Numer Algor 2018 to appear (Springer) https://doi.org/10.1007/s11075-018-0486-2

Improved Optimization Methods for Image Registration Problems

Ke Chen · Geovani Nunes Grapiglia · Jinyun Yuan · Daoping Zhang

Abstract In this paper we propose new multilevel optimization methods for minimizing continuously differentiable functions obtained by discretizing models for Image Registration problems. These multilevel schemes rely on a novel Two-Step Gauss-Newton method, in which a second step is computed within each iteration by minimizing a quadratic approximation of the objective function over a certain two-dimensional subspace. Numerical results on Image Registration problems show that the proposed methods can outperform the standard multilevel Gauss-Newton method.

Keywords Image Registration · Multilevel Strategy · Gauss-Newton Method · Subspace Methods

This work was partially supported by UK EPSRC (grants EP/K036939/1 and EP/N014499/1) and by the Newton Research Collaboration Programme (grant NRCP 1617/6/187).

K. Chen,
Centre for Mathematical Imaging Techniques and Department of Mathematical Sciences. The University of Liverpool, Merseyside L697ZL, Liverpool, UK
k.chen@liverpool.ac.uk

G. N. Grapiglia,
Departamento de Matemática, Universidade Federal do Paraná, Centro Politécnico, Cx. postal 19.081, 81531-980, Curitiba, Paraná, Brazil
grapiglia@ufpr.br

J. Yuan,
Departamento de Matemática, Universidade Federal do Paraná, Centro Politécnico, Cx. postal 19.081, 81531-980, Curitiba, Paraná, Brazil
jin@ufpr.br

D. Zhang,
Centre for Mathematical Imaging Techniques and Department of Mathematical Sciences. The University of Liverpool, Merseyside L697ZL, Liverpool, UK
Daoping.Zhang@liverpool.ac.uk

1 Introduction

Image Registration is the task of overlaying two or more images of the same subject taken at different times, from different viewpoints or by different sensors. The goal of registration is to find a function that maps points of one image to the corresponding points of the other image, providing a geometric alignment between the images. This process compensates the motion of the subject or some difference between the sensors, allowing the images to be compared and analyzed in a common reference frame [6]. A very important application is the registration of medical images obtained from Computed Tomography (CT), Magnetic Resonance Imaging (MRI) or Ultrasound (US), for example. In this context, Image Registration helps in the direct comparison of images taken at different stages of progression of a disease (e.g., a tumor growth), which is essential for the correct diagnosis of the disease, for planning the treatment and for monitoring the response of the patient [11].

Mathematically, the Image Registration problem can be described in the following way. Consider two images, R and T. Image R (called reference) is kept unchanged, while image T (called template) is kept transformed. These images can be viewed as compactly supported functions $R, T : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^d$ is a bounded convex domain and d is the dimension of the images. Without loss of generality, in this work we shall consider $d = 2$. For each pixel $x = (x_1, x_2) \in \Omega$, the values $R(x)$ and $T(x)$ describe the darkness of x in images R and T, respectively. The goal of registration is to find a *displacement field* $\mathbf{u} : \mathbb{R}^2 \to \mathbb{R}^2$ such that $T(x + \mathbf{u}(x))$ is *similar* to $R(x)$ with respect to some metric. Let us denote by $T(\mathbf{u})$ the function given by

$$T(\mathbf{u})(x) = T(x + \mathbf{u}(x)).$$

Then, given a metric $D(.\,, .)$ for measuring the dissimilarity between any two images, the image registration problem can be stated as the following optimization problem:

$$\min_{\mathbf{u}} D(R, T(\mathbf{u})). \tag{1}$$

A usual choice for $D(.\,, .)$ is the L^2-norm

$$D(R, T(\mathbf{u})) = \frac{1}{2} \int_{\Omega} \left(T(x + \mathbf{u}(x)) - R(x)\right)^2 \, \mathrm{d}\Omega. \tag{2}$$

Problem (1) is an ill-posed problem. Thus, to avoid meaningless solutions, a regularization term is included in the objective function of (1). The resulting problem is

$$\min_{\mathbf{u}} J(\mathbf{u}) \equiv D(R, T(\mathbf{u})) + \lambda S(\mathbf{u}), \tag{3}$$

where $\lambda > 0$ is a regularization parameter. The role of the regularizer is to modify problem (1) such that it becomes solvable. A usual choice for $S(\,.\,)$ is

$$S(\mathbf{u}) = \frac{1}{2} \int_{\Omega} |B(\mathbf{u}(x))|^2 \, \mathrm{d}\Omega. \tag{4}$$

where B is some differential operator.

