An ITK Implementation of the Symmetric Log-Domain Diffeomorphic Demons Algorithm

Florence Dru, Tom Vercauteren

An ITK Implementation of the Symmetric Log-Domain Diffeomorphic Demons Algorithm

Release 0.0.2

Florence Dru[1] and Tom Vercauteren[2]

May 25, 2009

[1]Asclepios Research Group, INRIA Sophia Antipolis
[2]Mauna Kea Technologies, Paris

Abstract

This article provides an implementation of the symmetric log-domain diffeomorphic image registration algorithm, or symmetric demons algorithm for short. It generalizes Thirion's demons and the diffeomorphic demons algorithm. The main practical advantages of the symmetric demons with respect to the other demons variants is that is provides the inverse of the spatial transformation at no additional computational cost and ensures that the registration of image A to image B provides the inverse of the registration from image B to image A. The algorithm works completely in the log-domain, i.e. it uses a stationary velocity field to encode the spatial transformation as its exponential. Within the Insight Toolkit (ITK), the classical demons algorithm is implemented as part of the finite difference solver framework. Our code reuses and extends this generic framework. The source code is composed of a set of reusable ITK filters and classes together with their unit tests. We also provide a small example program that allows the user to compare the different variants of the demons algorithm. This paper gives an overview of the algorithm, an overview of its implementation and a small user guide to ease the use of the registration executable.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3060]

Contents

Forewords

This article is a companion paper to the MICCAI 2008 paper [19] entitled "Symmetric Log-Domain Diffeomorphic Registration: A Demons-based Approach". It is intended to share the source code of the algorithm. As such it provides only basic information about the theory and does not present an evaluation of the method. The reader is thus invited to refer to [19] for the theoretical aspects and for an evaluation of the algorithm.

1 Introduction

Since Thirion's seminal paper [14], the demons algorithm has become a popular method for the problem of intra-modality deformable image registration. The demons algorithm has successfully been used by several teams [22, 21]. An open source implementation of it is available in the Insight Toolkit [8]. The success of this method in the field of biomedical imaging can largely be explained by its efficiency. Thirion introduced *demons* that push according to local characteristics of the images in a similar way Maxwell did for solving the Gibbs paradox. The forces are inspired from the optical flow equations [2] and the method alternates between computation of the forces and their regularization by a simple Gaussian smoothing.

With the advent of computational anatomy and in the absence of a justified physical model of inter-subject variability, statistics on diffeomorphisms have become an important topic [1, 4, 15]. Diffeomorphic registration algorithms are at the core of this research field since they often provide the *input data*. They usually rely on the computationally heavy solution of a partial differential equation [7, 13, 9, 12, 3] or use very small optimization steps [6]. In [18, 20], the authors proposed an alternative approach for diffeomorphic registration that builds on Thirion's demons algorithm to provide a computationnaly efficient diffeomorphic image registration algorithm. The diffeomorphic demons source code was released in the Insight Journal [17] and is now included in ITK. Although the diffeomorphic demons algorithm already has some interesting properties for computational anatomy, it does not allow for easily accessing the inverse of the spatial transformation and is not symmetric with respect to the order of the images that needs to be registered register. In [19], the authors proposed to build on the diffeomorphic demons to provide these two missing properties. They propose an image registration scheme that uses a demons-like alternate optimization approach for efficiency but represents the complete deformation as an exponential of a smooth velocity field. This approach will hereafter be referred to as a log-domain one.

This paper aims at releasing an open-source implementation of the symmetric log-domain demons image registration algorithm, or symmetric demons for short.

2 Overview of the Algorithm

2.1 The Demons Algorithm

It has been shown in [5] that the demons algorithm can be seen as an optimization of a global energy. The main idea is to introduce a hidden variable in the registration process: correspondences. We then consider the regularization criterion as a prior on the smoothness of the transformation s. Instead of requiring that point correspondences between image pixels (a vector field c) be exact realizations of the transformation, one allows some error at each image point.

