Incorporating Metric Flows and Sparse Jacobian Transformations in ITK

Release 0.10

Mathieu De Craene[1], Aloys du Bois d'Aische[1], Benoît Macq[1], Simon K. Warfield[2,3]

March 8, 2006

1. Communications and Remote Sensing Laboratory,
Université catholique de Louvain, Louvain-la-Neuve, BELGIUM.
decraene@tele.ucl.ac.be
2. Surgical Planning Laboratory and
3. Computational Radiology Laboratory,
Brigham and Women's Hospital, Harvard Medical School, Boston MA, USA.

Abstract

Various metrics have been proposed in the literature for performing intrinsic automatic image to image registration. Among these measures, mutual information is a very popular one because of its robustness and accuracy for a wide variety of applications. In this paper, we propose a filter for performing non-rigid registration by estimating a dense deformation field derived from the mutual information metric. This filter takes place in the ITK PDE deformable registration design like the Demons algorithm of Thirion. We also show how the concept of metric flow is conceptually linked to the concept of metric derivative for a prior transformation model by the transformation jacobian. We also suggest a sparse implementation of the `GetJacobian()` method for reducing the computation time of a metric derivative for local transformations models.

Contents

1 Introduction

Intrinsic registration[8] refers to automatic registration techniques using the images signal intensities to compute the best alignment. In this paper, we will focus the case of *global similarity metrics* considering the mapping of the entire fixed image to moving image domains.

For maximizing these kind of metrics, the optimization strategy varies from the representation of the transformation, either parametric or non-parametric. In the case of a parametric representation, a finite number of parameters allows to represent rigid as well as non-rigid transformations with reasonable complexity. In this context, a broad range of numerical optimization schemes can be used (gradient based, stochastic search (SPSA[11], One Plus One[12], Powell[7], . . .).

When it comes to estimate a dense deformation field where the displacement vectors at each voxel are considered separately, the dimension of the optimization space tends to be so big that a variational method is often the privileged approach in the literature. The most popular variational registration technique is the optical flow[14] algorithm and has been widely used for intra-subject and atlas to patient registration. More recently, Hermosillo[5] has described a generalization of the optical flow algorithm for different well know metrics in medical images registration (including mean squared difference, mutual information and normalized correlation). In this paper, we first recall in Section 2 the main concepts required for computing the flow of a general metric. Section 3 explains how the concept of flow and the concept of derivative for a given transformation model are related. Section 4 proposes to include the concepts seen in the previous sections in ITK using helper classes to avoid copies of the same code in the toolkit. Finally, Section 5 illustrate the use of mutual information flow with or without prior regularization on clinical some clinical cases.

2 Metric flow

If no specific model is chosen for the transformation maximizing the global similarity between the two images to align, a common requirement is to search for a displacement field $\mathbf{u}$ making the cost function $\mathcal{I}$ stationary [5]. The problem is then reformulated as finding the displacement field $\mathbf{u}$ such that

$$\left.\frac{\partial \mathcal{I}(\mathbf{u}+\epsilon\mathbf{h})}{\partial \epsilon}\right|_{\epsilon=0} = 0, \quad \forall \mathbf{h} \tag{1}$$

Equation (1) means that for $\mathbf{u}$ optimal, adding any continuous function $\mathbf{h}$ scaled by an $\epsilon \in \mathbb{R}$ parameter has an optimal solution for $\epsilon = 0$.

For simplification purposes, we will consider in this paper the simple case of a cost function which does not contain a regularization term. The cost function can thus be written in a general way as

$$\mathcal{I}(\mathbf{u}) = \frac{1}{|\Omega|} \int_{\Omega} \phi(\mathbf{x}, \mathbf{u}(\mathbf{x}) \mathrm{d}\mathbf{x} \tag{2}$$

A regularization term would include in Equation (2) a dependency regarding the spatial derivatives of the displacement field.

The derivative in Equation (1) can be written like a scalar product between the perturbation function $\mathbf{h}$ and the partial derivative of ϕ regarding $\mathbf{u}$ [5]

$$\left.\frac{\partial \mathcal{I}(\mathbf{u} + \epsilon \mathbf{h})}{\partial \epsilon}\right|_{\epsilon=0} = \frac{1}{|\Omega|} \int_{\Omega} \mathbf{h}(\mathbf{x}) \cdot \phi_{,u}(\mathbf{x}) \mathrm{d}\mathbf{x} = 0 \quad \forall \mathbf{h} \tag{3}$$

Since Equation (3) must hold $\forall h$, the displacement field u making the cost functional stationary must satisfy the following condition on Ω

$$\text{Metric flow} \stackrel{\text{def}}{=} \phi_{,u}(\mathbf{x}) = 0 \quad \forall \mathbf{x} \in \Omega \tag{4}$$

The most common way of solving Equation (4) is a gradient descent scheme computing at each iteration an incremental displacement field that is then added to the current displacement field : $u_{t+1} = u_t + \Delta u_t$ with

$$\Delta u_t = -\eta \cdot \nabla_u \mathcal{I} \tag{5}$$

where η is a learning rate also referred to as *time step* in the ITK [6] implementation.

