

Original csv file



	A	B	C
1	206	49	65
2	203	63	85
3	120	103	88
4	244	63	91
5	191	59	94
6	45	206	97
7	198	123	99
8	194	100	102
9	267	159	106
10	263	127	108
11	203	118	110
12	115	124	111
13	95	85	113
14	272	144	114
15	80	65	116
16	104	128	116
17	53	120	117
18	276	148	118
19	209	61	120
20	35	98	121
21	240	76	122

New csv file



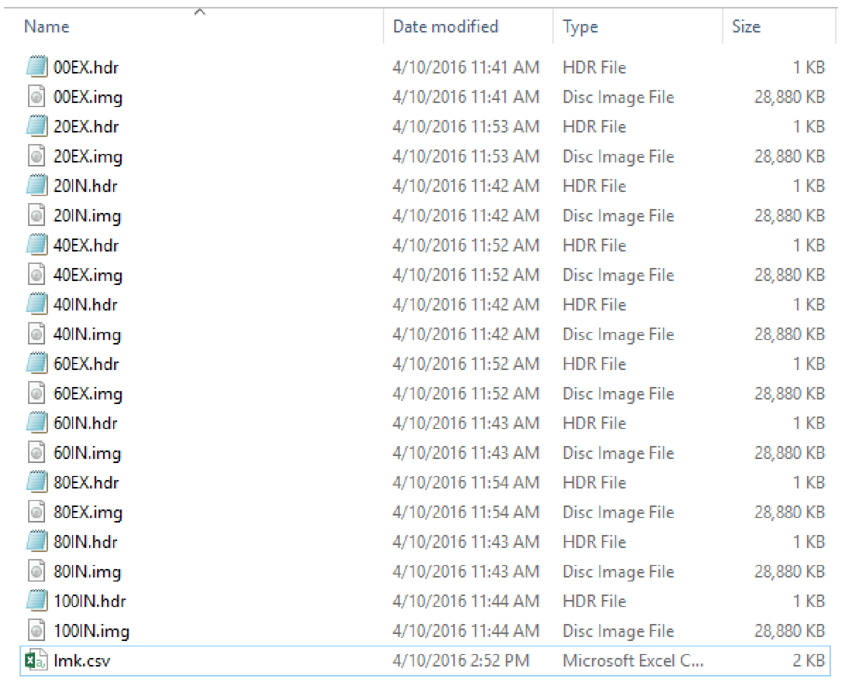
lmk.csv

CSV (Comma delimited) (*.csv)

Excel Workbook (*.xlsx)
Excel Macro-Enabled Workbook (*.xlsm)
Excel Binary Workbook (*.xlsb)
Excel 97-2003 Workbook (*.xls)
XML Data (*.xml)
Single File Web Page (*.mht;*.mhtml)
Web Page (*.htm;*.html)
Excel Template (*.xltx)
Excel Macro-Enabled Template (*.xltm)
Excel 97-2003 Template (*.xlt)
Text (Tab delimited) (*.txt)
Unicode Text (*.txt)
XML Spreadsheet 2003 (*.xml)
Microsoft Excel 5.0/95 Workbook (*.xls)
CSV (Comma delimited) (*.csv)

Save as csv file

Now, you can delete all 16 bit images and the original landmark file, and the folder "…\ImageJ\plugins\MVL\PFS-002" will be left with the following files:



Name	Date modified	Type	Size
00EX.hdr	4/10/2016 11:41 AM	HDR File	1 KB
00EX.img	4/10/2016 11:41 AM	Disc Image File	28,880 KB
20EX.hdr	4/10/2016 11:53 AM	HDR File	1 KB
20EX.img	4/10/2016 11:53 AM	Disc Image File	28,880 KB
20IN.hdr	4/10/2016 11:42 AM	HDR File	1 KB
20IN.img	4/10/2016 11:42 AM	Disc Image File	28,880 KB
40EX.hdr	4/10/2016 11:52 AM	HDR File	1 KB
40EX.img	4/10/2016 11:52 AM	Disc Image File	28,880 KB
40IN.hdr	4/10/2016 11:42 AM	HDR File	1 KB
40IN.img	4/10/2016 11:42 AM	Disc Image File	28,880 KB
60EX.hdr	4/10/2016 11:52 AM	HDR File	1 KB
60EX.img	4/10/2016 11:52 AM	Disc Image File	28,880 KB
60IN.hdr	4/10/2016 11:43 AM	HDR File	1 KB
60IN.img	4/10/2016 11:43 AM	Disc Image File	28,880 KB
80EX.hdr	4/10/2016 11:54 AM	HDR File	1 KB
80EX.img	4/10/2016 11:54 AM	Disc Image File	28,880 KB
80IN.hdr	4/10/2016 11:43 AM	HDR File	1 KB
80IN.img	4/10/2016 11:43 AM	Disc Image File	28,880 KB
100IN.hdr	4/10/2016 11:44 AM	HDR File	1 KB
100IN.img	4/10/2016 11:44 AM	Disc Image File	28,880 KB
lmk.csv	4/10/2016 2:52 PM	Microsoft Excel C...	2 KB

[Software usage]:

There are three major steps when using this 4D landmarking software to label landmarks on 4DCT data sets.

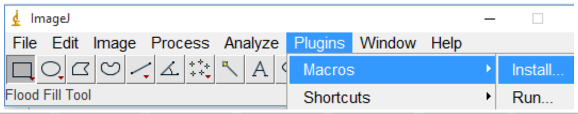
The first step is to acquire automatically detected landmark on one phase image within the 4D data set, preferably the extreme exhale phase, 00EX.

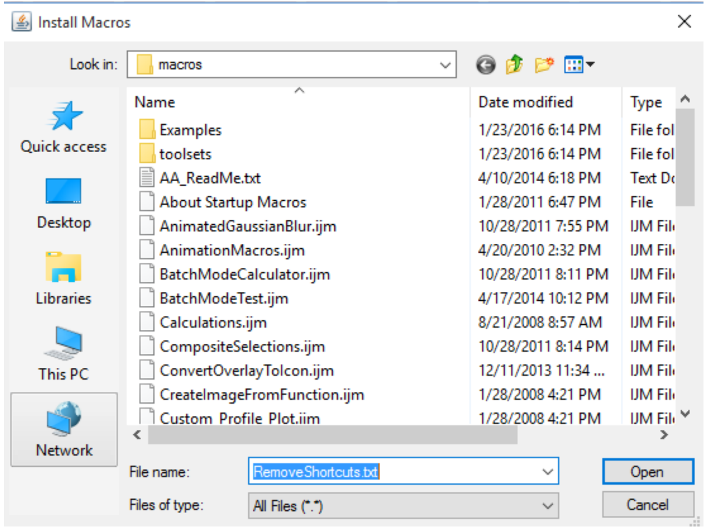
The second step is to use the "normalization" feature multiple times to label corresponding landmarks on another phase within the 4D data set, preferably on the extreme inhale phase, 100IN.

The third step is to use the "landmark location linear interpolation" feature to label landmarks on all the remaining phases (20IN, 40IN, 60IN, 80IN, 80EX, 60EX, 40EX, 20EX).

What we have done with the landmark file has taken care of the first step. Now we talk about the second and the third step in detail.

[0] In the ImageJ panel, click “Plugins” → “Macros” →”Install…”. Navigate to “…\ImageJ\macros” and select “RemoveShortcuts.txt” and click “Open”.



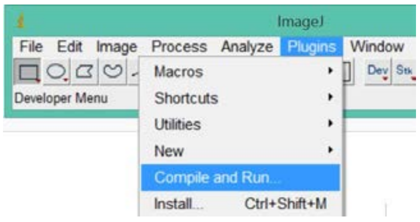


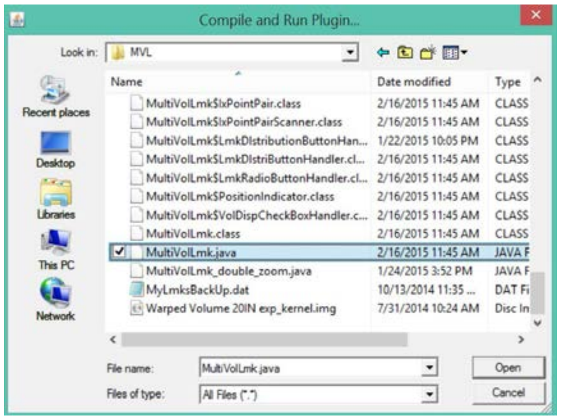
Look in: macros

Name	Date modified	Type
Examples	1/23/2016 6:14 PM	File fol
toolsets	1/23/2016 6:14 PM	File fol
AA_ReadMe.txt	4/10/2014 6:18 PM	Text Dc
About Startup Macros	1/28/2011 6:47 PM	File
AnimatedGaussianBlur.ijm	10/28/2011 7:55 PM	IJM Fil
AnimationMacros.ijm	4/20/2010 2:32 PM	IJM Fil
BatchModeCalculator.ijm	10/28/2011 8:11 PM	IJM Fil
BatchModeTest.ijm	4/17/2014 10:12 PM	IJM Fil
Calculations.ijm	8/21/2008 8:57 AM	IJM Fil
CompositeSelections.ijm	10/28/2011 8:14 PM	IJM Fil
ConvertOverlayTolcon.ijm	12/11/2013 11:34 ...	IJM Fil
CreateImageFromFunction.ijm	1/28/2008 4:21 PM	IJM Fil
Custom_Profile_Plot.ijm	1/28/2008 4:21 PM	IJM Fil

File name: RemoveShortcuts.txt
Files of type: All Files (*.*)

[1] In the ImageJ panel, click “Plugins” → “Compile and Run…”. Navigate to “…\ImageJ\plugins\MVL” and select “MultiVolLmk.java” source file and click “Open”.





