A versatile framework for images registration

Olivier Commowick

February 3, 2022

Abstract

Bla

Contents

We present in this appendix the generic registration algorithm used in several chapters (for different applications and image types). This algorithm relies on the block-matching algorithm introduced by [12] for rigid registration. It is in fact very versatile, usually requiring only the similarity measure between blocks (and a re-orientation scheme for images of oriented models such as DT or DCM images) to be defined to perform the registration. We explore in the first sections the individual elements of the registration and provide the general algorithms in Section 4.

1 Block-matching for medical images registration

We define the registration algorithm as the one that seeks a transformation T so that a floating image resampled by T, $F \circ T$, matches as much as possible a reference image R: $R \approx F \circ T$. With that goal in mind, the core part of the registration algorithm is based on an iterative framework which iterates three main steps:

- Define a set of blocks, i.e. a subset of voxels - often a cube around a given point in the image, on the reference image R. Each block is defined by its center x_i: $B(x_i) \equiv B_i$
- Match these blocks from image R to image F, i.e. find the best local transformation A_i such that a similarity measure $S(R(x), F \circ A(x))$ over the block B_i is optimal
- From the set of transformations obtained, compute an update global transformation δT then used to update the current global transformation T

This algorithm is the block-matching core that is central to all variants that are used in this document. This core is included in an iterative algorithm (see Section 4), itself often included in a multi-resolution (pyramidal) framework to first get back large displacements from coarse resolutions and then smaller / finer displacements from finer resolutions, in a robust and faster manner. This core part has been used in many algorithms ranging from linear [12, 8] to non linear, diffeomorphic registration [7], with various applications such as tensor registration [16]. It requires mainly the definition of a similarity metric S between blocks to be used on a specific modality. On scalar images [10], given the relatively small size of a block (thus containing two or few different tissues), a linear relationship between intensities is usually enough and a square correlation coefficient is used to match images. Apart from the similarity, several points need to be defined at the general level to have an overview of the algorithm:

- The nature of transformations A and T
- How to go from a set of transformations A_i and blocks B_i to a global (linear or non linear) update transformation δT?
- Ensuring (or not) a symmetric transformation

2 Which local transformations between blocks?

Local transformations between a block in R and F are generally assumed to be linear for two reasons: 1- the block is usually sufficiently small for this assumption to be true, and 2- the search space over which the transformation has to be estimated needs to remain small enough for computation time reasons. A transformation A_i in $\mathbb{R}^N$ is therefore represented as a $(N + 1) \times (N + 1)$ matrix. In the following, let us consider without loss of generality that $N = 3$.

2.1 Local translation

The most common transformation used in block-matching is a local translation, i.e. a move of the block in the three directions. In other words, A_i is represented as:

$$A_i = \begin{pmatrix} I_3 & t_i \\ 0 & 1 \end{pmatrix} \tag{1}$$

where I_3 is the 3×3 identity matrix and t_i is a $\mathbb{R}^3$ vector. In addition, its matrix logarithm is defined as a null matrix with a translation equal to that of A_i (t_i). This transformation has proven to be very useful and is also the fastest to estimate. Originally, these three parameters have been interpreted as a sliding of the block along the grid of voxels of the second image. The easiest optimization method for a local block is thus a discrete grid search on the voxel grid of the floating image, which has the advantage of not requiring any interpolation (and is thus fast). Although enough to recover globally sub-voxel precision thanks to the iterations of the block-matching algorithm, such an optimization may miss fine sub-voxel displacements. Recent algorithms [8] have therefore studied optimization over the whole $\mathbb{R}^3$ space, which may use a gradient-based algorithm if gradient of the similarity measure is available or gradient-free optimization such as the BOBYQA algorithm [14].