2.1 Application to the mean square error metric

The simple case of a mean square error metric can be considered in a first time for applying the concepts of Section 2. In this case the metric can written as

$$MSE(\mathbf{u}) = \frac{1}{|\Omega|} \int_{\Omega} \left(m(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\right)^2 \mathrm{d}\mathbf{x} \tag{6}$$

where m (f) stands for the moving (fixed) image intensity function and $\mathbf{u}(\mathbf{x})$ is the displacement field in $\mathbf{x}$

The derivative in Equation (1) is equal to

$$\left.\frac{\partial MSE(\mathbf{u} + \epsilon \mathbf{h})}{\partial \epsilon}\right|_{\epsilon=0} = \frac{1}{|\Omega|} \int_{\Omega} 2 \cdot \left(m(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\right) \cdot \nabla m(\mathbf{x} + \mathbf{u}(\mathbf{x})) \cdot \mathbf{h}(\mathbf{x}) \cdot \mathrm{d}\mathbf{x} \tag{7}$$

By identifying Equation (7) with Equation (3), the metric flow for the MSE metric appears as

$$\phi_{,u}^{MSE}(\mathbf{x}) = \frac{1}{|\Omega|} 2 \cdot \left(m(\mathbf{x} + \mathbf{u}(\mathbf{x})) - f(\mathbf{x})\right) \cdot \nabla m(\mathbf{x} + \mathbf{u}(\mathbf{x})) \tag{8}$$

This equation reminds from an optical flow equation except that the denominator has been omitted. Cachier [1] has shown that the denominator of Thiron's evolution equation can be obtained by a second order analyze.

2.2 Application to the mutual information metric

The same approach can be applied to the mutual information metric. Mutual information is computed from the joint p^{fm} and marginal (p^f and p^m) histograms of the fixed and moving images using

$$\begin{aligned} MI = & \sum_{i_1,i_2} p^{fm}(i_1,i_2)\log(p^{fm}(i_1,i_2)) \\ & - \sum_{i_2} p^m(i_2)\log(p^m(i_2)) - \sum_{i_1} p^f(i_1)\log(p^f(i_1)) \end{aligned} \tag{9}$$

From (9), the derivative of (1) can be computed as [13] (Eq. 23)

$$\left.\frac{\partial MI_{\mathbf{u}+\epsilon\mathbf{h}}}{\partial\epsilon}\right|_{\epsilon=0} = \sum_{i_1,i_2} \left.\frac{\partial p^{f,m}_{\mathbf{u}+\epsilon\mathbf{h}}(i_1,i_2)}{\partial\epsilon}\right|_{\epsilon=0} \cdot \log\left(\frac{p^{f,m}_{\mathbf{u}+\epsilon\mathbf{h}}(i_1,i_2)}{p^m_{\mathbf{u}+\epsilon\mathbf{h}}(i_2)}\right) \tag{10}$$

Mattes [9] has proposed the following estimator of the joint probability density function (PDF) between the fixed and moving image.

$$p^{f,m}_{\mathbf{u}}(i_1,i_2) = \frac{1}{|\Omega|}\int_\Omega \beta^1(f(\mathbf{x})-i_1)\beta^3(m(\mathbf{x}+\mathbf{u})-i_2)\mathrm{d}\mathbf{x} \tag{11}$$

Using the estimator of Equation (11), the derivative regarding ϵ can be expanded as

$$\left.\frac{\partial MI_{\mathbf{u}+\epsilon\mathbf{h}}}{\partial\epsilon}\right|_{\epsilon=0} = \frac{1}{|\Omega|}\int_\Omega \underbrace{\nabla m(\mathbf{x}+\mathbf{u})\kappa^{f,m}_{\mathbf{u}}(\mathbf{x})}_{\phi^{MI}_{,u}(\mathbf{x})}\mathbf{h}(\mathbf{x})\mathrm{d}\mathbf{x} \tag{12}$$

where $\kappa^{f,m}_{\mathbf{u}}(\mathbf{x})$ is defined by

$$\kappa^{f,m}_{\mathbf{u}}(\mathbf{x}) = \frac{1}{|\Omega|}\sum_{i_1,i_2}\log\left(\frac{p^{f,m}_{\mathbf{u}}(i_1,i_2)}{p^m_{\mathbf{u}}(i_2)}\right)\beta^1(f(\mathbf{x})-i_1)\beta^{3'}(m(\mathbf{x}+\mathbf{u})-i_2) \tag{13}$$

Equations 12 and 13 show that the mutual information flow is the product between the moving image gradient (as in the case of the MSE metric) multiplying a weighted sum over all bins of the joint PDF variation generated by the $\epsilon\mathbf{h}$ variation in the displacement field.