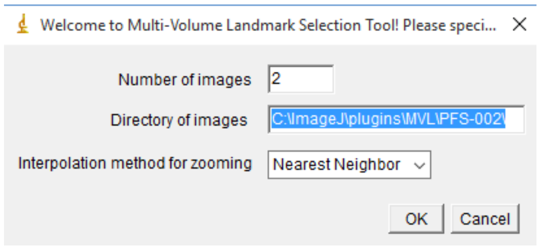
Look in: MVL

Name	Date modified	Type
MultiVolLmk$IxPointPair.class	2/16/2015 11:45 AM	CLASS
MultiVolLmk$IxPointPairScanner.class	2/16/2015 11:45 AM	CLASS
MultiVolLmk$LmkDistributionButtonHan...	1/22/2015 10:05 PM	CLASS
MultiVolLmk$LmkDistriButtonHandler.cl...	2/16/2015 11:45 AM	CLASS
MultiVolLmk$LmkRadioButtonHandler.cl...	2/16/2015 11:45 AM	CLASS
MultiVolLmk$PositionIndicator.class	2/16/2015 11:45 AM	CLASS
MultiVolLmk$VolDispCheckBoxHandler.c...	2/16/2015 11:45 AM	CLASS
MultiVolLmk.class	2/16/2015 11:45 AM	CLASS
✓ MultiVolLmk.java	2/16/2015 11:45 AM	JAVA F
MultiVolLmk_double_zoom.java	1/24/2015 3:52 PM	JAVA F
MyLmksBackUp.dat	10/13/2014 11:35 ...	DAT Fi
Warped Volume 20IN exp_kernel.img	7/31/2014 10:24 AM	Disc In

File name: MultiVolLmk.java
Files of type: All Files (*.*)

[2] In the pop-up dialogue box, input the number of volumes we want to label landmarks on. For our current step, just input “2”. The “Directory of images” is our folder “...\ImageJ\plugins\MVL\PFS-002\”. Note that you need to enter the full path of the folder containing the images and append “\” at the end of the path. The “Interpolation method for zooming” shall be set as “Nearest Neighbor” for our purpose.

Click “OK”.

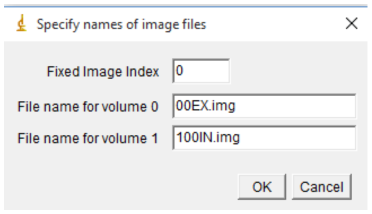

Welcome to Multi-Volume Landmark Selection Tool! Please speci...

Number of images: 2

Directory of images: C:\ImageJ\plugins\MVL\PFS-002\

Interpolation method for zooming: Nearest Neighbor

[3] In the next pop-up dialogue box, input the file names of the 8-bit images and specify the index of the fixed image (4DCT images are labeled by indices 0, 1, 2, ..., N-1 inside the program. We need specify to the index of the image that will be used as fixed image in the registration/normalization feature). In our case, we would use “00EX.img” (the extreme exhale phase) as fixed image and “100IN.img” (the extreme inhale phase) as moving image, and we shall set the file names accordingly. Then click “OK”.

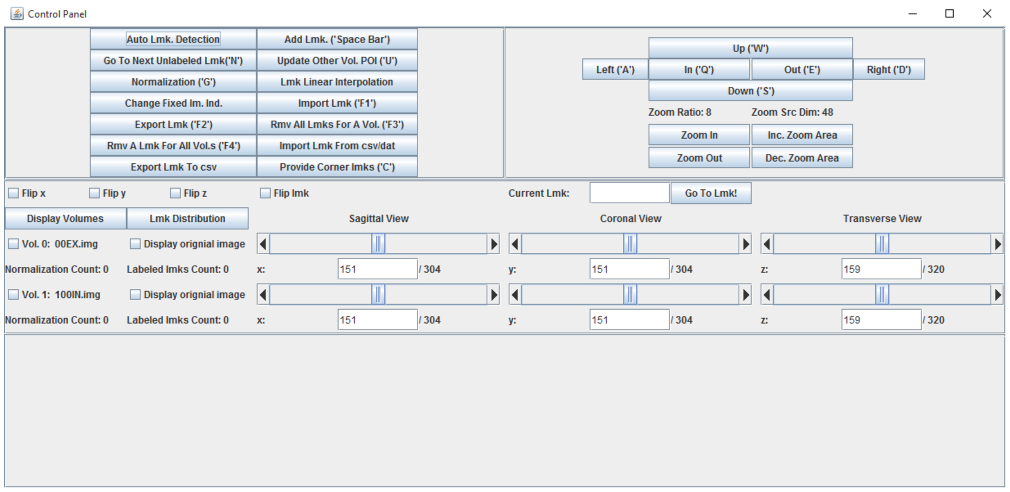

Specify names of image files

Fixed Image Index: 0

File name for volume 0: 00EX.img

File name for volume 1: 100IN.img

[4] The control panel and the display window will be generated.


Control Panel

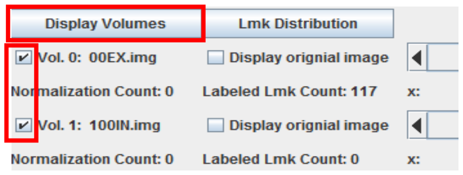
Zoom Ratio: 8 Zoom Src Dim: 48

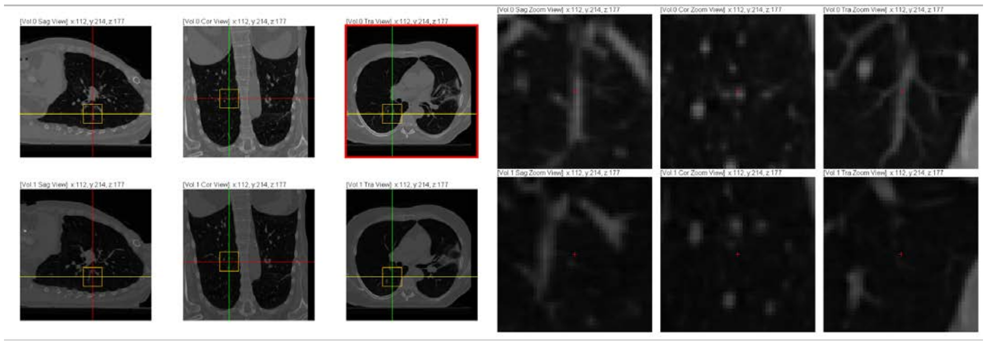
Current Lmk:

Vol. 0: 00EX.img
Normalization Count: 0 Labeled lmks Count: 0 x: 151 / 304 y: 151 / 304 z: 159 / 320

Vol. 1: 100IN.img
Normalization Count: 0 Labeled lmks Count: 0 x: 151 / 304 y: 151 / 304 z: 159 / 320

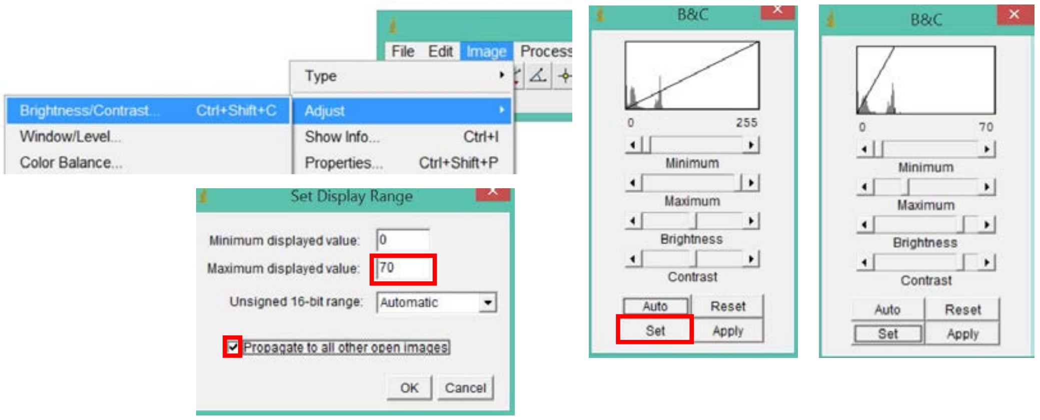
Now, select which images we want to display by checking the checkboxes to the left of the image file names and click "Display Volumes" to show the images.





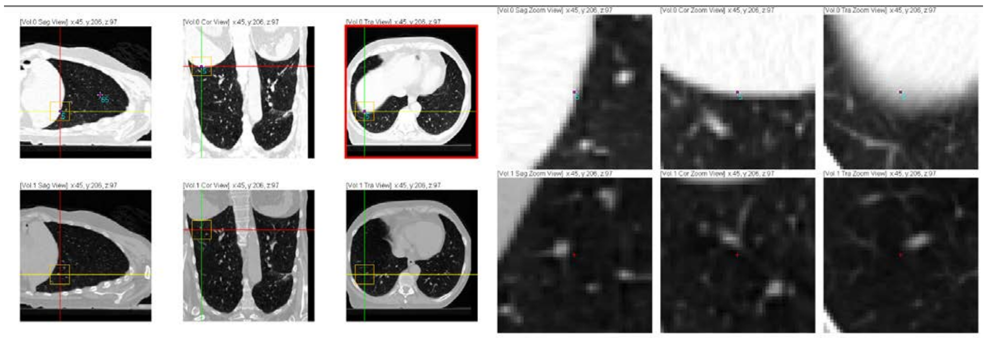
When displaying image volumes, without specific orientation information, different software may use different default orientations. And thus, landmarks on a specific image volume acquired using one software may have "flipped" coordinates with respect to landmarks obtained using another software on the same volume. The "flip x", "flip y", "flip z" and "flip lmk" features were implemented to accommodate for this difference. For now, these "flip" features will not be used in our inter-observer experiment.