Given a *fixed image* $F(.)$ and a *moving image* $M(.)$, we end-up with the global energy:

$$E(c,s) = \frac{1}{\sigma_i^2}\,\mathrm{Sim}\,(F,M\circ c) + \frac{1}{\sigma_x^2}\,\mathrm{dist}\,(s,c)^2 + \frac{1}{\sigma_T^2}\,\mathrm{Reg}\,(s)\,, \tag{1}$$

$$\mathrm{Sim}\,(F,M\circ s) = \frac{1}{2}\,\|F - M\circ s\|^2 = \frac{1}{2|\Omega_P|}\sum_{p\in\Omega_P}|F(p)-M(s(p))|^2\,, \tag{2}$$

where Ω_P is the region of overlap between F and $M\circ s$, σ_i accounts for the noise on the image intensity, σ_x accounts for a spatial uncertainty on the correspondences and σ_T controls the amount of regularization we need. We classically have $\mathrm{dist}\,(s,c) = \|c-s\|$ and $\mathrm{Reg}\,(s) = \|\nabla s\|^2$ but the regularization can also be modified to handle fluid-like constraints [5].

Within this framework, the demons registration can be explained as an alternate optimization over s and c. It can conveniently be summarized into the algorithm below:

Algorithm 1 (Demons Algorithm)**.**

- Choose a starting spatial transformation (a vector field) s
- Iterate until convergence:
  - Given s, compute the demons forces update field u by minimizing $E_s^{\mathrm{corr}}(u) = \|F - M\circ(s+u)\|^2 + \frac{\sigma_i^2}{\sigma_x^2}\|u\|^2$ with respect to u
  - If a fluid-like regularization is used, let $u \leftarrow K_{\mathrm{fluid}} \star u$ where K_{fluid} is a Gaussian convolution kernel
  - Let $s \leftarrow s+u$
  - If a diffusion-like regularization is used, let $s \leftarrow K_{\mathrm{diff}} \star s$ where K_{diff} is a Gaussian convolution kernel

In [16], it was shown that a Newton method on $E_s^{\mathrm{corr}}(u)$ provided the folowing optimization step:

$$u(p) = -\frac{F(p) - M\circ s(p)}{\|J^p\|^2 + \frac{\sigma_i^2(p)}{\sigma_x^2}}J^{pT} \tag{3}$$

where a local estimation $\sigma_i(p) = |F(p) - M\circ c(p)|$ of the image noise is used and where $J^p = -\nabla_p^T(M\circ s)$ with a Gauss-Newton method, $J^p = -\frac{1}{2}\left(\nabla_p^T F + \nabla_p^T(M\circ s)\right)$ with the efficient second-order minimization (ESM) method of [11] and $J^p = -\nabla_p^T F$ with Thirion's rule. Note that σ_x then controls the maximum step length: $\|u(p)\| \leq \sigma_x/2$.

2.2 Newton Methods for Lie Groups

The most straightforward way to adapt the demons algorithm to make it diffeomorphic is to optimize (1) over a space of diffeomorphisms. This can be done as in [11, 10] by using an intrinsic update step

$$s \leftarrow s \circ \exp(u), \tag{4}$$

on the Lie group of diffeomorphisms. This approach obviously requires an algorithm to compute the exponential for the Lie group of interest. Thanks to the scaling and squaring approach of [1], this exponential can efficiently be computed for diffeomorphisms with just a few compositions:

Algorithm 2 (Fast Computation of Vector Field Exponentials)**.**

- Choose N such that $2^{-N}u$ is close enough to 0, e.g. $\max \left\| 2^{-N}u(p) \right\| \leq 0.5$
- Perform an explicit first order integration: $v(p) \leftarrow 2^{-N}u(p)$ for all pixels
- Do N (not 2^N!) recursive squarings of v: $v \leftarrow v \circ v$

2.3 Diffeomorphic Demons

By plugging the above Newton method tools for Lie groups within the alternate optimization framework of the demons, we get the following non-parametric diffeomorphic image registration algorithm [18]:

Algorithm 3 (Diffeomorphic Demons Iteration)**.**

- Compute the demons forces update field u using (3)
- If a fluid-like regularization is used, let $u \leftarrow K_{\text{fluid}} \star u$.
- Let $c \leftarrow s \circ \exp(u)$, where $\exp(u)$ is computed using Algorithm 2
- If a diffusion-like regularization is used, let $s \leftarrow K_{\text{diff}} \star s$

2.4 Log-domain Diffeomorphic Demons

The first problem with the diffeomorphic demons in algorithm 3 is that it does not easily give access to the inverse of the spatial transformation. Even though the exponential mapping in algorithm 2 readily support the computation of the inverse through $\exp(-u)$, the diffeomorphic demons only uses it to encode the adjustment made at each iteration to the current transformation.