3 Projecting the metric flow for prior transformation models with the transformation jacobian

In this section, a connection is established between the metric flow and the derivative for a given transformation model which uses the `GetJacobian()` of the `itk::Transform` base class.

In the following, we assume that the transformation $T(x,\{p_k\})$ can be developed at a coordinate x using a first order development

$$\Delta\mathbf{T}(\mathbf{x}) \cong \sum_k^K \underbrace{\frac{\partial\mathbf{T}(\mathbf{x})}{\partial p_k}}_{\text{Transformation jacobian}_k \overset{\text{def}}{=} J_k(x)} \Delta p_k \tag{14}$$

where the $\{p_k\}$ designate the transformation parameters. Equation (14) brings also forward the concept of transformation jacobian as it is described in the ITK Software Guide [6] (Section 8.8.2).

The total displacement at a given iteration of the optimization procedure is the sum of the current displacement field and the variation of the transformed coordinate

$$\mathbf{u}_{\text{tot}}(\mathbf{x}) = (\mathbf{T}(\mathbf{x}) + \Delta\mathbf{T}(\mathbf{x})) - \mathbf{x} = \mathbf{u}(\mathbf{x}) + \Delta\mathbf{T}(\mathbf{x}) \tag{15}$$

Instead of looking for a dense deformation field, classical numerical optimization schemes look at each iteration for the gradient of the metric 2 regarding the transformation parameters. Using the approximation of Equation (14), this gradient can be computed by

$$\begin{aligned} \frac{\partial \mathcal{I}(\mathbf{u}(\mathbf{x}) + \Delta\mathbf{T}(\mathbf{x}))}{\partial p_k} &= \frac{1}{|\Omega|} \int_\Omega \phi_{,\mathbf{u}}(\mathbf{x}) \cdot \frac{\partial \mathbf{u}(\mathbf{x})}{\partial p_k} \mathrm{d}\mathbf{x} \\ &= \frac{1}{|\Omega|} \int_\Omega \phi_{,\mathbf{u}}(\mathbf{x}) \cdot \mathbf{J_k}(\mathbf{x}) \mathrm{d}\mathbf{x} \end{aligned} \tag{16}$$

Equation (16) show that the derivative of the metric $\mathcal{I}$ is simply equal to the scalar product between the metric flow and the transformation jacobian. This result confirms the intuition that both approaches should be linked since they both result from a first order analysis of the metric variations. In the metric flow case, the variation of the metric is prompted by a continuous perturbation function h. In the parametric case, the variation is prompted by discrete perturbation of the vector of transformation parameters.

For speeding up the computation of Equation (16), the continuous integral can be approximated by random samples scattered uniformly over the fixed image domain

$$\frac{\partial \mathcal{I}(\mathbf{u}(\mathbf{x}) + \Delta\mathbf{T}(\mathbf{x}))}{\partial p_k} = \frac{1}{N_s} \sum_i \phi_{,\mathbf{u}}(\mathbf{x_i}) \cdot \mathbf{J_k}(\mathbf{x_i}) \tag{17}$$

3.1 Example : translation transformation

In the case of a translation transform, the jacobian of the transformation is the identity matrix.

$$\frac{dy(x)}{dt} = (dy(x)/dt_x, dy(x)/dt_y, dy(x)/dt_x) = I_3 \tag{18}$$

Injecting this expression of the transformation jacobian in Equation (16) leads to

$$\frac{\partial \mathcal{I}(\mathbf{u}(\mathbf{x}) + \Delta\mathbf{T}(\mathbf{x}))}{\partial t_k} = \frac{1}{|\Omega|} \int_\Omega \phi_{,\mathbf{u}}(\mathbf{x}) \cdot \hat{\mathbf{e}}_\mathbf{k} \mathrm{d}\mathbf{x} \tag{19}$$

where t_k stands for the translation parameter in xyz and $\hat{e}_k(x)$ designates the k^{th} basis vector. This shows that in the case of a translation transform, the metric gradient as implemented currently in ITK can be seen as an average of the flow vectors (8) over the entire image domain.

3.2 Local transformations

A simple distinction can be made among the large number of transformation models : one the one hand, *global transformations* can be defined as transformations whose each parameter acts on the entire image

domain, on the other hand *local transformations* are transformations whose each parameter acts only on a region of the fixed image domain.

On an implementation point of view, the main difference for computing the metric derivative as described in Section 3 is the sparse appearance of the transformation jacobian. Indeed, in the case of a local transformation, only a few subset of parameters acts on a given coordinate of the fixed image domain. Therefore, including in ITK a support for dealing with a sparse representation of the transformation jacobian could provide a speed-up in non-rigid registration where this derivative is explicitly computed.

BSpline transformation

Using the notation of Mattes [9], a BSpline transformation is characterized by the following expression of the displacement field

$$\mathbf{u}(\mathbf{x}) = \sum_k \mathbf{p_k} \cdot \beta^{(3)} \left(\frac{\mathbf{x} - \lambda_j}{\Delta\rho} \right) \tag{20}$$

where the subset of k indices covers the 64 control points nearest to the x coordinate and β is the product of three separate kernels in xyz.