We can see the images seem pretty dark, which is not helpful for the landmarking task. To improve the brightness and contrast of the image, click "Image"→"Adjust"→"Brightness/Contrast…" on the imageJ panel and a menu called "B&C" will pop-up. Click "Set" in the "B&C" menu. In the pop-up dialogue, set the "Maximum displayed value" to be around "70" and check "Propagate to all open Images" and click "OK".

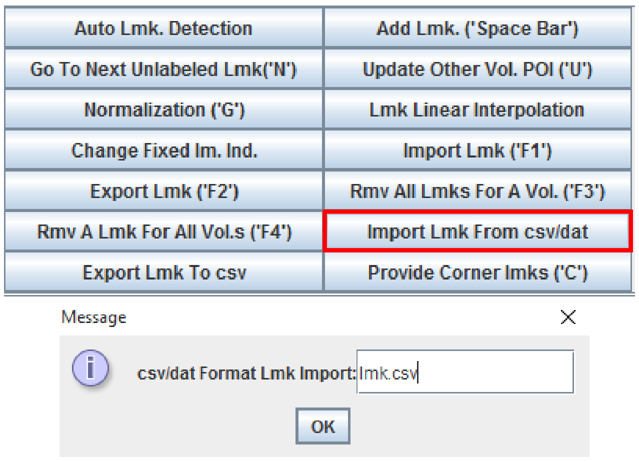


Minimum displayed value: 0
Maximum displayed value: 70
Unsigned 16-bit range: Automatic
[✓] Propagate to all other open images

The internal structures of the image will become a lot clearer to see.

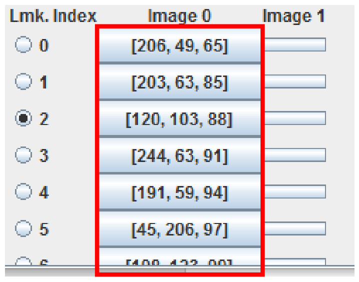


[5] Next, we will load in the landmarks stored in file "lmk.csv". Click "Import Lmk From csv/dat ". In the pop-up window, input in the file name "lmk.csv "and click "OK".



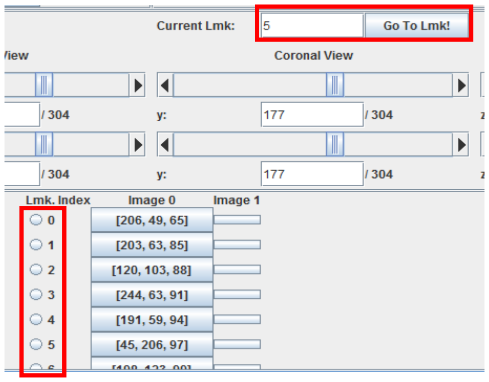
csv/dat Format Lmk Import: lmk.csv

In the control panel, we would notice that the landmarks have been loaded.



Lmk. Index	Image 0	Image 1
○ 0	[206, 49, 65]	
○ 1	[203, 63, 85]	
● 2	[120, 103, 88]	
○ 3	[244, 63, 91]	
○ 4	[191, 59, 94]	
○ 5	[45, 206, 97]	

You can click on the radio buttons to view specific landmarks. You can also input the landmark index and click “Go To Lmk!” to visit a specific landmark.



Current Lmk: 5 | Go To Lmk!

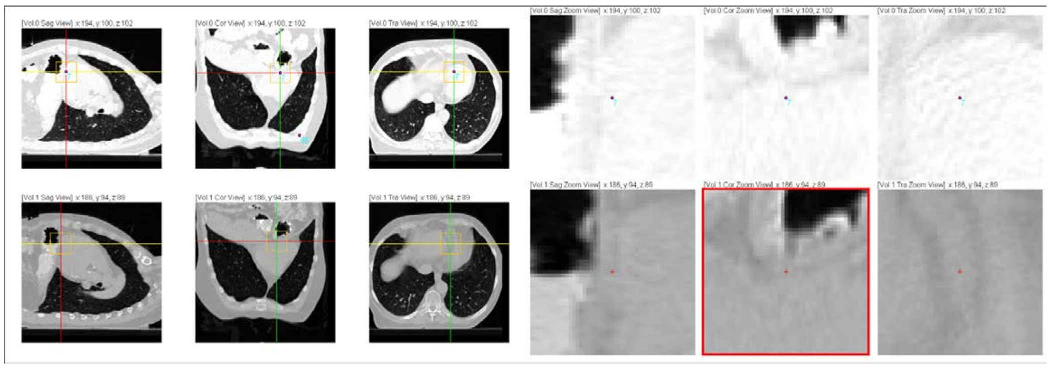
/iew | Coronal View

/ 304 | y: 177 / 304

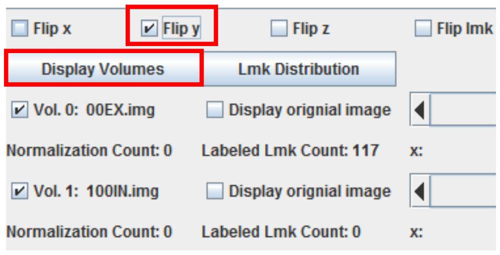
/ 304 | y: 177 / 304

Lmk. Index	Image 0	Image 1
○ 0	[206, 49, 65]	
○ 1	[203, 63, 85]	
○ 2	[120, 103, 88]	
○ 3	[244, 63, 91]	
○ 4	[191, 59, 94]	
○ 5	[45, 206, 97]	

However, as you may have noticed, in the image display panel, the landmark locations seem weird. Some of them are not even in the lung region. This is because the software that generated those automatically detected landmarks uses a different default orientation compared to the imageJ software. For this specific data set, we need to flip the Y orientation of our images to be compatible with the auto-detected landmarks.



Check the “flip Y” checkbox in the control panel and click “Display Volumes” again.



☐ Flip x ☑ Flip y ☐ Flip z ☐ Flip lmk

Display Volumes | Lmk Distribution

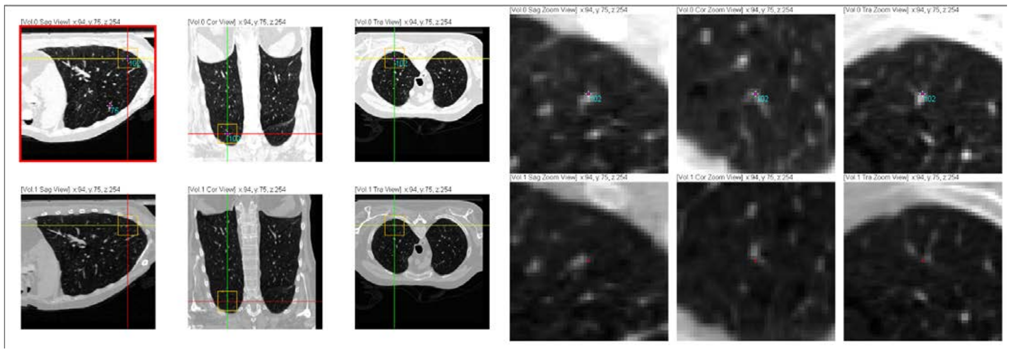
☑ Vol. 0: 00EX.img ☐ Display orignial image

Normalization Count: 0 Labeled Lmk Count: 117 x:

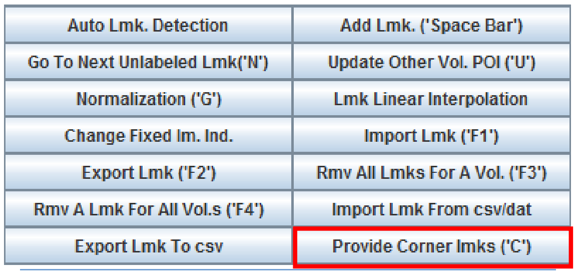
☑ Vol. 1: 100IN.img ☐ Display orignial image

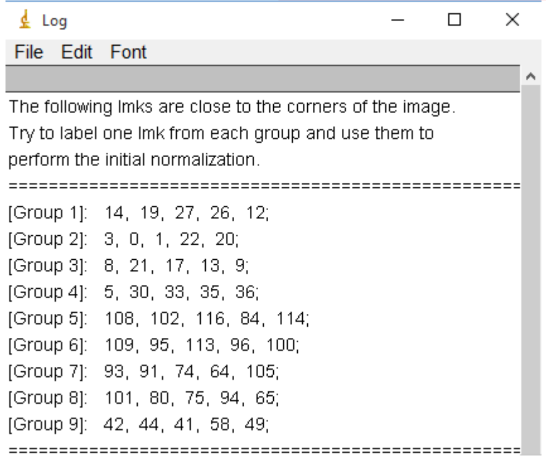
Normalization Count: 0 Labeled Lmk Count: 0 x:

The images will now be displayed with their Y dimension flipped to be compatible with the orientation of the landmarks. We can check and see that the landmarks make a lot more sense than previously.