2.2 Rigid transformation and beyond

Among linear transformations, virtually any can be searched for between a block and an image. One particularly interesting and that we have explored in several recent works [8, 7] is the rigid transformation. It has several interesting properties that allow for its simple optimization and that will be useful to estimate global linear or non linear transformations. First of all, in homogeneous coordinates, A_i is represented as follows:

$$A_i = \begin{pmatrix} R_i & t_i \\ 0 & 1 \end{pmatrix} \tag{2}$$

where t_i is again a translation (also integrating the center of rotation), and R_i is a rotation matrix i.e. $R_i R_i^T = I_3$ and $||R_i||_F = 1$. Interestingly, Rodrigues' formulas allow for the explicit computation of its logarithm and exponential [4]. In particular, the matrix logarithm of a rigid matrix A is expressed as:

$$\log(A) = \begin{pmatrix} w_\times & l \\ 0 & 0 \end{pmatrix} \tag{3}$$

where $w_\times$ is the cross-product matrix of a vector of rotation angles $w = (w_0, w_1, w_2)^T$ and l an arbitrary $\mathbb{R}^3$ vector. For the explicit logarithm and exponential formulas, refer to [4]. The parameterization of the transformation through the matrix exponential of $\log(A)$ is particularly interesting as it allows for 1- a clear and separate depiction of the six degrees of freedom (three scalars of l and the angles in w) of the transformation (instead of the rigid rotation matrix where parameters are entangled), and 2- a direct expression of the rigid transformation in its Lie group structure, useful to perform transformation extrapolation.

Coming back to block-matching, one can now search for rigid transformations between a block and the floating image instead of just looking for translations. This is particularly useful for registration of images with articulated structures such as the spine [7]. However, since the search space is now much larger (six variables instead of three), a brute force discrete search as in the translation case is much longer and, depending on the similarity measure optimized, a gradient-based or gradient-free optimization is more adapted.

As a final remark, we presented here the search over rigid transformations but any other transformation with well defined parameters covering its entire spectrum may be used.

3 Global transformation extrapolation

We now have from the previous step (actually the real block-matching step), a set of blocks B_i defined on an image R and corresponding linear transformations $\hat{A}_i$ best matching them onto F. We now briefly detail how to use these local transformations to infer the update transformation δT. This procedure depends on the nature of local transformations and the nature of the global transformation sought.

3.1 Global linear transformations

Let us consider that we are looking for a global transformation that is linear. Two cases are still possible. First, if the $\hat{A}_i$ are translations, it is relatively easy to estimate any linear transformation best summarizing the local displacements obtained. Particularly, Chapter 8 of [13] explains very well how, through a linear least squares formulation, to estimate from a set of translations either a global translation, rigid or affine transformation and we refer the reader to these formulas for more details.

The second case arises when transformations $\hat{A}_i$ are more than just translations. In that case, we resort to the log-Euclidean framework for linear transformations [1] and thus formulate the least squares optimization directly on the matrix logarithms:

$$\log(\delta T) = \arg\max_{M} \sum_i || \log(\hat{A}_i) - M||^2 \tag{4}$$

This directly leads to $\log(\delta T)$ being the weighted average of the input log-transformations. For both cases, it is important to note that block-matching is not exempt from outliers which may degrade the obtained transformation. Numerous options may be applied to deal with this problem including weighting the individual $\hat{A}_i$ by the optimum similarity measure value they obtained, or performing robust estimation such as least trimmed squares or M-estimation [15].

3.2 Non linear transformations

When non linear transformations are sought, the extrapolation step is more difficult since many parameters come into play i.e. one 3D vector per voxel. In this section we will consider diffeomorphisms defined by their SVF as the final transformation being sought. While encompassing a reduced set of the diffeomorphisms that may be encountered (contrarily to LDDMM [3]), they have interesting properties that will be heavily used in this section, and that were well defined in the log-Euclidean frameworks for diffeomorphisms [2] and polyaffine transformations [1].

In particular, we consider that δT is defined by its SVF δS i.e. a vector field whose exponential is δT: $\delta T = \exp(\delta S)$. The goal is to extrapolate the “log-vector” at each voxel of δS from the sparse set of optimal block transformations $\hat{A}_i$ located at their block centers x_i. For this task, we will heavily use the matrix logarithm of linear transformations, which is explicitly defined for translations and rigid transformations.

As a side note to this class of transformations, it has been demonstrated that computing the SVF from a transformation T is computationally expensive [2]. In addition, SVF having nice properties for statistics computation, it is desirable to always keep T implicit and instead compute the SVF S. Transposing the transformation composition in that space however requires to use the BCH approximation [5, 17].