The log-domain demons algorithm [19] solves this by encoding the complete spatial transformation through the exponential $s = \exp(v)$. The key is then to use the Baker-Campbell-Hausdorff (BCH) formula [4] to compute the update transformation $s \circ \exp(u) = \exp(v) \circ \exp(u)$ as an exponential $\exp(w)$. With some notational abuses, this boils down to:

$$\log(\exp(v) \circ \exp(u)) \approx v + u + \frac{1}{2}[v,u] + \frac{1}{12}[v,[v,u]] + \ldots, \tag{5}$$

where $[v,u]$ is the the Lie bracket:

$$[v,u](p) = \mathrm{Jac}(v)(p).u(p) - \mathrm{Jac}(u)(p).v(p). \tag{6}$$

Then in order to be consistent with the log-domain representation but keep the simplicity of the demons algorithm, the Gaussian smoothing is performed directly in the log-domain. This can be summarized as follows:

Algorithm 4 (Log-domain Diffeomorphic Demons Iteration)**.**

- Compute the demons forces update field u using (3)
- If a fluid-like regularization is used, let $u \leftarrow K_{\text{fluid}} \star u$.
- Let $v \leftarrow \log(\exp(v) \circ \exp(u))$, where $\log(\exp(v) \circ \exp(u))$ is computed using the BCH approximation (5) and $\exp(u)$ is computed using Algorithm 2
- If a diffusion-like regularization is used, let $v \leftarrow K_{\text{diff}} \star v$
- Compute $s = \exp(v)$ and, if necessary, $s^{-1} = \exp(-v)$ using Algorithm 2

2.5 Symmetric Log-domain Diffeomorphic Demons

The most obvious way of getting a symmetric registration framework from the non-symmetric one is by symmetrizing the global energy:

$$s_{opt} = \arg\min_s \left(E(I_0, I_1, s) + E(I_1, I_0, s^{-1}) \right), \tag{7}$$

To efficiently solve for this symmetrized cost function, the symmetric log-domain demons algorithm, or symmetric demons for short, computes independently a forward, u^{forw}, and and a backward, u^{back}, update and performs a symmetrized update. The symmetric demons simply average, in the log-domain, the forward and backward iterations:

$$v \leftarrow \frac{1}{2} K_{\text{diff}} \star \left(\log\left(\exp(v) \circ \exp(K_{\text{fluid}} \star u^{\text{forw}}) \right) - \log\left(\exp(-v) \circ \exp(K_{\text{fluid}} \star u^{\text{back}}) \right) \right). \tag{8}$$

By keeping only the two first terms of the BCH approximation, the algorithm can be summarized as follows:

Algorithm 5 (Simple Symmetric Demons Iteration)**.**

- Compute the forward demons forces u^{forw} using (3) where I_0 plays the role of the fixed image F and I_1 plays the role of the moving image M
- Compute the backward demons forces u^{back} using (3) where I_1 plays the role of the fixed image F and I_0 plays the role of the moving image M
- For fluid-like regularization let $u \leftarrow \frac{1}{2} K_{\text{fluid}} \star (u^{\text{forw}} - u^{\text{back}})$ else let $u \leftarrow \frac{1}{2}(u^{\text{forw}} - u^{\text{back}})$
- For diffusion-like regularization let $v \leftarrow K_{\text{diff}} \star (v + u)$ else let $v \leftarrow v + u$

For clarity a schematic representation of this algorithm is also presented in Fig. 1.