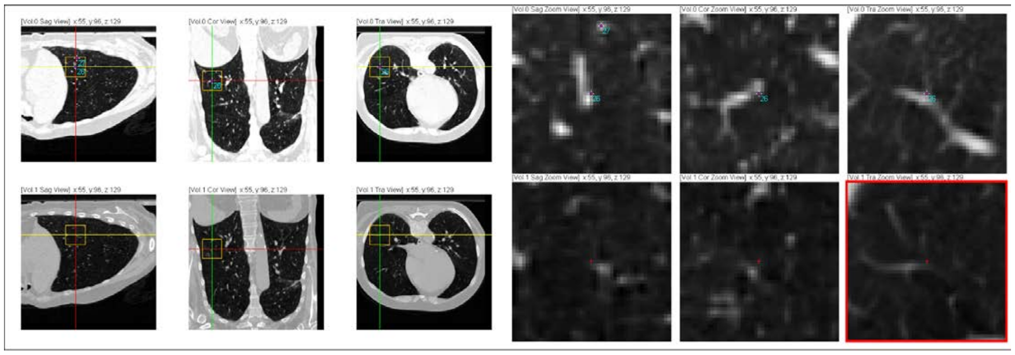
[6] Now, we can start labeling landmarks on the 8-bit 100IN phase image. Click on "Provide Corner lmks" button on the control panel. A message window will pop up, showing us several groups of indices of landamarks located around the corners and boundaries on the 00EX image.





```
The following lmks are close to the corners of the image.
Try to label one lmk from each group and use them to
perform the initial normalization.
==================================================
[Group 1]:   14,  19,  27,  26,  12;
[Group 2]:   3,  0,  1,  22,  20;
[Group 3]:   8,  21,  17,  13,  9;
[Group 4]:   5,  30,  33,  35,  36;
[Group 5]:   108,  102,  116,  84,  114;
[Group 6]:   109,  95,  113,  96,  100;
[Group 7]:   93,  91,  74,  64,  105;
[Group 8]:   101,  80,  75,  94,  65;
[Group 9]:   42,  44,  41,  58,  49;
==================================================
```

You are recommended to label one landmark from each of the groups of indices on the 100IN phase image. Some landmarks are easier to label than others, so if it's hard to find correspondence for one landmark in a certain group, just try to find the corresponding location for another landmark in the same group. You can start by clicking on the radio button for one of the landmarks in Group 1, say, landmark #26. You will be taken to the location of the #26 landmark on the 00EX image and you will be taken to the "suggested" location for the corresponding landmark on the 100IN image.



At first, the "suggested" landmark locations will be pretty inaccurate, because it will just be the same set of coordinates as the landmark on the 00EX phase. But as you label more and more landmarks and use the "Normalization" a couple of times, the suggested landmark locations will be more and more accurate.

But for now, you need to navigate through the 100IN phase image, and try your best to locate the corresponding point with respect to landmark #26 on 00EX phase image. You need to have an anatomical view that you are currently focused on. Within this view you can easily traverse "up", "down", "left", "right", "in" or "out" by pressing keyboard buttons "w", "s", "a", "d", "q" or "e" (you can also scroll the middle mouse button to go in and out). You can also switch between different anatomical views by pressing "i", "j", "k" and "l". The currently focused-on view will have a red box around it.

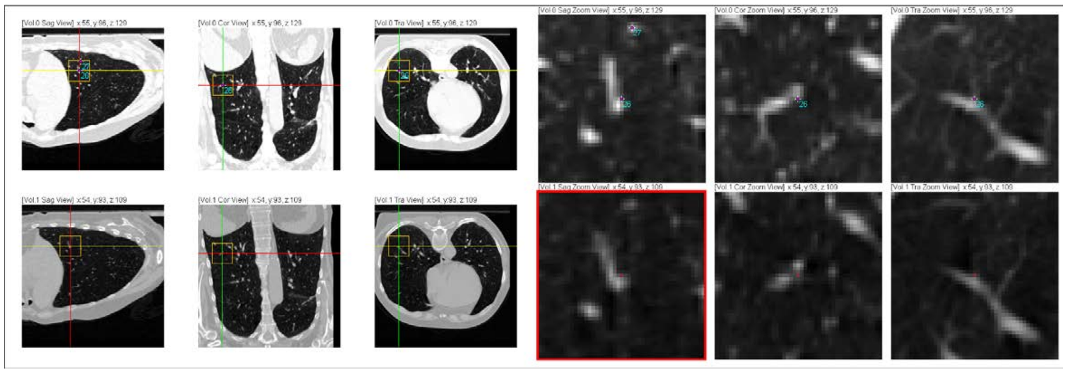
When trying to find the corresponding landmark locations, be sure to take all three anatomical views into account, and also remember to look at both the standard view and the zoomed view for correspondence. Most of the times, it would be impossible to have a perfect match in shape for all three views. So you must decide which point is the overall best fit.

After navigating through the 100IN phase image volume, hopefully you will end up with an ideal location to be set as the corresponding landmark location on 100IN for landmark #26.

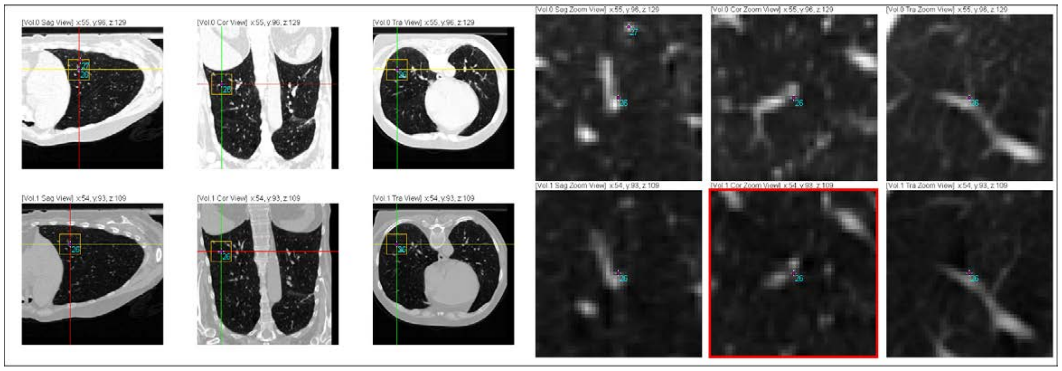
Note, once you have clicked a radio button to the left of a landmark index or input a landmark index and clicked the "Go To Lmk!" button, you will automatically enter the landmark selection mode. This means, if you go to any point in whichever image volume in the image display panel and press "Space Bar", that point will be selected as the corresponding landmark point for that landmark index in that image volume. If the location of a landmark index has previously been selected in an image volume, and you clicked on another position within this volume and pressed "Space Bar", then that landmark index will be updated to have this new location in this volume. If the location for a landmark index has

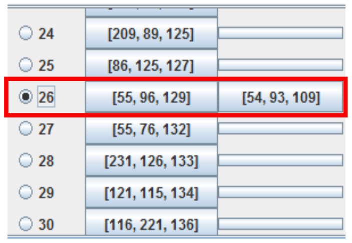
not been chosen in an image volume, clicking on a location within this volume and press “Space Bar” will simply set the location for this landmark index in this volume.

The following is my choice of landmark #26 on 100IN phase image. You can see that this location achieves relatively good match on two of the three anatomical views. But in the middle view (coronal), the shape does not match too well. But upon further experiment, you will find that any other point will give even worse correspondence. So this location would be our choice.


[Vol 0 Sag View] x:55, y:96, z:129
[Vol 0 Cor View] x:55, y:96, z:129
[Vol 0 Tra View] x:55, y:96, z:129
[Vol 0 Sag Zoom View] x:55, y:96, z:129
[Vol 0 Cor Zoom View] x:55, y:96, z:129
[Vol 0 Tra Zoom View] x:55, y:96, z:129
[Vol 1 Sag View] x:54, y:93, z:109
[Vol 1 Cor View] x:54, y:93, z:109
[Vol 1 Tra View] x:54, y:93, z:109
[Vol 1 Sag Zoom View] x:54, y:93, z:109
[Vol 1 Cor Zoom View] x:54, y:93, z:109
[Vol 1 Tra Zoom View] x:54, y:93, z:109

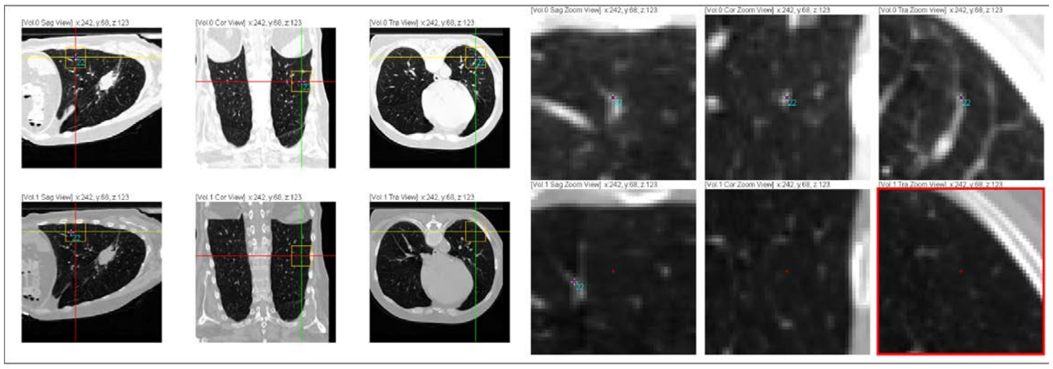
Now, press “Space Bar” on the keyboard, and this point will be selected as the corresponding point for landmark #26 on 100IN phase image.