The 64 non-zero columns of the transformation jacobian at a given point $\mathbf{x}$ have the following aspect [9]

$$\frac{\partial \mathbf{u}(\mathbf{x})}{\partial p_{k,x}} = \left[\beta^{(3)} \left(\frac{\mathbf{x} - \lambda_j}{\Delta\rho} \right), 0, 0 \right]^T \tag{21}$$

for the x component of $\mathbf{p_k}$ (the y and z components follow similarly).

In our implementation, instead of returning a full transformation jacobian, a method called `GetSparseJacobian()` returns each non-null column of the jacobian among with the column indice in the full matrix. The advantage of this representation is to speed up the computation of the metric derivative when projecting the metric flow on the vector of transformation parameters.

Volumetric mesh transformation

Volumetric meshes are an alternative way of generating a local transformation model.In this approach, the domain of interest is divided into a mesh of elements.Numerous techniques have been proposed for designing elements which follow accurately the borders of the different structures. These techniques generally allow to find a compromise between two conflicting requirements : faithful surfaces representation and well shaped elements (see for instance [10, 4, 3, 2]).

Inside each element of the volumetric mesh, the displacement field u is estimated by

$$u_l(x) = \sum_{n \in \text{Nodes}} u_l^n N_{el}^n(x) \tag{22}$$

where u_l^n is the l^{th} component of the displacement for the n^{th} node. N_{el}^n represents the shape function associated with the n^{th} node. Shape functions are non-null in the element volume only. The shape functions of a node tends to zero around the other node of the same element. In the case of tetrahedral meshes, the shape functions are often chosen as linear functions.

4 Helper classes and programming design

The code attached to this submission attempts to integrate the concepts described in this paper into the current ITK design. The following of this section describes the main features of our implementation.

4.1 Volumetric mesher and Elastic transformation

The `itkElasticTransform` class implements a generic elastic transformation using the existing FEM design of the toolkit. The transformation is parametrized by the displacements at the nodes of a volumetric mesh. The mesh is generated by another object deriving of an abstract class called `itkMesher`. As an example two mesh generation schemes are proposed in this paper. A first one (`itkMesherTetrahedrons`) generates a regular grid of tetrahedrons by splitting a lattice of cubes in a five tetrahedrons pattern[4] . The second (`itkMesherTetrahedronsBCC`) starts from two interleaved regular grids and connects them with tetrahedrons in a red-green pattern [10].

The mesher object is templated over the image dimension and the type of element used in the volumetric mesh (tetrahedral elements in the examples in this paper). The elastic transformation is also templated over the element type (in addition to the coordinate type and the input/output space dimensions like any other transformation). In the ITK-FEM design, each element is associated with a material property (e.g. linear elasticity). Such a material property intervenes for including a regularization strategy in the registration pipeline.

This can be for instance achieved by adding the deformation energy to the cost function passed to the optimizer. For this purpose, the elastic transformations returns the deformation energy using the `GetDeformationEnergy` method. Another possibility is to use a gradient descent approach adding at each iteration the energy derivative by use of the `GetDerivativeDeformationEnergy` method.

4.2 Hermosillo mutual information flow

The `itkHermosilloMutualInformationFilter` and `itkHermosilloMutual-InformationFunction` objects implements the mutual information flow concepts into the ITK PDE deformable registration design.

At each iteration, the joint probability densities are estimated like in the Mattes implementation by sampling the fixed image domain and updating the probability densities for the current displacement field.

When the `ComputeUpdate` method is called at each coordinate of the dense deformation field, the variation of the joint regarding the displacement at this point is computed by deriving Equation (11). The metric flow is then computed from the PDF derivative using Equation (10). As explained in the next subsection, our implementation uses an helper class for sharing functionalities with the existing Mattes mutual information class.

4.3 Mattes mutual information class and helper

Since the computation of this flow is really similar to the computation of the derivative in the itkMattes-MutualInformationImageToImageMetric, we moved some of the code into an helper class which is used in common by the Mattes mutual information metric class and the Hermosillo mutual information flow class.

The concept of local jacobian transformation has been added in the new Mattes mutual information class for speeding up the computation of the metric derivative. This information is used in the `ComputePDFDerivatives` class : only the slices in the joint PDF derivative corresponding to non-null columns in the jacobian are updated. However, the resulting improvement in computation time does not yet match the performances reached by a B-Spline transformation, since for this transformation, all jacobian values are computed only once and stored in memory. This assumes that the jacobian will always return the same value at a given coordinate for any set of transformation parameters. Even if it is true for B-Spline transformation and transformation based on volumetric meshes, this property can not be generalized to any local jacobian transformation.