Note that (3) is an infinite-dimensional optimization problem. In general, this type of problem cannot be solved analytically, requiring therefore the use of numerical schemes. There are two main numerical approaches to solve infinite-dimensional optimization problems. The first approach, referred as *optimize-then-discretize*, consists in differentiate the objective function (3) to obtain the continuous Euler-Lagrange equation, discretize these equations, and then solve numerically the resulting finite-dimensional equations. The second approach, referred as *discretize-then-optimize*, consists in discretize the objective function (3) and then solve the resulting finite-dimensional optimization problem by some optimization algorithm. Usually, the discrete optimization problem has a very large number of variables. To solve it, several researchers apply a Gauss-Newton method with a line-search (e.g., Armijo line-search) embedded in a *coarse-to-fine* Multilevel optimization strategy. In this strategy, images are registered progressively from lower resolutions to higher resolutions, providing (by interpolation) the initial point for the finest resolution. See, for example, [1, 7, 14].

In this paper, we propose two simple techniques to improve the performance of the Multilevel Gauss-Newton algorithm on Image Registration Problems. The first technique consists in the possible use of a second step within each iteration of the Gauss-Newton method. This step is computed by minimizing a quadratic approximation of the objective function over a two-dimensional subspace. This subspace is spanned by the steepest descent direction and by the L-BFGS direction with respect to the current point given by the Gauss-Newton step. If such *subspace step* provides any decrease in the objective function, it is accepted, otherwise it is discarded. The second technique is a modification of the standard coarse-to-fine Multilevel strategy. At each level, instead of using directly the interpolated solution of the previous level as the initial point, we try to find a better initial point by minimizing a quadratic approximation of the objective function over the subspace spanned by the interpolated solutions of *all* the previous levels. If this new point results in a decrease of the objective function value, it is accepted as the new initial point, otherwise we proceed as in the standard coarse-to-fine approach.

The paper is organized as follows. In Section 2, we describe the methods resulting from the two proposed techniques. We also present a convergence analysis for these schemes. In Section 3, we report the results of extensive numerical experiments showing the effectiveness of our new methods. Finally, in Section 4, we summarize the contributions of this work and indicate some directions for future research.

2 Optimization Methods

In this section we present the optimization methods resulting from the use of our two novel subspace techniques, which are inspired by [13]. For clarity, we start by describing the standard Multilevel Gauss-Newton algorithm.

2.1 Multilevel Gauss-Newton Algorithm

Consider the optimization problem

$$\min_{\mathbf{u}\in\mathcal{V}} J(\mathbf{u}), \tag{5}$$

where J is a function from an infinite-dimensional vector space $\mathcal{V}$ to $\mathbb{R}$. Let $\mathcal{V}_l$ be a finite-dimensional subspace of $\mathcal{V}$ with basis $\left\{\phi_l^{(j)}\right\}_{j=1}^{n_l}$ at grid level l, where n_l is the dimension of $\mathcal{V}_l$. By definition, it means that given any $\mathbf{u}_l \in \mathcal{V}_l$ there exists a vector $u_l = (u_l^{(1)}, \ldots, u_l^{(n_l)}) \in \mathbb{R}^{n_l}$ such that

$$\mathbf{u}_l = \sum_{j=1}^{n_l} u_l^{(j)} \phi_l^{(j)}. \tag{6}$$

Suppose that we have nested spaces $\mathcal{V}_{N_0} \subset \ldots \subset \mathcal{V}_{N-1} \subset \mathcal{V}_N \subset \mathcal{V}$. For each level l, we shall consider the discrete functional $J_l : \mathbb{R}^{n_l} \to \mathbb{R}$ given by

$$J_l(u_l) = J(\mathbf{u}_l), \tag{7}$$

where $\mathbf{u}_l$ is computed by (6). Thus, on level l, the discretized version of (5) is