[Vol 0 Sag View] x:55, y:96, z:129
[Vol 0 Cor View] x:55, y:96, z:129
[Vol 0 Tra View] x:55, y:96, z:129
[Vol 0 Sag Zoom View] x:55, y:96, z:129
[Vol 0 Cor Zoom View] x:55, y:96, z:129
[Vol 0 Tra Zoom View] x:55, y:96, z:129
[Vol 1 Sag View] x:54, y:93, z:109
[Vol 1 Cor View] x:54, y:93, z:109
[Vol 1 Tra View] x:54, y:93, z:109
[Vol 1 Sag Zoom View] x:54, y:93, z:109
[Vol 1 Cor Zoom View] x:54, y:93, z:109
[Vol 1 Tra Zoom View] x:54, y:93, z:109

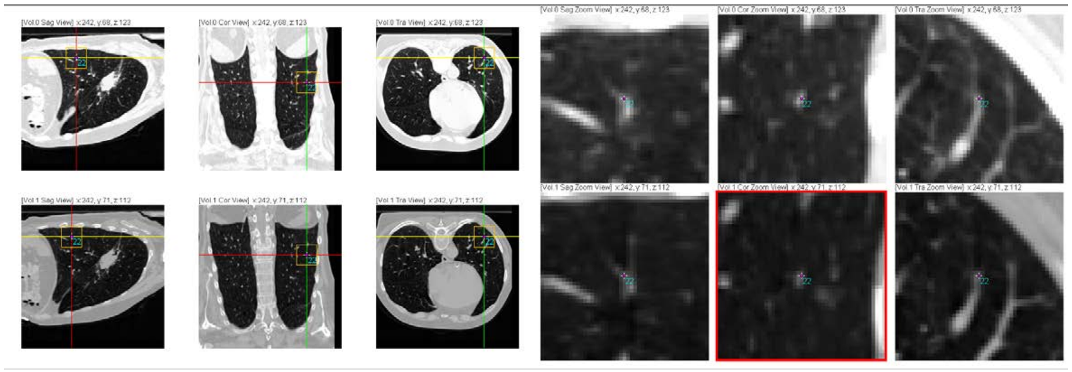


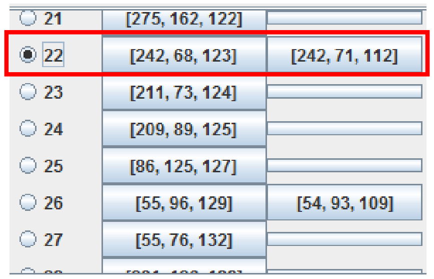
○ 24	[209, 89, 125]	
○ 25	[86, 125, 127]	
● 26	[55, 96, 129]	[54, 93, 109]
○ 27	[55, 76, 132]	
○ 28	[231, 126, 133]	
○ 29	[121, 115, 134]	
○ 30	[116, 221, 136]	

[7] Next, let’s proceed to another landmark within the second group of “corner landmarks”, say #22. Again, the suggested landmark location is less than ideal.


[Vol 0 Sag View] x:242, y:68, z:123
[Vol 0 Cor View] x:242, y:68, z:123
[Vol 0 Tra View] x:242, y:68, z:123
[Vol 0 Sag Zoom View] x:242, y:68, z:123
[Vol 0 Cor Zoom View] x:242, y:68, z:123
[Vol 0 Tra Zoom View] x:242, y:68, z:123
[Vol 1 Sag View] x:242, y:68, z:123
[Vol 1 Cor View] x:242, y:68, z:123
[Vol 1 Tra View] x:242, y:68, z:123
[Vol 1 Sag Zoom View] x:242, y:68, z:123
[Vol 1 Cor Zoom View] x:242, y:68, z:123
[Vol 1 Tra Zoom View] x:242, y:68, z:123

After navigating through the volume 100IN, we can find the best fit, and press “Space Bar” to select it.

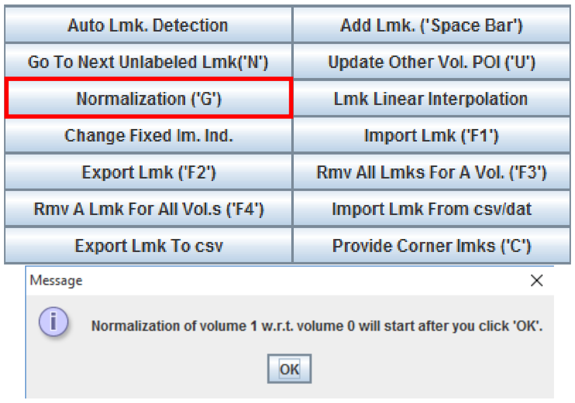

[Vol 0 Sag View] x:242, y:68, z:123
[Vol 0 Cor View] x:242, y:68, z:123
[Vol 0 Tra View] x:242, y:68, z:123
[Vol 0 Sag Zoom View] x:242, y:68, z:123
[Vol 0 Cor Zoom View] x:242, y:68, z:123
[Vol 0 Tra Zoom View] x:242, y:68, z:123
[Vol 1 Sag View] x:242, y:71, z:112
[Vol 1 Cor View] x:242, y:71, z:112
[Vol 1 Tra View] x:242, y:71, z:112
[Vol 1 Sag Zoom View] x:242, y:71, z:112
[Vol 1 Cor Zoom View] x:242, y:71, z:112
[Vol 1 Tra Zoom View] x:242, y:71, z:112



○ 21	[275, 162, 122]	
● 22	[242, 68, 123]	[242, 71, 112]
○ 23	[211, 73, 124]	
○ 24	[209, 89, 125]	
○ 25	[86, 125, 127]	
○ 26	[55, 96, 129]	[54, 93, 109]
○ 27	[55, 76, 132]	

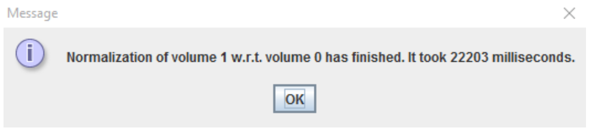
Repeat this process, until you have labeled one corresponding landmark from each of the groups of "corner landmarks".

[8] Now is when the magic starts to happen. After you have labeled one corner landmark from each group, you can click on the "Normalization" button on the control panel. You will be prompted with a message saying that the normalization that tries to align 100IN phase with respect to 00EX phase based on the landmarks you have labeled will start. Click "OK"



Normalization of volume 1 w.r.t. volume 0 will start after you click 'OK'.

After a short while, you will be notified that the normalization has finished. Click OK.

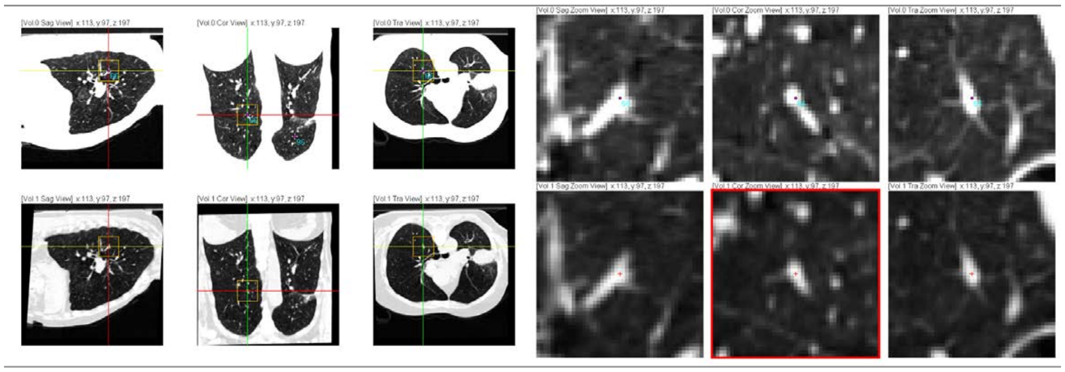


Normalization of volume 1 w.r.t. volume 0 has finished. It took 22203 milliseconds.

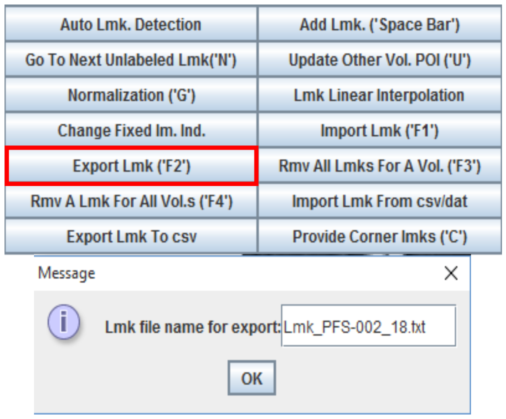
Now, if you watch closely, you would find that the 100IN phase image being displayed has changed a little. This is because the normalization feature made the 100IN phase image more similar to the 00EX phase image. In general, the suggested landmark location for any further landmarks to be labeled will also be a little more accurate, which means you will take less effort to find a corresponding point based on the suggested landmark locations.