In the ideal case, the helper class should implement a `GetFlow(PointType&)` which would be used by the mutual information flow filter and the mutual information metric when computing the derivative using Equation (17). However, in the case of the mutual information metric, this would require two iterations over the random fixed image samples : a first one for updating the probability densities and a second one for computing the metric derivative. If this is not the problem for the case of the mutual information flow (the update of probability densities approximated using random samples if far less time consuming that visiting all the displacement field voxels), it would decrease the performances of the `GetValueAndDerivative` method in the mutual information metric class. For this reason, only the random sampling of the fixed image domain, the computation of the moving image derivative at a mapped coordinate and the mapping of a moving or fixed image intensity to an histogram bin index have been moved to the Helper class.

5 Experiments

This section describes some experiments comparing three algorithms

- The non-rigid B-Spline deformation model with LBFGSB optimizer and Mattes mutual information metric (the code is taken from the file `DeformableRegistration8.cxx` in the `Examples/Registration/` ITK directory
- The non-rigid FEM deformation model with LBFGSB optimizer and Mattes mutual information
- The mutual information flow algorithm.

The parameters for each of these algorithms are described in the `README.txt` file inside the `Build/` directory of this submission.

5.1 Sphere to ellipsoid matching

For synthetic experiments, a matching of a sphere to an ellipsoid is proposed. The background-foreground intensities in the ellipsoid are switched compared to the sphere for simulating a multi-modal registration experiment. Figure 1 shows the two images to matched in the first row. The sphere is taken as moving image and must be deformed to an ellipsoid. The second row of Figure 1 plots the sphere deformed by the three transformation model (B-Spline, volumetric mesh and mutual information flow). For the B-Spline deformation, we use a grid of 5 on the fixed image region (which means a total grid of 8). For the volumetric mesh, we used the BCC regular tetrahedral mesher with a resolution of 10 pixels in each dimension for the two interleaved grids. The three versions of the deformed sphere are stored in the `Build/results` directory of this submission. Three tests have been added to the `CMakeLists.txt` for ensuring that the three algorithms always produce identical results on this testing data-set.

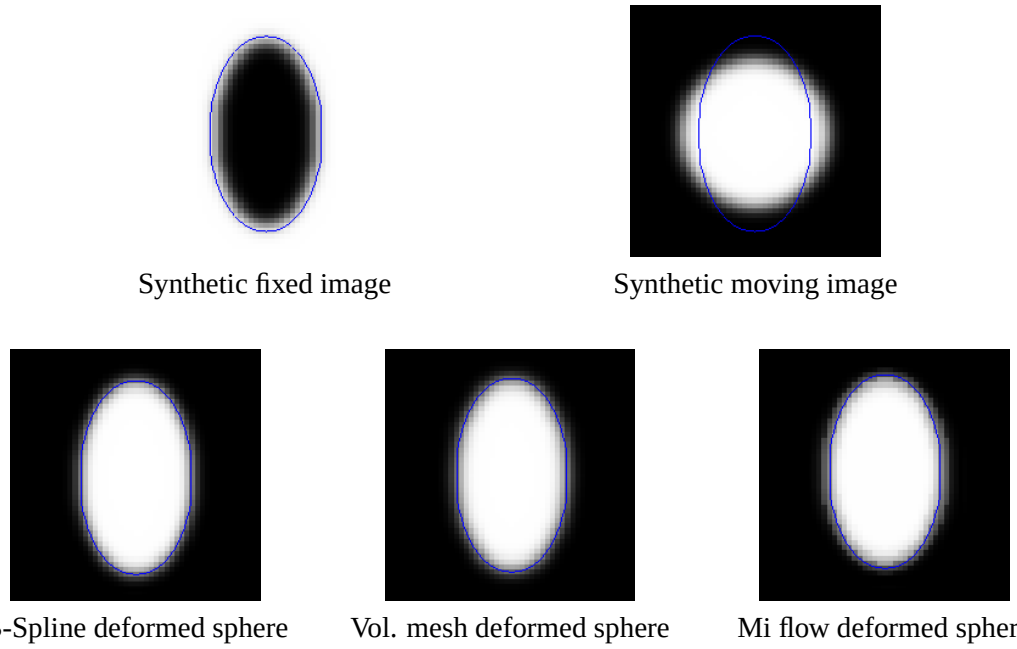

Figure 1: Sphere to ellipsoid matching for testing the three different deformation models (B-Spline, volumetric model and dense deformation flow). The upper row shows the images to be matched, the bottom row show the deformed sphere using each algorithm with the target contour in blue.

The code for running this experiment can be found in the files `registrationFEM.cxx`, `registrationBS.cxx` and `registrationMIFLOW.cxx` in the `Source` directory.

Figure 2 shows the three dimensional displacement field produced by the mutual information flow algorithm. An important observation from practical experiments is the importance of the number of bins chosen in the joint PDF estimation. A closer look to Equation (10) shows potential numerical issues due to the $\frac{p^{f,m}}{p^m}$ ratio, amplified by the $\log(\cdot)$ function. Empty bins where $p^{f,m}$ tends to zero can require the multiplication between small and large numbers. This problem is not specifically related to mutual information flow but also appears for computing the derivative mutual information regarding transformation parameters. However in this case, the effect is attenuated by the averaging of all flow vectors on the support of each basis function.