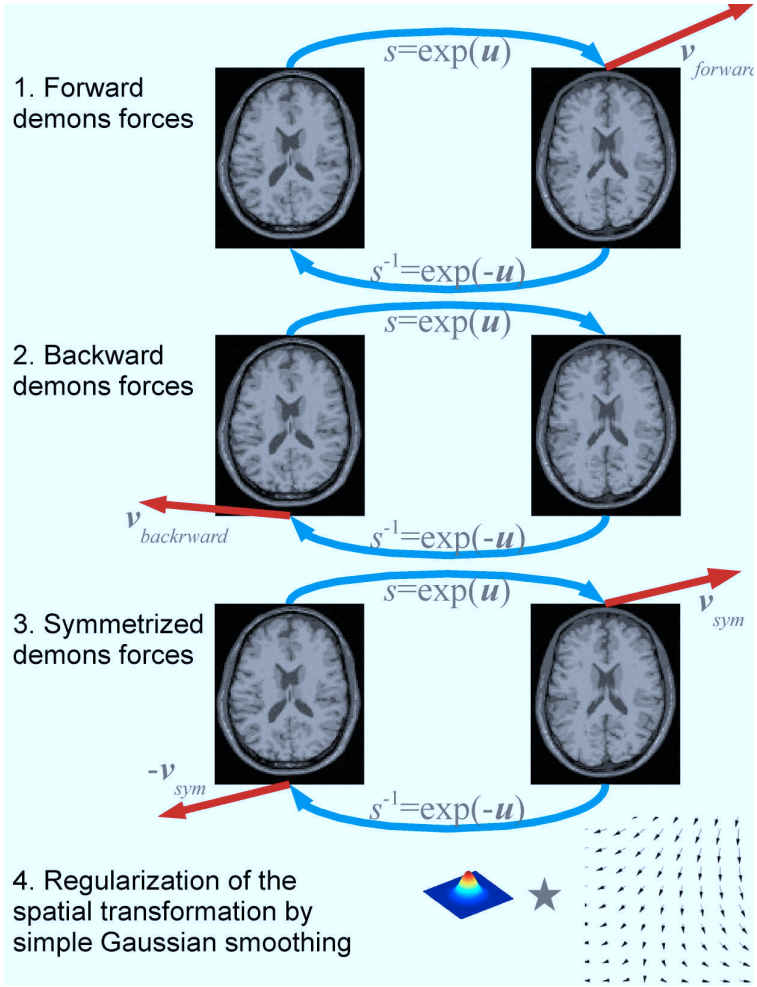

Figure 1: A schematic overview of the symmetric demons algorithm.

3 Brief Note on the Implementation

Our implementation tries to follow the style and design of the Insight Toolkit. All our filters are N-dimensional and are templated over the important types such as the pixel types. In most cases we tried to divide the algorithm into meaningful and reusable classes.

Our algorithm works completely in the log-domain. We therefore implemented a filter that enables the registration of two images in the log-domain, where any transformation is represented as the exponential of a velocity field: the `LogDomainDeformableRegistrationFilter` class. This class is then used as the base class for `SymmetricLogDomainDemonsRegistrationFilter` which is a specialisation version of the `LogDomainDeformableRegistrationFilter` that enables the symmetry and the use of the diffeomorphic demons for the computation of the update step.

As shown in Algorithm 4 in order to cast the update step used in the diffeomorphic demons into a log-domain update we implemented the `VelocityFieldBCHCompositionFilter` that compute Baker-Campbell-Hausdorff formula on two vector fields. This implied the implementation of a filter that computes the Lie bracket of two vector fields: the `VelocityFieldLieBracketFilter`.

In addition to these main classes, our submission also includes a log-domain version of the diffeomorphic demons, which is a specialisation of the `LogDomainDeformableRegistrationFilter` class and is a similar to `SymmetricLogDomainDemonsRegistrationFilter` but without the symmetric aspect of the registration framework.

The algorithms can also use an initial velocity field obtained from a prior registration. Typically a user may want to perform an affine registration before moving to non-rigid registration. In order to ease the creation of a velocity field given a spatial transformation (an itk::Transform), we also have implemented a `TransformToVelocityFieldSource` class. Currently `TransformToVelocityFieldSource` supports up to affine transformations only.

Below is the list of classes, with brief descriptions, that we provide and use within our method:

- **itk::SymmetricLogDomainDemonsRegistrationFilter<TFixedImage, TMovingImage, TField>:** Deformably register two images in the log-domain using a diffeomorphic demons algorithm with a symmetrized optimization scheme
- **itk::LogDomainDemonsRegistrationFilter<TFixedImage, TMovingImage, TField>:** Deformably register two images in the log-domain using a diffeomorphic demons algorithm
- **itk::LogDomainDeformableRegistrationFilter<TFixedImage, TMovingImage, TField>:** Deformably register two images using a PDE-like algorithm where the transformation is represented as the exponential of a velocity field
- **itk::DisplacementFieldCompositionFilter<TInputImage, TOutputImage>:** Given two spatial transformations s_l and s_r represented by the displacement fields ds_l and ds_r, this filter computes the displacement field ds that represents the spatial transformation $s = s_l \circ s_r$
- **itk::VelocityFieldLieBracketFilter<TInputImage, TOutputImage>:** Compute the Lie bracket of two vector fields using the formula $[v,u](p) = \mathrm{Jac}(v)(p).u(p) - \mathrm{Jac}(u)(p).v(p)$. See [19]
- **itk::VelocityFieldBCHCompositionFilter<TInputImage, TOutputImage>:** Compute Baker-Campbell-Hausdorff formula on two vector fields. See [4] and [19]
- **itk::MultiResolutionLogDomainDeformableRegistration<TFixexImage, TMovingImage, TField, TRealType>:** Framework for performing multi-resolution log-domain deformable registration
- **itk::TransformToVelocityFieldSource<TOutputImage, TTransformPrecisionType>:** Generate a velocity field from a input spatial transform

4 Users' Guide

From a user's point of view the most important file of our submission is the example application provided in `LogDomainDemonsRegistration.cxx`. The goal of this example is to provide a command-line tool to perform an intra-modality deformable registration with a chosen variant of the demons. This tool works in both 2D and 3D and can trivially be extended to other dimensions.