[9] Repeat the process in step [7], and label another set of landmarks from the groups. Then use the "normalization" feature once again to even better align the 100IN phase image to the 00EX phase image.

After repeating this process of labeling a few landmarks (around 10) and performing normalization based on labeled landmarks for 3 to 4 times, the 100IN phase image would look quite similar to the 00EX phase image, and the suggested landmark locations would be quite accurate, meaning you only have to navigate through the volume a little until you reach an ideal corresponding point. At this stage, you can ignore the provided groups of "corner landmarks" and just label whatever landmarks you want. If you feel the suggested location is still not accurate enough, you can perform more normalizations once you labeled more landmarks. If you feel the suggested locations are already sufficiently good, you can just keep on labeling the remaining landmarks without using normalization any more. We can see in the figure below, that the suggested landmark locations are already pretty good, after performing two normalizations.

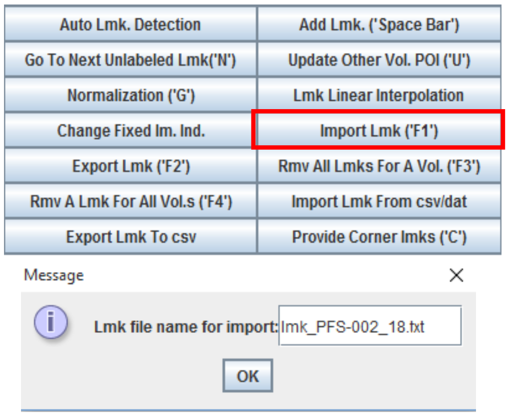


During the landmark labeling process, you can store your labeled landmarks at any time. Click the "export lmk" button in the control panel, give a name to the landmark file, and save it in ".txt" format. This way, you can label some corresponding landmarks, save your progress and return to the landmarking task whenever you want.



Lmk file name for export: Lmk_PFS-002_18.txt

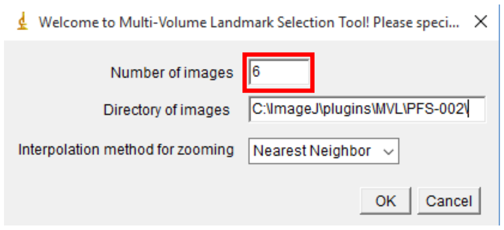
When you want to resume landmarking, just re-open the software, reload and display the images, adjust the brightness and contrast, flip the images in Y direction and import your previously label landmarks. To do the import, click the "import lmk" button in the control panel, and input the ".txt" landmark file name you previously exported and click OK. Note: don't use "Import Lmk From csv/dat" for this purpose.



Lmk file name for import: lmk_PFS-002_18.txt

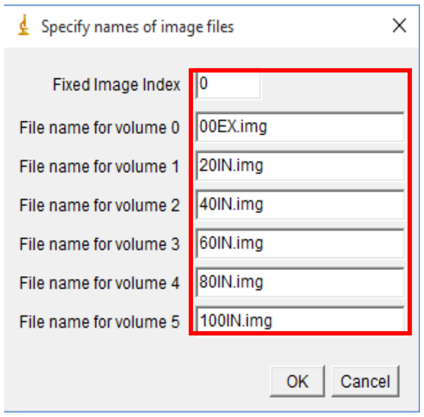
[10] In the end, you will have labeled all corresponding landmarks on 100IN and 00EX phase images. Next, you need to export the landmarks on these two extreme phases into a txt file, e.g. "lmk_extremes.txt". And close the software.

[11] Now, we shall label landmarks on the intermediate phases. We will first label all the inhale phases and then all the exhale phases. So, re-open the software, but this time choose number of images to be 6.


Welcome to Multi-Volume Landmark Selection Tool! Please speci...
OK
Cancel

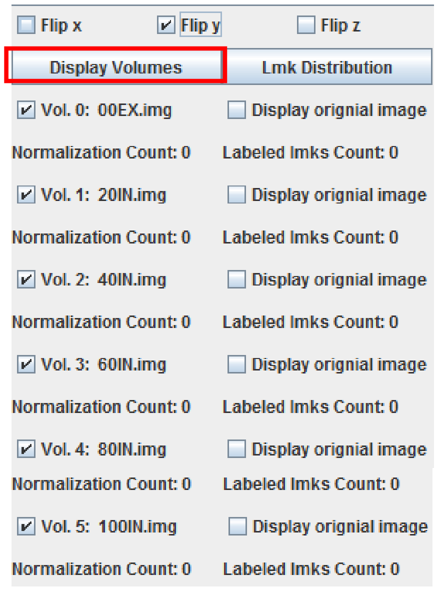
Number of images: 6
Directory of images: C:\ImageJ\plugins\MVL\PFS-002\
Interpolation method for zooming: Nearest Neighbor

Fill in the image file names and click OK:



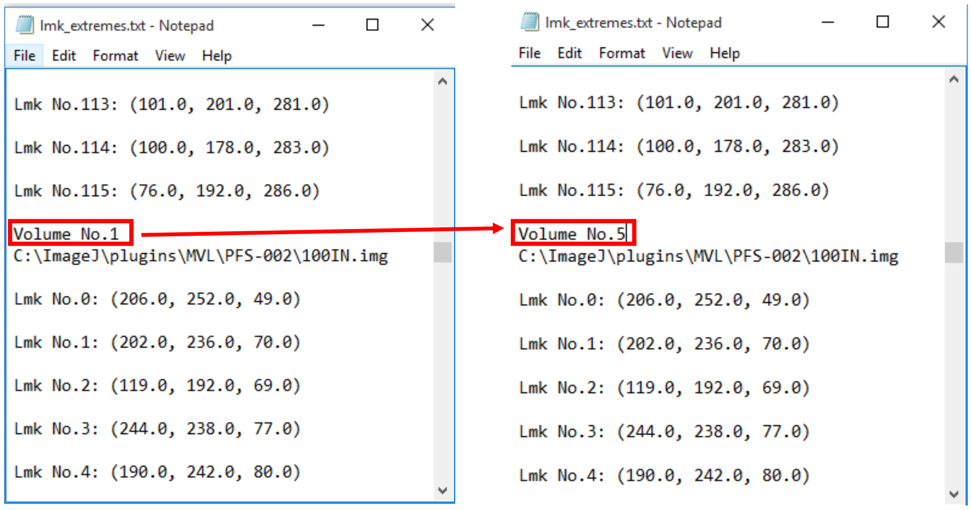
Fixed Image Index: 0
File name for volume 0: 00EX.img
File name for volume 1: 20IN.img
File name for volume 2: 40IN.img
File name for volume 3: 60IN.img
File name for volume 4: 80IN.img
File name for volume 5: 100IN.img

[12] Display all 6 images, flip the images in Y direction, and adjust brightness and contrast.



✓ Vol. 0: 00EX.img	Display orignial image
Normalization Count: 0	Labeled lmks Count: 0
✓ Vol. 1: 20IN.img	Display orignial image
Normalization Count: 0	Labeled lmks Count: 0
✓ Vol. 2: 40IN.img	Display orignial image
Normalization Count: 0	Labeled lmks Count: 0
✓ Vol. 3: 60IN.img	Display orignial image
Normalization Count: 0	Labeled lmks Count: 0
✓ Vol. 4: 80IN.img	Display orignial image
Normalization Count: 0	Labeled lmks Count: 0
✓ Vol. 5: 100IN.img	Display orignial image
Normalization Count: 0	Labeled lmks Count: 0

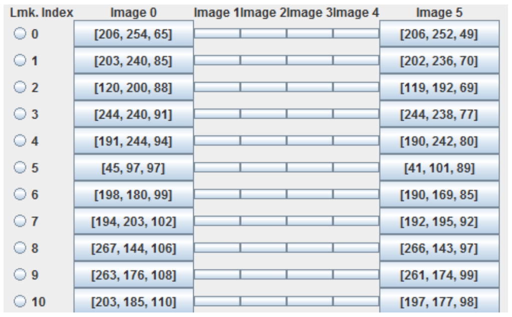
[13] Manually modify the previously labeled landmark file "lmk_extremes.txt" by changing the volume index of 100IN phase image from "1" to "5", because the internal image volume index for 100IN phase is "5" in current stage.



```
Lmk No.113: (101.0, 201.0, 281.0)

Lmk No.114: (100.0, 178.0, 283.0)

Lmk No.115: (76.0, 192.0, 286.0)

Volume No.1
C:\ImageJ\plugins\MVL\PFS-002\100IN.img

Lmk No.0: (206.0, 252.0, 49.0)

Lmk No.1: (202.0, 236.0, 70.0)

Lmk No.2: (119.0, 192.0, 69.0)

Lmk No.3: (244.0, 238.0, 77.0)

Lmk No.4: (190.0, 242.0, 80.0)
```