3.2.1 Gaussian extrapolation

The first and simplest way to extrapolate a dense SVF is to use Gaussian extrapolation [6, 9]. From the set of $\log(\hat{A}_i)$ and transformations locations x_i, we first construct a sparse field C where each voxel corresponding to x_i is affected by the displacement generated by $\log(\hat{A}_i)$: $C(x_i) = \log(\hat{A}_i)x_i$. The Gaussian extrapolation then builds a dense SVF δS from C and a sparse field W of weights w_i attributed to each pairing (for example the optimal value of the similarity measure): $W(x_i) = w_i$. The extrapolated δS is then defined as:

$$\delta S = \frac{G_\sigma * WC}{G_\sigma * W} \tag{5}$$

where σ is the standard deviation of the Gaussian extrapolation kernel G_σ. This extrapolation works perfectly in a region where enough input matchings are present, e.g. inside the brain. On the contrary, in regions far away from the blocks, this extrapolation is meaningless and may lead to artificially large deformations. To counter this effect, an additional post-processing is performed on the obtained SVF: when far enough from any matching ($G_\sigma * W$ below a certain threshold), δS is gradually set to identity (i.e. zero velocity field).

As for linear transformation computation, outliers in pairings need to be accounted for. A simple operation to do so is to compute at voxel locations x_i the norm of the difference between $\delta S(x_i)$ and $C(x_i)$: $r_i = ||C(x_i) - \delta S(x_i)||$. From these, the mean displacement difference r over the whole image is computed as well as the variance σ_r^2:

$$\left\{ \begin{array}{lcl} r & = & \frac{1}{N}\sum_i r_i \\ \sigma_r^2 & = & \frac{1}{N-1}\sum_i (r_i - r)^2 \end{array} \right. \tag{6}$$

We can reject pairings from C for which the residual $r_i > r + \alpha\sigma_r$ and recompute from it an outlier free δS.

3.2.2 M-Smoothing extrapolation

Although simple, the Gaussian extrapolation does not completely incorporate outlier rejection in its process in the sense that one would need to iterate over the rejection process to do so. We have therefore introduced in [7] an extrapolation approach similar to the M-smoothing filter proposed by [11]. This approach looks for the best log-transformation $\log(S_k)$ at each voxel of δS by minimizing the following criterion:

$$(\log S_1, \ldots, \log S_n) = \underset{\log S_1,\ldots,\log S_n}{\arg\min} \left[\sum_{k=1}^{n} \sum_{i \in V_k} w_i \rho\left(||\log S_k - \log A_i||^2\right) d\left(|x_k - x_i|^2\right) \right] \tag{7}$$

where ρ is robust error norm (typically linked to an M-estimator, here the Welsch function), V_k is a neighborhood around voxel i (note that the sum over $i \in V_k$ only considers voxels where a transformation A_i was estimated) and d is a spatial error norm. This criterion can be minimized using gradient descent which for a particular adaptive, data-dependent step size leads to the following update formula for each $\log S_k$:

$$\log S_k^{t+1} = \frac{\sum_{i \in V_k} w_i \rho'\left(||\log S_k - \log A_i||^2\right) d\left(|x_k - x_i|^2\right) \log A_i}{\sum_{i \in V_k} w_i \rho'\left(||\log S_k - \log A_i||^2\right) d\left(|x_k - x_i|^2\right)} \tag{8}$$

where ρ' acts as a tonal kernel, which for the Welsch function ρ is written as $\rho'(a^2) = \exp\left(-a^2/2\lambda^2\right)$, and d acts as a spatial kernel, here a Gaussian kernel: $d(a^2) = \exp\left(-a^2/2\sigma^2\right)$. The gradient descent is initialized with $\rho'(a^2) = 1$. These two kernels account simultaneously for the spatial proximity of A_i and its local agreement with other local transformations. From the obtained $\log S_k$, we finally obtain δS at each voxel by $\delta S(x_k) = \log S_k x_k$.

4 Asymmetric or symmetric registration

From the previous sections, we now have all the necessary elements (apart from a few such as being able to resample images or the similarity measure which is not the topic here) to perform the registration of two images. The final step is to combine all of these into an algorithm. At this stage, it is crucial to note that the block-matching core algorithm is intrinsically an asymmetric one: images R and F do not play the same role and reverting their use does not lead to the exact inverse of δT. Options are however available to ensure this, and this is why we detail three algorithms, going from no symmetry to more and more symmetry.