The user can choose input images, the variant of the demons that should be used and the type of output that should me stored. The image IO operations use standard ITK filters meaning that hall file formats supported by ITK can be used. This is true for both the images and the possible spatial transformation.

Below is the list of options of the command-line tool:

- **-f/–fixed-image=STRING:** Fixed image filename - mandatory argument
- **-m/–moving-image=STRING:** Moving image filename - mandatory argument
- **-b/–input-field=STRING:** Input velocity field filename - default empty
- **-p/–input-transform=STRING:** Input transform filename - default empty
- **-o/–output-image=STRING:** Output filename - default output.mha
- **-outputDef-field=STRING:** Output deformation field filename - default OUTPUTIMAGENAME-deformationField.mha
- **-outputVel-field=STRING:** Output velocity field filename - default OUTPUTIMAGENAME-velocityField.mha
- **-r/–true-field=STRING:** Ground truth deformation field filename (useful for synthetic experiments)
- **-i/–num-iterations=UINTx...xUINT:** List of number of iterations for each multi-scale pyramid level - default 15x10x5
- **-s/–vel-field-sigma=FLOAT:** Smoothing sigma for the velocity field (pixel units) - default 1.5
- **-g/–up-field-sigma=FLOAT:** Smoothing sigma for the update field (pixel units) - default 0.0
- **-l/–max-step-length=FLOAT:** Maximum length of an update vector (pixel units) - default 2.0
- **-a/–update-rule=UINT:** Type of update rule. 0: $\exp(v) \leftarrow \exp(v) \circ \exp(u)$ (log-domain), 1: $\exp(v) \leftarrow$ symmetrized($\exp(v) \circ \exp(u)$) (symmetric log-domain) - default 1
- **-t/–gradient-type=UINT:** Type of gradient used for computing the demons force.0 is symmetrized, 1 is fixed image, 2 is warped moving image, 3 is mapped moving image - default 0
- **-c/–num-bch-terms=UINT:** Number of terms in the Baker-Campbell-Hausdorff (BCH) expansion - default 2
- **-e/–use-histogram-matching:** Use histogram matching prior to registration (e.g. for different MR scanners)
- **-d/–verbose=UINT:** Algorithm verbosity. If a verbose mode is used the application will compute a set of statistics and write them to a file -default 1
- **-h:** Display a help message and exit

This command-line tool is used within a unit test triggered by CMake.

5 Conclusion

We have proposed an open-source ITK implementation of the symmetric demons algorithm. To the best of our knowledge, this is the first Insight Journal submission that allows for symmetric, a.k.a. inverse-consistent, non-rigid registration. The design of our implementation tries to follow the design of ITK and thus provides templated N-dimensional filters. The code should be easily integrated to ITK and provide reusable blocks.

A Software Requirements

You need to have the following software installed:

- Insight Toolkit: minimum version 3.10 - recommended version 3.14
- CMake: minimum version 2.6

References

[1] Vincent Arsigny, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. A Log-Euclidean framework for statistics on diffeomorphisms. In Rasmus Larsen, Mads Nielsen, and Jon Sporring, editors, *Proceedings of the 9th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI'06)*, volume 4190 of *Lecture Notes in Computer Science*, pages 924–931. Springer-Verlag, 2006. 1, 2.2

[2] John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, February 1994. 1

[3] M. Faisal Beg, Michael I. Miller, Alain Trouvé, and Laurent Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *International Journal of Computer Vision*, 61(2):139–157, February 2005. 1

[4] Matias Bossa, Monica Hernandez, and Salvador Olmos. Contributions to 3D diffeomorphic atlas estimation: Application to brain images. In Nicholas Ayache, Sébastien Ourselin, and Anthony J. Maeder, editors, *Proceedings of the 10th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI'07)*, volume 4792 of *Lecture Notes in Computer Science*, pages 667–674, Brisbane, Australia, October 2007. Springer-Verlag. 1, 2.4, 3