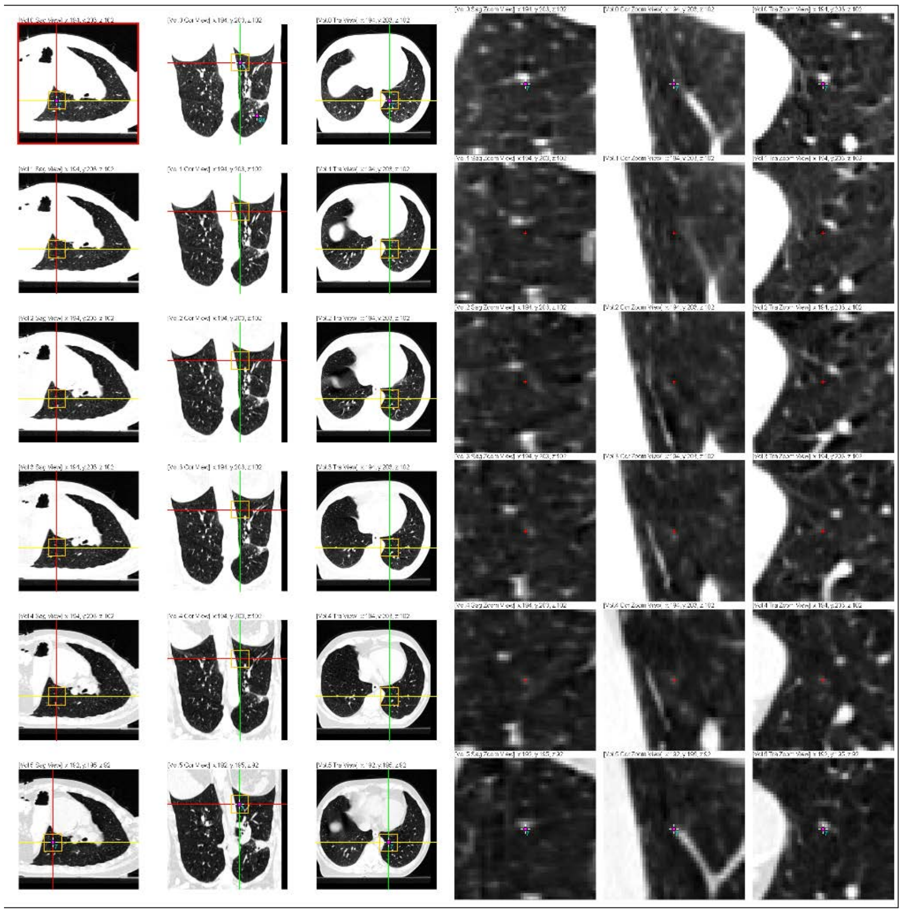
```
Lmk No.113: (101.0, 201.0, 281.0)

Lmk No.114: (100.0, 178.0, 283.0)

Lmk No.115: (76.0, 192.0, 286.0)

Volume No.5
C:\ImageJ\plugins\MVL\PFS-002\100IN.img

Lmk No.0: (206.0, 252.0, 49.0)

Lmk No.1: (202.0, 236.0, 70.0)

Lmk No.2: (119.0, 192.0, 69.0)

Lmk No.3: (244.0, 238.0, 77.0)

Lmk No.4: (190.0, 242.0, 80.0)
```

Import the landmark file "lmk_extremes.txt" and you will notice in the control panel, all previous landmarks are imported.

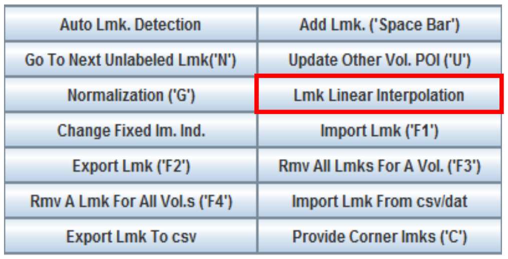


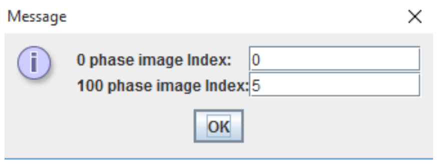
Lmk. Index	Image 0	Image 1	Image 2	Image 3	Image 4	Image 5
0	[206, 254, 65]					[206, 252, 49]
1	[203, 240, 85]					[202, 236, 70]
2	[120, 200, 88]					[119, 192, 69]
3	[244, 240, 91]					[244, 238, 77]
4	[191, 244, 94]					[190, 242, 80]
5	[45, 97, 97]					[41, 101, 89]
6	[198, 180, 99]					[190, 169, 85]
7	[194, 203, 102]					[192, 195, 92]
8	[267, 144, 106]					[266, 143, 97]
9	[263, 176, 108]					[261, 174, 99]
10	[203, 185, 110]					[197, 177, 98]

Now the display image window will look like this:

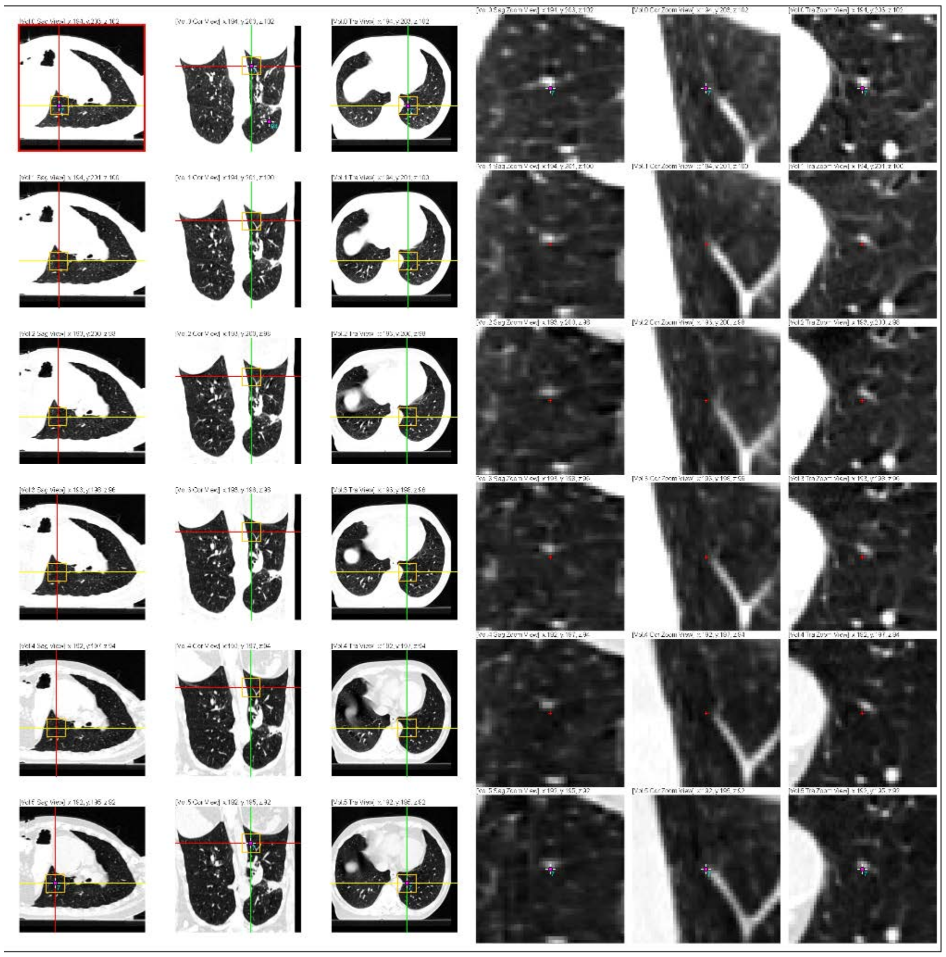


[14] Click the “Lmk Linear Interpolation” button on the control panel, input the corresponding image volume indices of 00EX and 100IN phase images, and click OK. Then, the suggested landmark locations on the intermediate phases will be generated.





0 phase image Index: 0
100 phase image Index: 5



We can clearly see that after using the landmark location linear interpolation feature, the suggested landmark locations on the intermediate phases become a lot more meaningful. And you only need to fine-tune the location a little bit to get the ideal correspondence.

[15] Finish labeling all the landmarks on all the intermediate phases based on the suggested locations.

[16] Export the landmarks into file, e.g., "lmk_extremes_inhale.txt".

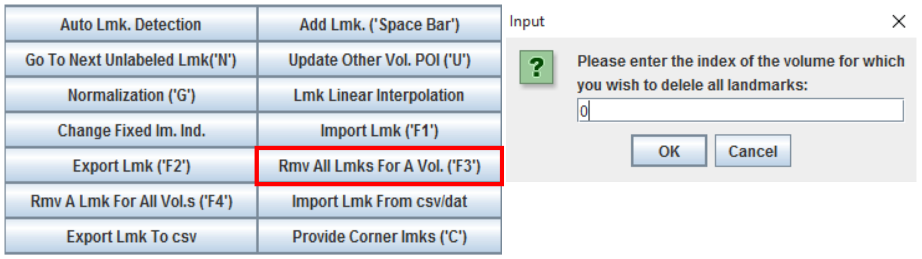
[17] Close the program, load in the intermediate exhale phase images and import the "lmk_extreme.txt" landmark file again.

[18] Label all landmarks on the intermediate exhale phases and export the landmarks into file, e.g., "lmk_extremes_exhale.txt"

Congratulations! You are almost done…

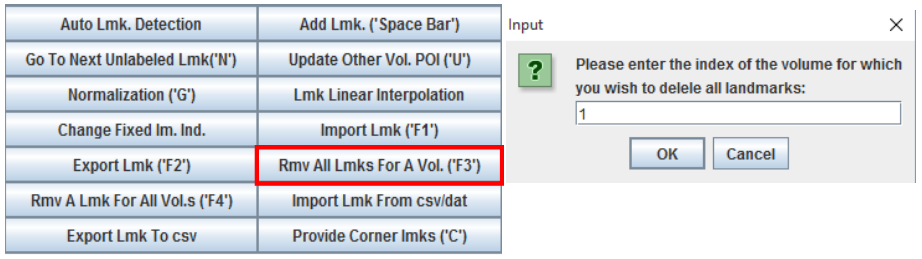
[10.5] As you may have noticed, the accuracy of the 100IN phase landmarks is very important, not only for its own sake but also for labeling landmarks on all other intermediate phases. So, it's necessary to make sure that the corresponding landmarks labeled on 100IN phase is as accurate as possible.