$$\min_{u_l\in\mathbb{R}^{n_l}} J_l(u_l). \tag{8}$$

In the discretize-then-optimize approach, our goal is to obtain an approximate solution of (5) by solving iteratively its discrete version (8) for $l = N$. This can be done by using the coarse-to-fine multilevel strategy, in which problems of the form (8) are solved consectively for $l = N_0, \ldots, N-1, N$, and the initial point $u_{l+1,0}$ for the discrete problem on level $l+1$ is generated by "prolongating" the solution u_l^* obtained on level l. We shall denote by P_l^{l+1} the prolongation operator from level l to level $l+1$. Thus, in the coarse-to-fine strategy we have

$$u_{l+1,0} = P_l^{l+1} u_l^*, \quad l = N_0, \ldots, N-1. \tag{9}$$

Given an initial guess $u_{l,0}$ for the solution of (8), Newton's Method generates a sequence $\{u_{l,k}\}$ by the rule $u_{l,k+1} = u_{l,k} + t_{l,k} d_{l,k}$, with

$$\nabla^2 J_l(u_{l,k}) d_{l,k} = -\nabla J_l(u_{l,k}), \tag{10}$$

where $\nabla J_l(u_{l,k})$ and $\nabla^2 J_l(u_{l,k})$ are the gradient and the hessian of J_l at $u_{l,k}$, respectively. However, in many situations the structure of the objective J_l gives

$$\nabla^2 J_l(u_{l,k}) = H_{l,k} + A_{l,k}, \tag{11}$$

where $H_{l,k} \in \mathbb{R}^{n_l \times n_l}$ is an "easy" to compute symmetric positive-definite matrix, while $A_{l,k}$ is "difficult" to compute. For these cases, the common approach is the Gauss-Newton Method, where sequence $\{u_{l,k}\}$ is defined similarly but, in contrast to (10), $d_{l,k}$ is obtained by solving the linear system

$$H_{l,k} d_{l,k} = -\nabla J_l(u_{l,k}). \tag{12}$$

If the stepsize $t_{l,k}$ is computed by the Armijo line-search, we have the following algorithm.

Algorithm 1 (Gauss-Newton Method): $u_l^* = GN(l, u_{l,0})$

Step 0 Compute $\nabla J_l(u_{l,0})$ and $H_{l,0} \approx \nabla^2 J_l(u_{l,0})$. Set $\eta = 10^{-4}$ and $k := 0$.
Step 1 If $u_{l,k}$ satisfies the stopping rules, stop and return $u_l^* = u_{l,k}$. Otherwise, go to Step 2.
Step 2 Compute $d_{l,k}$ by solving the Gauss-Newton linear system

$$H_{l,k} d_{l,k} = -\nabla J_l(u_{l,k}). \tag{13}$$

Step 3 Find the smallest integer $i_k \geq 0$ such that $t_{l,k} = (0.5)^{i_k}$ satisfies

$$J_l(u_{l,k} + t_{l,k} d_{l,k}) \leq J_l(u_{l,k}) + \eta t_{l,k} \nabla J_l(u_{l,k})^T d_{l,k}. \tag{14}$$

Step 4 Set $u_{l,k+1} = u_{l,k} + t_{l,k} d_{l,k}$ and compute $\nabla J_l(u_{l,k+1})$ and $H_{l,k+1} \approx \nabla^2 J_l(u_{l,k+1})$.
Step 5 Set $k := k + 1$ and go back to Step 1.

Remark 1 In the context of Image Registration problems, at Step 2 it is common the use of the following stopping rules:

$$|J_l(u_{l,k}) - J_l(u_{l,k})| \leq 10^{-3}(1 + |J_l(u_{l,0})|) \quad \text{if } k > 0, \tag{15}$$

$$\|u_{l,k} - u_{l,k-1}\|_2 \leq 10^{-2}(1 + \|u_{l,0}\|_2) \quad \text{if } k > 0, \tag{16}$$

$$\|\nabla J_l(u_{l,k})\|_2 \leq 10^{-2}(1 + |J_l(u_{l,0})|), \tag{17}$$

$$\|\nabla J_l(u_{l,k})\|_2 \leq \varepsilon, \tag{18}$$

and

$$k \geq k_{max}. \tag{19}$$

Specifically, the execution of the algorithm is interrupted when all conditions (15)-(17) are satisfied or when any of the conditions (18) and (19) holds.