Since the optimizer for the B-Spline and the volumetric mesh is the same (LBFGSB optimizer), we can plot the value of the cost function at each iteration to compare the convergence of these two algorithms as shown in Figure 3. Even if the number of parameters is more or less equivalent for both transformation models (1536 for B-Spline and 1473 for the volumetric mesh), the convergence is faster in this case with a B-Spline deformation model.

5.2 Lung CT images matching

We also ran the three algorithms on a data-set of CT lung images (512 voxels). [1] For the B-Spline deformation, we use a grid of 8 on the fixed image region (which means a total grid of 8). For the volumetric mesh, we used the BCC regular tetrahedral mesher with a resolution of 90 pixels two interleaved grids. All com-

[1] These images can be donwloaded at this URL : http://euterpe.tele.ucl.ac.be/Waleo2/insight

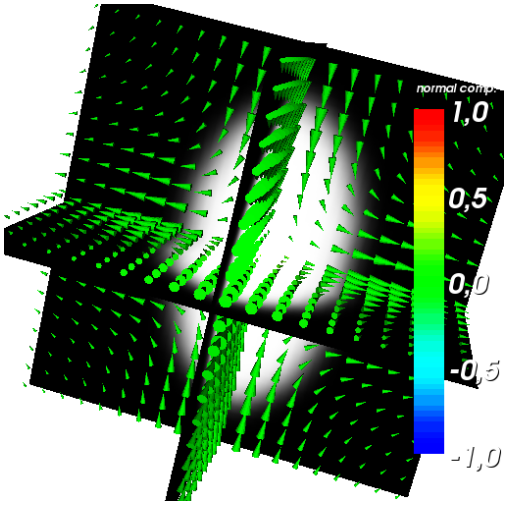

Figure 2: Dense deformation field obtained by mutual information flow matching for a time step of 500 and 250 iterations. The color codes the relative magnitude of the component normal to the slicing plane.

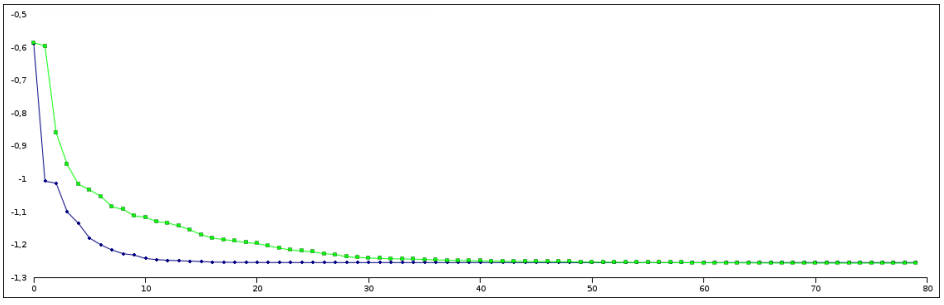

Figure 3: Mutual information plotted at each iteration of the LBFGSB optimizer for a B-Spline deformation model (lower curve, ♦) and a volumetric mesh deformation model (upper curve, ■). The final cost function value is the same with both deformation models but the B-Spline deformation model converges faster.