It's highly recommended that after you finish labeling the landmarks on 100IN phase and export the landmarks into "lmk_extremes.txt" file (steps **[1]** ~ **[10]**), you clear the already labeled landmarks by clicking "Remove All Lmk For A Vol" button in the control panel. Input the image index 0 to remove all landmarks on the 00EX phase. Then, click the "Remove All Lmk For A Vol" button again and input image index 1 to remove all landmarks on the 100IN phase.



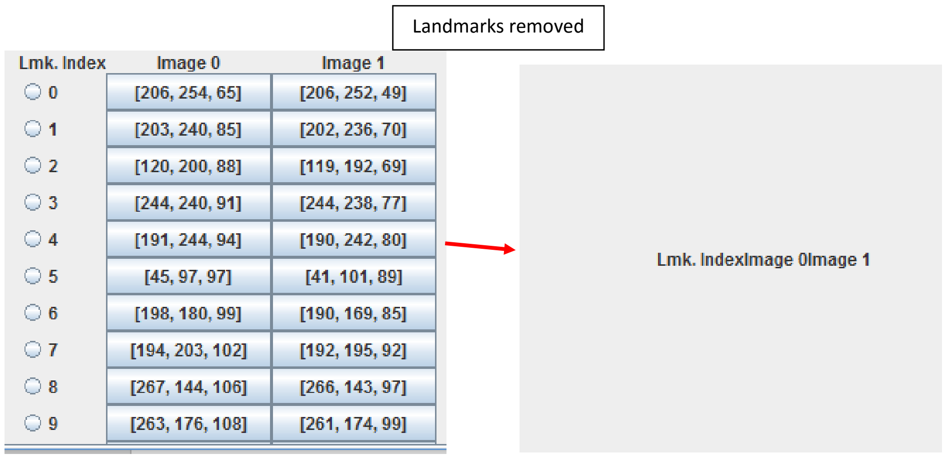
Please enter the index of the volume for which you wish to delele all landmarks:

0



Please enter the index of the volume for which you wish to delele all landmarks:

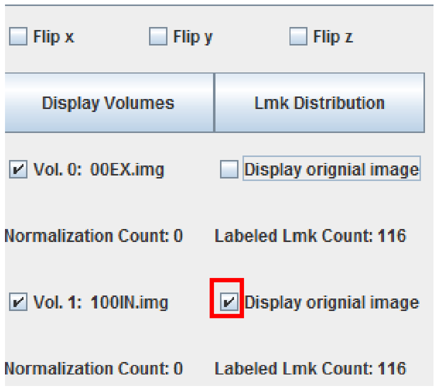
1


Landmarks removed

Lmk. Index	Image 0	Image 1
0	[206, 254, 65]	[206, 252, 49]
1	[203, 240, 85]	[202, 236, 70]
2	[120, 200, 88]	[119, 192, 69]
3	[244, 240, 91]	[244, 238, 77]
4	[191, 244, 94]	[190, 242, 80]
5	[45, 97, 97]	[41, 101, 89]
6	[198, 180, 99]	[190, 169, 85]
7	[194, 203, 102]	[192, 195, 92]
8	[267, 144, 106]	[266, 143, 97]
9	[263, 176, 108]	[261, 174, 99]

Lmk. IndexImage 0Image 1

Then check the “Display original image” checkbox for 100IN phase image and click “Display Volume”.



Vol. 0: 00EX.img [✓] | Display orignial image [ ]

Normalization Count: 0 | Labeled Lmk Count: 116

Vol. 1: 100IN.img [✓] | Display orignial image [✓]

Normalization Count: 0 | Labeled Lmk Count: 116

This will void all the previous normalization attempts to make the 100IN phase image look similar to the 00EX phase image, and display the 100IN phase image in its vanilla shape. Now, you can import the “lmk_extremes.txt” landmark files and re-check the corresponding landmarks you labeled on the original un-deformed 100IN phase image. If any landmark needs to be modified, just click on the radio button to the left of the landmark index or input the landmark index and click the “Go To Lmk!” button. Then click on or navigate to the modified landmark location and press “Space Bar” to update the landmark location.

REFERENCES

[1] Serdar K Balci, Polina Golland, Martha Shenton, and William M Wells. Free-form b-spline deformation model for groupwise registration. In *Medical image computing and computer-assisted intervention: MICCAI... International Conference on Medical Image Computing and Computer-Assisted Intervention*, volume 10, page 23. NIH Public Access, 2007.

[2] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):567–585, 1989.

[3] G. E. Christensen and H. J. Johnson. Consistent image registration. 20(7):568–582, July 2001.

[4] Gary E Christensen, Joo Hyun Song, Wei Lu, Issam El Naqa, and Daniel A Low. Tracking lung tissue motion and expansion/compression with inverse consistent image registration and spirometry. *Medical Physics*, 34(6):2155–2163, 2007.

[5] Stella Flampouri, Steve B Jiang, Greg C Sharp, John Wolfgang, Abhijit A Patel, and Noah C Choi. Estimation of the delivered patient dose in lung IMRT treatment based on deformable registration of 4d-ct data and monte carlo simulations. *Physics in Medicine and Biology*, 51(11):2763, 2006.

[6] Vladlena Gorbunova, Jon Sporring, Pechin Lo, Martine Loeve, Harm A Tiddens, Mads Nielsen, Asger Dirksen, and Marleen de Bruijne. Mass preserving image registration for lung ct. *Medical Image Analysis*, 16(4):786–795, 2012.

[7] Sarang C Joshi and Michael I Miller. Landmark matching via large deformation diffeomorphisms. *Image Processing, IEEE Transactions on*, 9(8):1357–1370, 2000.

[8] Stefan Klein, Marius Staring, Keelin Murphy, Max A Viergever, and Josien PW Pluim. Elastix: A toolbox for intensity-based medical image registration. *IEEE Transactions on Medical Imaging*, 29(1):196–205, 2010.

[9] Guang Li, Deborah Citrin, Kevin Camphausen, Boris Mueller, Chandra Burman, Borys Mychalczak, Robert W Miller, and Yulin Song. Advances in 4d medical imaging and 4d radiation therapy. *Technology in cancer research & treatment*, 7(1):67–81, 2008.

[10] Jamie R McClelland, Jane M Blackall, Ségolene Tarte, Adam C Chandler, Simon Hughes, Shahreen Ahmad, David B Landau, and David J Hawkes. A continuous 4d motion model from multiple respiratory cycles for use in lung radiotherapy. *Medical Physics*, 33(9):3348–3358, 2006.

[11] CT Metz, Stefan Klein, Michiel Schaap, Theo van Walsum, and Wiro J Niessen. Nonrigid registration of dynamic medical imaging data using nd+t b-splines and a groupwise optimization approach. *Medical Image Analysis*, 15(2):238–249, 2011.

[12] Keelin Murphy, Bram van Ginneken, Josien Pluim, Stefan Klein, and Marius Staring. Semi-automatic reference standard construction for quantitative evaluation of lung ct registration. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2008*, pages 1006–1013, 2008.

[13] J. M. Reinhardt, K. Ding, K. Cao, G. E. Christensen, E. A. Hoffman, and S. V. Bodas. Registration-based estimates of local lung tissue expansion compared to xenon-CT measures of specific ventilation. *Medical Image Analysis*, 12(6):752–763, 2008.

[14] 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. *IEEE Transactions on Medical Imaging*, 18(8):712–721, 1999.

[15] Michiel Schaap, Coert T Metz, Theo van Walsum, Alina G van der Giessen, Annick C Weustink, Nico R Mollet, Christian Bauer, Hrvoje Bogunović, Carlos Castro, Xiang Deng, et al. Standardized evaluation methodology and reference database for evaluating coronary artery centerline extraction algorithms. *Medical Image Analysis*, 13(5):701–714, 2009.

[16] Hari Sundar, Harold Litt, and Dinggang Shen. Estimating myocardial motion by 4d image warping. *Pattern Recognition*, 42(11):2514–2526, 2009.

[17] Jef Vandemeulebroucke, Simon Rit, Jan Kybic, Patrick Clarysse, and David Sarrut. Spatiotemporal motion estimation for respiratory-correlated imaging of the lungs. *Medical Physics*, 38(1):166–178, 2011.

[18] Sarah Gerard Yeary, Gary E. Christensen, John E. Bayouth, Sandeep Bodduluri, Yue Pan, Junfeng Guo, Kaifang Du, Joo H. Song, Bowen Zhao, Ipek Oguz, and Joseph M. Reinhardt. 4D lung CT segmentation for radiation therapy applications. In *ICART: Imaging and Computer Assistance in Radiation Therapy*, pages 50–57, 2015.

[19] Mehmet Yigitsoy, Christian Wachinger, and Nassir Navab. Temporal groupwise registration for motion modeling. In *Information Processing in Medical Imaging*, pages 648–659. Springer, 2011.

[20] Y. Yin, E. A. Hoffman, and C.-L. Lin. Mass preserving non-rigid registration of CT lung images using cubic B-spline. *Medical Physics*, 36(9):4213–4222, 2009.