Very often, the discretization of Image Registration problems generate problems where n_N is very big (e.g., $n_N > 10^6$). Thus, when we apply the Gauss-Newton Method to the discrete problem in the finest grid

$$\min_{u_l \in \mathbb{R}^{n_N}} J_N(u_N),$$

the solution of the linear system (13) at each iteration can consume a lot of time. Consequently, if the method starts from a bad initial point, it will take many iterations to reach a solution, which will make the total running time very big. However, if the method starts from a good initial point, it will take fewer iterations to reach a solution, which can lead to a significant reduction in the total running time. This is the motivation behind the coarse-to-fine multilevel strategy, which is a technique to generate initial points. The Multilevel Gauss-Newton Method can be summarized in the following way.

Algorithm 2 (Multilevel Gauss-Newton Method)

Step 0 Set $u_{N_0,0} = (0, \ldots, 0) \in \mathbb{R}^{N_0}$ and $l := N_0$ (coarsest level).
Step 1 Compute $u_l^* = GN(l, u_{l,0})$.
Step 2 If $l = N$ (finest level), stop and return u_N^*. Otherwise, go to Step 3.
Step 3 Set $u_{l+1,0} = P_l^{l+1} u_l^*$, $l := l + 1$ and go back to Step 1.

2.2 Two-Step Gauss-Newton Algorithm

In order to enhance the performance of the Gauss-Newton method, we consider the use of a second step after the Gauss-Newton step within each iteration. If we obtain any reduction in the objective function value, the new step is accepted. Otherwise, the new step is rejected. Since we are dealing with large-scale problems, this additional step must be cheap to compute. Therefore, we propose the following subspace procedure. Denote by $\hat{u}_{l,k+1}$ the Gauss-Newton iterate computed at Step 4 of Algorithm 1, that is,

$$\hat{u}_{l,k+1} = u_{l,k} + t_{l,k} d_{l,k}, \text{ with } H_{l,k} d_{l,k} = -\nabla J_l(u_{l,k}).$$

Let $\hat{H}_{l,k+1}$ be the Gauss-Newton approximation to $\nabla^2 J_l(\hat{u}_{l,k+1})$ and consider the quadratic model of J_l around $\hat{u}_{l,k+1}$:

$$m_l(\hat{u}_{l,k+1} + d) \equiv J_l(\hat{u}_{l,k+1}) + \nabla J_l(\hat{u}_{l,k+1})^T d + \frac{1}{2} d^T \hat{H}_{l,k+1} d.$$

We compute the second step $\hat{d}_{l,k+1}$ by minimizing $m_l(\hat{u}_{l,k+1} + d)$ over the subspace

$$\mathcal{S}_{l,k+1} = \text{span}\left(\left\{ d_{l,k+1}^{SD}, d_{l,k+1}^{QN} \right\} \right),$$

where $d_{l,k+1}^{SD} = -\nabla J_l(\hat{u}_{l,k+1})$ and $d_{l,k+1}^{QN} = -B_{l,k+1} \nabla J_l(\hat{u}_{l,k+1})$ with $B_{l,k+1}$ being the approximation to $\left(\nabla^2 J_l(\hat{u}_{l,k+1})\right)^{-1}$ given by the Limited-memory BFGS (L-BFGS) formula [5]. More specificaly,

$$\hat{d}_{l,k+1} = \alpha_1 d_{l,k+1}^{SD} + \alpha_2 d_{l,k+1}^{QN}, \tag{20}$$

where $\alpha = (\alpha_1, \alpha_2) \in \mathbb{R}^2$ is a solution of the quadratic minimization problem

$$\min_{\alpha \in \mathbb{R}^2} \; g_{l,k+1}^T \alpha + \frac{1}{2} \alpha^T Q_{l,k+1} \alpha, \tag{21}$$

with

$$g_{l,k+1} = \begin{bmatrix} \nabla J_l(\hat{u}_{l,k+1})^T d_{l,k+1}^{SD} \\ \nabla J_l(\hat{u}_{l,k+1})^T d_{l,k+1}^{QN} \end{bmatrix} \tag{22}$$