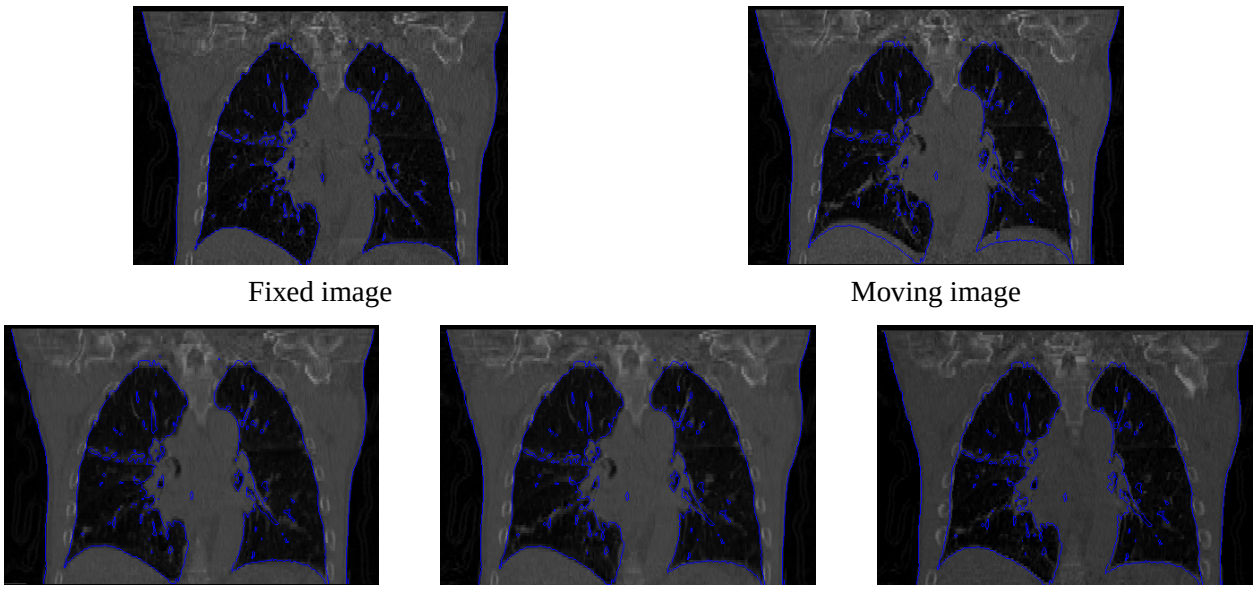

Figure 4: Mutual information based matching of two CT respiratory phases using three deformation models (B-Spline, volumetric model and dense deformation flow). The upper row shows one coronal slice of the fixed and moving images. The bottom row shows the deformed moving image using each algorithms. Lung contours in the fixed image are overlaid over the original and deformed versions of the moving image. No significant difference is observed in coronal views between the three algorithms.

mand lines for generating the results are stored in the `registration-XX.sh` (where XX designates the algorithm) scripts in the `Build` directory. All other parameters are contained in the `cxx` files and identical to those used for the previous experiment.

Figures 4 and 5 show for one sagittal and one coronal slice the results of the three algorithms. The lung contours in the fixed image are overlaid over the original and deformed moving images. The transformation to recover is mainly a vertical shift in the bottom slices of the image. The three algorithm are able to provide a good estimation of the deformation. Some differences appear in areas where the deformation is non-linear (B-Splines perform slightly better than a tetrahedral elements of a BCC mesh with linear shape functions). A dense deformation is the most flexible representation of the deformation and mutual information flow therefore gives a better image-contours match. However, mutual information flow is more sensitive to local artifacts in the image (like at the bottom right of the slice shown in Figure 5).

6 Conclusion

This paper has introduced two possible algorithmic contributions for the ITK toolkit.

The first one is to introduce in ITK a new deformation model based on a volumetric mesh with shape functions like in a finite element approach. The mesh is used as a transformation model and can be plugged with any similarity metric or optimization method available in the toolkit. Elastic transformation models raise the possibility to associate mechanical properties with each element of the mesh. Such property is potentially very useful in the design of regularization strategy : different materials could be used for meshing