4.1 Asymmetric registration

The first, classical, registration built around the block-matching is the asymmetric one. It is described in Algorithm 1.

Algorithm 1 Asymmetric Block-Matching Registration Algorithm

```
1: for $p = 1...P$, iteration on pyramid levels, do
2:     for $l = 1...L$, iterations, do
3:         Resample $F$ with $T$
4:         Match $R$ and $F \circ T$: $\delta T \leftarrow$ block-match$(R, F \circ T)$
5:         Update $T$ by composing it with $\delta T$
6:         If needed, regularize $T$ (elastic-like)
```

From two images, a reference R and a floating image F, the algorithm seeks T by running a multi-resolution pyramid. At each step, the previously described components are put together to estimate update δT (or δS if the transformation computed is non linear) and compose it with the current estimate of T (BCH approximation for SVF). In this case, R and F clearly have an asymmetric role, blocks being always defined on R and only F being resampled.

4.2 Symmetric registration

In the previous algorithm, blocks are always defined on R while F is always the floating image. This second algorithm, presented in Algorithm 2, aims at symmetrizing this definition of blocks and ensuring that the obtained transformation when inverting F and R roles is the same up to an inverse, hereafter called inverse symmetry.

Again, T and δT are replaced by S and δS when dealing with non linear transformations. In this algorithm, blocks are defined at each step both on R and on F and used to estimate two asymmetric updates: δT_F and δT_R (respectively δS_F and δS_R for non linear transformations). To account for these two updates and ensure inverse symmetry, the composition step is modified and preceded by an averaging of the two updates (for linear transformations using the matrix logarithm, and for SVF the log-Euclidean framework):

Algorithm 2 Symmetric Block-Matching Registration Algorithm

```
1: for $p = 1...P$, iteration on pyramid levels, do
2:   for $l = 1...L$, iterations, do
3:     Resample $F$ with $T$ and $R$ with $T^{-1}$
4:     Match $R$ and $F \circ T$: $\delta T_F \leftarrow$ block-match$(R, F \circ T)$
5:     Match $F$ and $R \circ T^{-1}$: $\delta T_R \leftarrow$ block-match$(F, R \circ T^{-1})$
6:     Compute the transform update $\delta T$ from $\delta T_F$ and $\delta T_R$
7:     Update $T$ by composing it with $\delta T$
8:     If needed, regularize $T$ (elastic-like)
```

$$\delta T = \exp\left(\frac{1}{2}\left[\log(\delta T_R) - \log(\delta T_F)\right]\right) \tag{9}$$

$$\delta S = \frac{1}{2}\left(\delta S_R - \delta S_F\right) \tag{10}$$

4.3 Kissing symmetric registration

In direct symmetry, the images roles are not completely symmetric. In fact, one image in each way (from $F \circ T$ to R and from $R \circ T^{-1}$ to F) is never resampled: the transformation is applied only to one image at a time. Kissing symmetry instead seeks the transformation T so that $R \circ T^{-1}$ and $F \circ T$ match. T now encodes the half transformation between the images: this amounts to looking for an intermediate position in between the two images by moving both of them towards each other, thereby fully symmetrizing their roles. This registration is presented in Algorithm 3.

Algorithm 3 Kissing Symmetric Block-Matching Registration Algorithm

```
1: for $p = 1...P$, iteration on pyramid levels, do
2:   for $l = 1...L$, iterations, do
3:     Resample $F$ with $T$ and $R$ with $T^{-1}$
4:     Match $R \circ T^{-1}$ and $F \circ T$: $\delta T_F \leftarrow$ block-match$(R \circ T^{-1}, F \circ T)$
5:     Match $F \circ T$ and $R \circ T^{-1}$: $\delta T_R \leftarrow$ block-match$(F \circ T, R \circ T^{-1})$
6:     Compute the half transform update $\delta T$ from $\delta T_F$ and $\delta T_R$
7:     Update $T$ by composing it with $\delta T$
8:     If needed, regularize $T$ (elastic-like)
```

As for direct symmetry in Algorithm 2, the composition step is modified to compute δT, respectively δS, from the asymmetric updates:

$$\delta T = \exp\left(\frac{1}{4}\left[\log(\delta T_R) - \log(\delta T_F)\right]\right) \tag{11}$$

$$\delta S = \frac{1}{4}\left(\delta S_R - \delta S_F\right) \tag{12}$$

The only difference with direct symmetry is here 1/4 instead of 1/2 to account for the fact that we are looking for a transformation bringing the two images on a middle point where they match. Applying the final transformation T to F, is as simple as taking the square transformation (or multiply it by 2 in the “log-space”).