and

$$Q_{l,k+1} = \begin{bmatrix} (d_{l,k+1}^{SD})^T \hat{H}_{l,k+1} d_{l,k+1}^{SD} & (d_{l,k+1}^{SD})^T \hat{H}_{l,k+1} d_{l,k+1}^{QN} \\ (d_{l,k+1}^{QN})^T \hat{H}_{l,k+1} d_{l,k+1}^{SD} & (d_{l,k+1}^{QN})^T \hat{H}_{l,k+1} d_{l,k+1}^{QN} \end{bmatrix}. \tag{23}$$

Problem (21) is equivalent to the 2×2 linear system

$$Q_{l,k+1}\alpha = -g_{l,k+1}, \tag{24}$$

which makes the computation of the second step $\hat{d}_{l,k+1}$ in (20) very cheap. If $J_l(\hat{u}_{l,k+1} + \hat{d}_{l,k+1}) < J_l(\hat{u}_{l,k+1})$, then we accept the new step and we define $u_{l,k+1} = \hat{u}_{l,k+1} + \hat{d}_{l,k+1}$. Otherwise, we reject the new step and we define $u_{l,k+1} = \hat{u}_{l,k+1}$. The resulting Two-Step Gauss-Newton method can be summarized as follows.

Algorithm 3 (Two-Step GN Method): $u_l^* = 2SGN(l, u_{l,0})$

Step 0 Compute $\nabla J_l(u_{l,0})$ and $H_{l,0} \approx \nabla^2 J_l(u_{l,0})$. Set $B_{l,0} = I$, $m = 3$, $\eta = 10^{-4}$ and $k := 0$.
Step 1 If $u_{l,k}$ satisfies the stopping rules, stop and return $u_l^* = u_{l,k}$. Otherwise, go to Step 2.
Step 2 Compute $d_{l,k}$ by solving the Gauss-Newton linear system

$$H_{l,k}d_{l,k} = -\nabla J_l(u_{l,k}). \tag{25}$$

Step 3 Find the smallest integer $i_k \geq 0$ such that $t_{l,k} = (0.5)^{i_k}$ satisfies

$$J_l(u_{l,k} + t_{l,k}d_{l,k}) \leq J_l(u_{l,k}) + \eta t_{l,k}\nabla J_l(u_{l,k})^T d_{l,k}. \tag{26}$$

Step 4 Set $\hat{u}_{l,k+1} = u_{l,k} + t_{l,k}d_{l,k}$ and compute $\nabla J_l(\hat{u}_{l,k+1})$ and $\hat{H}_{l,k+1} \approx \nabla^2 J_l(\hat{u}_{l,k+1})$.
Step 5 Set $s_{l,k} = \hat{u}_{l,k} - u_{l,k}$ and $y_{l,k} = \nabla J_l(\hat{u}_{l,k+1}) - \nabla J_l(u_{l,k})$.
Step 6 (L-BFGS direction) Let $\hat{m} = \min\{k, m-1\}$. If $k > 0$, set $B_{l,0} = (s_{l,k-1}^T y_{l,k-1}/(y_{l,k-1})^T y_{l,k-1})I$. Update $B_{l,0}$ $\hat{m}+1$ times using the pairs $\{s_{l,j}, y_{l,j}\}_{j=k-\hat{m}}^{k}$, i.e., let

$$\begin{aligned} B_{l,k+1} = &\left(V_k^T \dots V_{k-\hat{m}}^T\right) B_{l,0} \left(V_{k-\hat{m}} \dots V_k\right) \\ &+\rho_{k-\hat{m}} \left(V_k^T \dots V_{k-\hat{m}+1}^T\right) s_{l,k-\hat{m}}(s_{l,k-\hat{m}})^T \left(V_{k-\hat{m}+1} \dots V_k\right) \\ &+\rho_{k-\hat{m}+1} \left(V_k^T \dots V_{k-\hat{m}+2}^T\right) s_{l,k-\hat{m}+1}(s_{l,i-\hat{m}+1})^T \left(V_{k-\hat{m}+2} \dots V_k\right) \\ &\vdots \\ &+\rho_k s_{l,k}(s_{l,k})^T, \end{aligned}$$