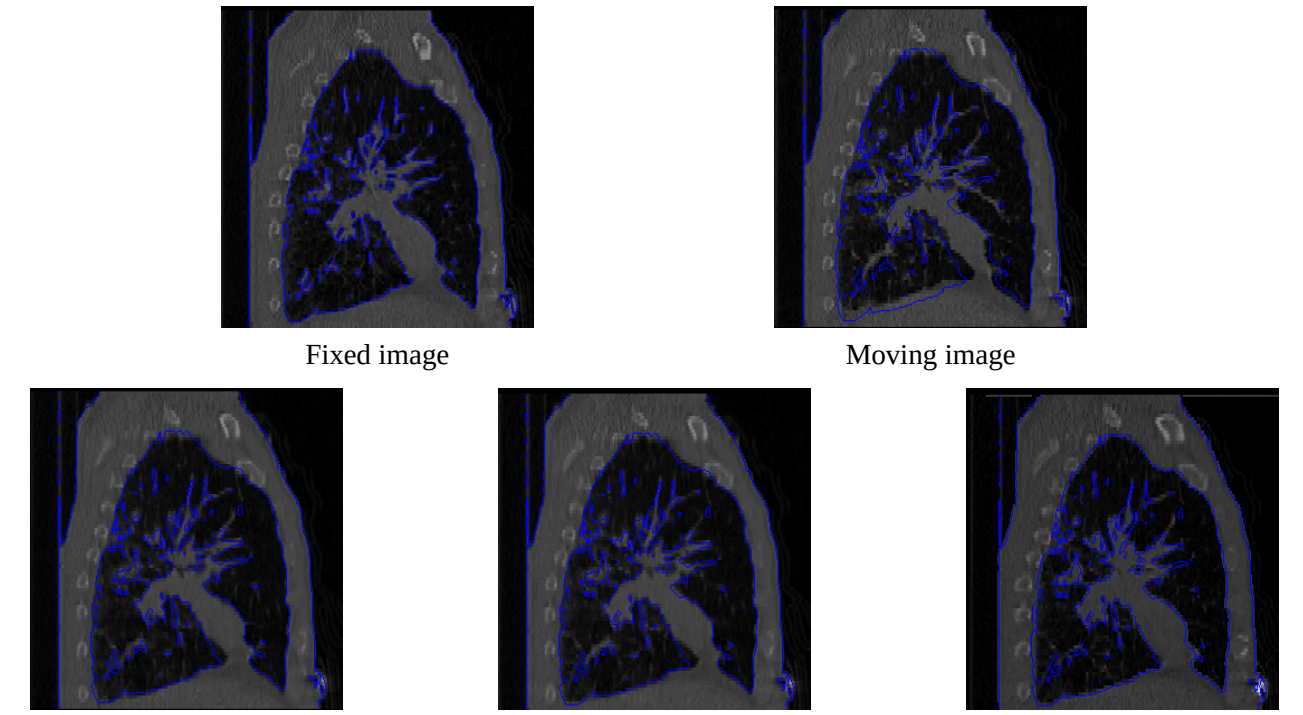

Figure 5: Mutual information based matching of two CT respiratory phases using three deformation models (B-Spline, volumetric model and dense deformation flow). The upper row shows one sagittal slice of the fixed and moving images. The bottom row shows the deformed moving image using each algorithms. Lung contours in the fixed image are overlaid over the original and deformed versions of the moving image. In regions where the deformation is non-linear, B-Splines perform slightly better than a BCC mesh with linear shape functions. Mutual information flow has the best flexibility in its transformation representation and can render pretty well non-linear deformations. However, mutual information flow is more sensitive to local artefact's in the image (see the bottom right of this slice for instance).

the fixed image region and allow this way an heterogeneous regularization.

The other contribution of this paper is the implementation of a dense deformation field estimation scheme optimizing mutual information. Since a significant part of this implementation is inspired from the Mattes implementation of mutual information, we propose to use an helper class for avoiding undesired code duplications.

References

[1] P. Cachier, X. Pennec, and N. Ayache. Fast non-rigid matching by gradient descent : study and improvements of the "demons" algorithm. Technical Report 3706, INRIA, june 1999. 2.1

[2] A. du Bois d'Aische, M. De Craene, S. Haker, N. Weisenfeld, C. Tempany, B. Macq, and S. K. Warfield. Improved non-rigid registration of prostate mri. In *7th Medical Image Computing and Computer-Assisted Intervention*, volume 3216 of *Lecture Notes in Computer Science*, pages 845–852. Springer, 2004. Saint-Malo, France. 3.2

[3] Matthieu Ferrant, Arya Nabavi, Benoit Macq, Peter McL. Black, Ferenc A. Jolesz, Ron Kikinis, and Simon K. Warfield. Serial Registration of Intraoperative MR Images of the Brain. *Med Image Anal*, 6(4):337–359, 2002. 3.2

[4] Matthieu Ferrant, Arya Nabavi, Benoit Macq, Ferenc A. Jolesz, Ron Kikinis, and Simon K. Warfield. Registration of 3D Intraoperative MR Images of the Brain Using a Finite Element Biomechanical Model. *IEEE Trans Med Imag*, 20:1384–1397, Dec 2001. 3.2, 4.1

[5] Gerardo Hermosillo, Christophe Chefd'Hotel, and Olivier Faugeras. A variational approach to multimodal image matching. Technical report, INRIA, Feb 2001. 1, 2, 2

[6] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The ITK Software Guide*. Kitware, Inc. ISBN 1-930934-15-7, http://www.itk.org/ItkSoftwareGuide.pdf, second edition, 2005. 2, 3

[7] F. Maes and A. Collignon. Multimodality image registration by maximization of mutual information. *IEEE Transactions on Medical Imaging*, 16, 1997. 1

[8] J. B. A. Maintz and M. A. Viergever. A survey of medical image registration. *Medical Image Analysis*, 2(1):1–36, 1998. 1

[9] D. Mattes, D.R. Haynor, H. Vesselle, T.K. Lewellen, and W. Eubank. Pet-ct image registration in the chest using free-form deformations. *IEEE Transaction on Medical Imaging*, 22(1):120–128, January 2003. 2.2, 3.2, 3.2

[10] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *12th International Meshing Roundtable*, pages 103–114, Sandia National Laboratories, september 2003. 3.2, 4.1

[11] J. C. Spall. Overview of the simultaneous perturbation method for efficient optimization. `http://techdigest.jhuapl.edu/td/td1904/spall.pdf`, *Hopkins APL Technical Digest*, 19:482–492, 1998. 1

[12] M. Styner, G. Gerig, C. Brechbuehler, and G. Szekely. Parametric estimate of inten- sity inhomogeneities applied to mri. *IEEE Transactions On Medical Imaging*, 19:153–165, 2000. 1

[13] P. Thevenaz and M.; Unser. Optimization of mutual information for multiresolution image registration. *Image Processing, IEEE Transactions on*, 9(12):2083 – 2099, Dec 2000. 2.2

[14] J.-P. Thirion. Image matching as a diffusion process: An analogy with maxwell's demons. *Med. Image Anal.*, 2(3):243–260, 1998. 1