[5] Pascal Cachier, Eric Bardinet, Didier Dormont, Xavier Pennec, and Nicholas Ayache. Iconic feature based nonrigid registration: The PASHA algorithm. *Computer vision and image understanding*, 89(2–3):272–298, February 2003. 2.1, 2.1

[6] Christophe Chefd'hotel, Gerardo Hermosillo, and Olivier Faugeras. Flows of diffeomorphisms for multimodal image registration. In *Proceedings of the IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI'02)*, pages 753–756, 2002. 1

[7] Gary E. Christensen, Richard D. Rabitt, and Michael I. Miller. Deformable templates using large deformation kinematics. *IEEE Transactions on Image Processing*, 5(10):1435–1447, October 1996. 1

[8] Luis Ibáñez, Will Schroeder, Lydia Ng, and Josh Cates. *The ITK Software Guide*. Kitware, Inc., 2 edition, 2005. 1

[9] Sarang C. Joshi and Michael I. Miller. Landmark matching via large deformation diffeomorphisms. *IEEE Transactions on Image Processing*, 9(8):1357–1370, August 2000. 1

[10] Robert Mahony and Jonathan H. Manton. The geometry of the Newton method on non-compact Lie-groups. *Journal of Global Optimization*, 23(3):309–327, August 2002. 2.2

[11] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'04)*, April 2004. 2.1, 2.2

[12] Stephen Marsland and Carole Twining. Constructing diffeomorphic representations for the group-wise analysis of non-rigid registrations of medical images. *IEEE Transactions on Medical Imaging*, 23(8):1006–1020, 2004. 1

[13] Michael I. Miller, Sarang C. Joshi, and Gary E. Christensen. Large deformation fluid diffeomorphisms for landmark and image matching. In Arthur Toga, editor, *Brain Warping*, pages 115–131. Elsevier, 1998. 1

[14] Jean-Philippe Thirion. Image matching as a diffusion process: An analogy with Maxwell's demons. *Medical Image Analysis*, 2(3):243–260, 1998. 1

[15] Marc Vaillant, Michael I. Miller, Laurent Younes, and Alain Trouvé. Statistics on diffeomorphisms via tangent space representations. *Neuroimage*, 23(S1):161–169, 2004. 1

[16] Tom Vercauteren, Xavier Pennec, Ezio Malis, Aymeric Perchant, and Nicholas Ayache. Insight into efficient image registration techniques and the demons algorithm. In Nico Karssemeijer and Boudewijn P. F. Lelieveldt, editors, *Proceedings of Information Processing in Medical Imaging (IPMI'07)*, volume 4584 of *Lecture Notes in Computer Science*, pages 495–506, Kerkrade, The Netherlands, July 2007. Springer-Verlag. 2.1

[17] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Diffeomorphic demons using ITK's finite difference solver hierarchy. *Insight Journal – ISC/NA-MIC Workshop on Open Science at MICCAI*, November 2007. Available online with source code at http://hdl.handle.net/1926/510. 1

[18] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Non-parametric diffeomorphic image registration with the demons algorithm. In Nicholas Ayache, Sébastien Ourselin, and Anthony J. Maeder, editors, *Proceedings of the 10th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI'07)*, volume 4792 of *Lecture Notes in Computer Science*, pages 319–326, Brisbane, Australia, October 2007. Springer-Verlag. 1, 2.3

[19] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Symmetric log-domain diffeomorphic registration: A demons-based approach. In Dimitris Metaxas, Leon Axel, Gabor Fichtinger, and Gábor Székely, editors, *Proceedings of the 11th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI'08)*, volume 5241 of *Lecture Notes in Computer Science*, pages 754–761, New York, USA, September 2008. Springer-Verlag. (document), 1, 2.4, 3

[20] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Diffeomorphic demons: Efficient non-parametric image registration. *Neuroimage*, 45(1, Supp.1):S61–S72, March 2009. 1

[21] He Wang, Lei Dong, Jennifer O'Daniel, Radhe Mohan, Adam S. Garden, K. Kian Ang, Deborah A. Kuban, Mark Bonnen, Joe Y. Chang, and Rex Cheung. Validation of an accelerated 'demons' algorithm for deformable image registration in radiation therapy. *Physics in Medicine and Biology*, 50(12):2887–2905, 2005. 1

[22] Jocasta A Webb, Alexandre Guimond, Neil Roberts, Paul Eldridge, David W Chadwick, Jean Meunier, and Jean-Philippe Thirion. Automatic detection of hippocampal atrophy on magnetic resonance images. *Magnetic Resonance Imaging*, 17(8):1149–1161, 1999. 1