with $\rho_j = 1/(s_{l,j})^T y_{l,j}$ and $V_j = I - \rho_j y_{l,j}(s_{l,j})^T$. Compute $d_{l,k+1}^{QN} = -B_{l,k+1}\nabla J_l(\hat{u}_{l,k+1})$.
Step 7 (Second Step) Let $d_{l,k+1}^{SD} = -\nabla J_l(\hat{u}_{l,k+1})$, compute $\alpha = (\alpha_1, \alpha_2) \in \mathbb{R}^2$ by solving (24) and then set $\hat{d}_{l,k+1} = \alpha_1 d_{l,k+1}^{SD} + \alpha_2 d_{l,k+1}^{QN}$.
Step 8 If $J_l(\hat{u}_{l,k+1} + \hat{d}_{l,k+1}) < J_l(\hat{u}_{l,k+1})$, set $u_{l,k+1} = \hat{u}_{l,k+1} + \hat{d}_{l,k+1}$ and compute $\nabla J_l(u_{l,k+1})$ and $H_{l,k+1} \approx \nabla^2 J_l(u_{l,k+1})$. Otherwise, set $u_{l,k+1} = \hat{u}_{l,k+1}$ and $H_{l,k+1} = \hat{H}_{l,k+1}$.
Step 9 Set $k := k + 1$ and go back to Step 1.

In practice, the matrices $B_{l,k+1}$ in the L-BFGS scheme are not formed explicitly. At each iteration all that we need is to compute the product $d_{l,k+1}^{QN} = -B_{l,k+1}\nabla J_l(\hat{u}_{l,k+1})$. This can be done efficiently in a matrix-free fashion by using the following algorithm [10]:

Algorithm 4. (Direction finding in L-BFGS)

Step 0 Set $q = \nabla J_l(\hat{u}_{l,k+1})$.

Step 1 For $j = (i-1) : (-1) : (i - \hat{m})$ do

$$\alpha_j = \rho_j (s^j)^T q$$

$$q = q - \alpha_j y^j$$

Step 2 Set $r = B_{l,0} q$.

Step 3 For $j = (i - \hat{m}) : 1 : (i-1)$ do

$$\beta = \rho_j (y^j)^T r$$

$$r = r + (\alpha_j - \beta) s^j$$

Step 4 Set $d_{l,k+1}^{QN} = -r$ and STOP.

Finally, if at Step 1 of Algorithm 2 we replace Gauss-Newton method by our new Two-Step Gauss-Newton method we obtain the multilevel algorithm below.

Algorithm 5 (Multilevel Two-Step Gauss-Newton Method)

Step 0 Set $u_{N_0,0} = (0, \ldots, 0) \in \mathbb{R}^{N_0}$ and $l := N_0$ (coarsest level).
Step 1 Compute $u_l^* = 2SGN(l, u_{l,0})$.
Step 2 If $l = N$ (finest level), stop and return u_N^*. Otherwise, go to Step 3.
Step 3 Set $u_{l+1,0} = P_l^{l+1} u_l^*$, $l := l + 1$ and go back to Step 1.

2.3 Convergence Analysis

The analysis of Algorithms 1 and 3 in a constrained setting can be done in an unified framework. In fact, consider the finite-dimensional optimization problem

$$\min_{u \in \mathbb{R}^n} \; J(u), \tag{27}$$

$$\text{s. t. } u \in X, \tag{28}$$

where $J : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function and $X \subset \mathbb{R}^n$ is an open set[1]. Clearly, problem (27)-(28) may have no solution. Thus, we seek for iterative methods that generate sequences $\{u_k\} \subset X$ of feasible points such that $\{J(u_k)\}$ is monotonically decreasing. By incorporating constraint (28) within the Armijo line-search in Algorithms 1 and 3 (and omitting the level index l), the resulting algorithms can be seen as particular cases of the following framework.

Algorithm A. (Feasible Direction Method)

Step 0 Given $u_0 \in X$, $B_0 \in \mathbb{R}^{n\times n}$ symmetric and positive-definite and $\eta \in (0,1)$, set $k := 0$.

Step 1 Compute $d_k = -B_k \nabla J(u_k)$.