References

[1] Vincent Arsigny, Olivier Commowick, Nicholas Ayache, and Xavier Pennec. A Fast and Log-Euclidean Polyaffine Framework for Locally Linear Registration. *Journal of Mathematical Imaging and Vision*, 33(2):222–238, 2009.

[2] Vincent Arsigny, Olivier Commowick, Xavier Pennec, and Nicholas Ayache. A Log-Euclidean Framework for Statistics on Diffeomorphisms. In *Proc. of the 9th International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI'06), Part I*, volume 4190 of *LNCS*, pages 924–931, Copenhaguen, Denmark, 2006.

[3] Mirza Faisal Beg, Michael 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, 2005.

[4] Jose-Luis Blanco. A tutorial on SE(3) transformation parameterizations and on-manifold optimization. Technical report, University of Malaga, 2010.

[5] Matias Bossa, Monica Hernandez, and Salvador Olmos. Contributions to 3D Diffeomorphic Atlas Estimation: Application to Brain Images. In *Medical Image Computing and Computer-Assisted Intervention*, volume 4791 of *LNCS*, pages 667–674, 2007.

[6] Olivier Commowick. *Design and Use of Anatomical Atlases for Radiotherapy.* PhD thesis, Université Nice Sophia Antipolis, February 2007.

[7] Olivier Commowick, Nicolas Wiest-Daesslé, and Sylvain Prima. Automated diffeomorphic registration of anatomical structures with rigid parts: application to dynamic cervical MRI. In *15th International Conference on Medical Image Computing and Computer Assisted Intervention*, volume 15 of *LNCS*, pages 163–70, Nice, France, October 2012. Springer.

[8] Olivier Commowick, Nicolas Wiest-Daesslé, and Sylvain Prima. Block-Matching Strategies for Rigid Registration of Multimodal Medical Images. In *9th IEEE International Symposium on Biomedical Imaging (ISBI'2012)*, pages 700–703, Barcelona, Spain, May 2012.

[9] Vincent Garcia, Olivier Commowick, and Grégoire Malandain. A Robust and Efficient Block Matching Framework for Non Linear Registration of Thoracic CT Images. In *Grand Challenges in Medical Image Analysis (MICCAI workshop)*, pages 1–10, Beijing, China, China, 2010.

[10] Grégoire Malandain, Eric Bardinet, Koen Nelissen, and Wim Vanduffel. Fusion of autoradiographs with an MR volume using 2-D and 3-D linear transformations. *Neuroimage*, 23(1):111–127, 2004.

[11] P. Mrazek, J. Weickert, and A. Bruhn. *On Robust Estimation and smoothing with Spatial and Tonal Kernels*, pages 335–352. Springer, 2006.

[12] Sébastien Ourselin, Alexis Roche, Sylvain Prima, and Nicholas Ayache. Block matching: A general framework to improve robustness of rigid registration of medical images. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, volume 1935 of *LNCS*, pages 557–566. Springer, 2000.

[13] Xavier Pennec. *L'incertitude dans les problèmes de reconnaissance et de recalage – Applications en imagerie médicale et biologie moléculaire.* Thèse de sciences (phd thesis), Ecole Polytechnique, December 1996.

[14] M.J.D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Centre for Mathematical Sciences, University of Cambridge, UK, 2009.

[15] Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection.* Wiley Series in Probability and Mathematical Statistics, 1987.

[16] Ralph O. Suarez, Olivier Commowick, Sanjay P. Prabhu, and Simon K. K. Warfield. Automated delineation of white matter fiber tracts with a multiple region-of-interest approach. *NeuroImage*, 59(4):3690–3700, February 2012.

[17] Tom Vercauteren, Xavier Pennec, Aymeric Perchant, and Nicholas Ayache. Symmetric log-domain diffeomorphic registration: A demons-based approach. In *Medical Image Computing and Computer-Assisted Intervention*, volume 5241 of *LNCS*, pages 754–761, 2008.