Step 2 Find the smallest $i_k \geq 0$ such that $t_k = (0.5)^{i_k}$ ensures

$$J(u_k + t_k d_k) \leq J(u_k) + \eta t_k \nabla J(u_k)^T d_k \quad \text{and} \quad u_k + t_k d_k \in X. \tag{29}$$

Define $\hat{u}_{k+1} = u_k + t_k d_k$.

Step 3 Find $u_{k+1} \in X$ such that $J(u_{k+1}) \leq J(\hat{u}_{k+1})$, choose $B_{k+1} \in \mathbb{R}^{n\times n}$ symmetric and positive-definite, set $k := k+1$ and go back to Step 1.

Remark 2 In Algorithm A, B_k is the inverse of the Gauss-Newton matrix, that is, $B_k = H_k^{-1}$. To better see the correspondence between Algorithm A and Algorithms 1 and 3, note that in Algorithm 1 we set $u_{k+1} = \hat{u}_{k+1}$ for all k, while in Algorithm 3 we may have $u_{k+1} \neq \hat{u}_{k+1}$ if the second step is successful.

We shall study the worst-case complexity and global convergence properties of Algorithm A. By worst-case complexity we mean an upper bound on the maximum number of iterations that Algorithm A may take to find an approximate critical point of J or a point near to the boundary of the feasible set. Our analysis is an adaptation of the analysis of Nesterov [9] for the Gradient Method. Consider the following assumptions:

A1 The objective $J : \mathbb{R}^n \to \mathbb{R}$ is differentiable and $\nabla J : \mathbb{R}^n \to \mathbb{R}^n$ is L-Lipschitz:

$$\|\nabla J(w) - \nabla J(u)\| \leq L\|w - u\|, \ \forall w, u \in \mathbb{R}^n.$$

A2 The set $L(u_0) = \{u \in \mathbb{R}^n \mid J(u) \leq J(u_0)\}$ is compact.

A3 There exist constants $c_1 \geq c_0 > 0$ such that

$$c_0 I \preceq B_k \preceq c_1 I \ \forall k.$$

[1] In Image Registration Problems, it is common the inclusion of the constraint $\det \nabla y > 0$, where $y(x) = x + \mathbf{u}(x)$.

The next lemma shows that if $\nabla J(u_k) \neq 0$, then there exists $i_k \geq 0$ such that conditions (29) hold. Therefore, Step 2 of Algorithm A is well-defined. The proof is based on elementary analysis arguments and it is included here for completeness.

Lemma 1 *Suppose that A1 holds. Given $\bar{u} \in X$, $B \in \mathbb{R}^{n\times n}$ positive definite and $\eta \in (0,1)$, let $d = -B\nabla J(\bar{u})$. If $\nabla J(\bar{u}) \neq 0$, then there exists $\delta > 0$ such that*

$$J(\bar{u} + td) \leq J(\bar{u}) + \eta t \nabla J(\bar{u})^T d \quad \textit{and} \quad \bar{u} + td \in X,$$

for all $t \in [0, \delta)$.

Proof Since X is an open set, there exists $\epsilon > 0$ such that

$$\|u - \bar{u}\| \leq \epsilon \Longrightarrow u \in X. \tag{30}$$

Thus, if we consider $u = \bar{u} + td$, it follows that

$$0 \leq t \leq \frac{\epsilon}{\|d\|} \Longrightarrow \bar{u} + td \in X. \tag{31}$$

Let us denote $\delta_1 = \epsilon/\|d\|$. On the other hand, as J is differentiable and $\eta \in (0,1)$, we have

$$\begin{aligned} \lim_{t\to 0} \frac{J(\bar{u}+td) - J(\bar{u})}{t} &= \nabla J(\bar{u})^T d = -\nabla J(\bar{u})^T B \nabla J(\bar{u}) \\ &< -\eta \nabla J(\bar{u})^T B \nabla J(\bar{u}) \\ &= \eta \nabla J(\bar{u})^T d. \end{aligned}$$

Hence, there exists $\delta_2 > 0$ such that

$$\frac{J(\bar{u}+td) - J(\bar{u})}{t} < \eta \nabla J(\bar{u})^T d,$$

for all $t \in (0, \delta_2)$. Therefore,

$$J(\bar{u} + td) \leq J(\bar{u}) + \eta t \nabla J(\bar{u})^T d, \ \forall t \in [0, \delta_2). \tag{32}$$

Finally, if we take $\delta = \min\{\delta_1, \delta_2\}$, it follows from (31) and (32) that

$$J(\bar{u} + td) \leq J(\bar{u}) + \eta t \nabla J(\bar{u})^T d \quad \text{and} \quad \bar{u} + td \in X, \ \forall t \in [0, \delta),$$

and the proof is complete. □

The lemma below gives a lower bound for the sequence $\{t_k\}$ and will be crucial to establish a lower bound for the functional decrease obtained in consecutive iterations of Algorithm A. Its proof is an adaptation of the proof of Lemma 11.1.1 in [12].

Lemma 2 *Suppose that A1 holds. Then, for all k, we have*

$$t_k \geq \min\left\{1, -\frac{(1-\eta)}{L}\left(-\frac{\nabla J(u_k)^T d_k}{\|d_k\|^2}\right), \frac{\Gamma(u_k)}{2\|d_k\|}\right\}, \tag{33}$$

where, for all $u \in X$,

$$\Gamma(u) = \inf_{w \notin X} \|u - w\|.$$

Proof If $i_k = 0$, then $t_k = 1$ and so (33) holds. Thus, suppose that $i_k > 0$. If $u_k + 2t_k d_k \in X$, then from the definition of i_k we know that $u_k + 2t_k d_k = u_k + (0.5)^{i_k - 1} d_k$ does not satisfy the inequality in (29). Thus

$$J(u_k + 2t_k d_k) > J(u_k) + 2\eta t_k \nabla J(u_k)^T d_k. \tag{34}$$

Since ∇J is L-Lipschitz, it follows that

$$J(u_k + 2t_k d_k) \leq J(u_k) + 2t_k \nabla J(u_k)^T d_k + 2Lt_k^2 \|d_k\|^2. \tag{35}$$

Then, combining (34) and (35) we have

$$\begin{aligned} J(u_k) + 2\eta t_k \nabla J(u_k)^T d_k &< J(u_k) + 2t_k \nabla J(u_k)^T d_k + 2Lt_k^2 \|d_k\|^2 \\ \Longrightarrow 2Lt_k^2 \|d_k\|^2 &> (\eta - 1) 2t_k \nabla J(u_k)^T d_k \\ \Longrightarrow t_k &> -\frac{(1-\eta)}{L}\left(\frac{\nabla J(u_k)^T d_k}{\|d_k\|^2}\right) \end{aligned}$$

and so, (33) also holds.

Finally, if $u_k + 2t_k d_k \notin X$, it follows from the definition of $\Gamma(u_k)$ that

$$2t_k \|d_k\| > \Gamma(u_k).$$

Thus, $t_k > \Gamma(u_k)/2\|d_k\|$, and once again (33) holds. □

Now we are in position to establish a worst-case complexity bound for Algorithm A.

Theorem 1 *Suppose that A1-A3 hold and let* $\{u_k\}$ *be a sequence generated by Algorithm A such that*

$$\Gamma(u_k) > \epsilon \quad \textit{and} \quad \|\nabla J(u_k)\| > \epsilon, \ \textit{for } k = 0, \ldots, T-1, \tag{36}$$

for a given precision $\epsilon > 0$. *Then,* $J(u)$ *is bounded from below by some* J_{low} *and we must have*

$$T \leq \left(\frac{J(u_0) - J_{low}}{\kappa_c}\right)\epsilon^{-2}, \tag{37}$$

where

$$\kappa_c = \min\left\{\eta c_0, \frac{\eta(1-\eta)c_0^2}{Lc_1^2}, \frac{\eta c_0}{2c_1}\right\}. \tag{38}$$

Proof By Step 3 of Algorithm A, we have $J(u_{k+1}) \leq J(\hat{u}_{k+1})$. Thus, combining (29) and and the lower bound for t_k in (33), we obtain the following lower bound for the decrease of the function value in consecutive iterations:

$$\begin{aligned} &J(u_k) - J(u_{k+1}) \geq J(u_k) - J(\hat{u}_{k+1}) \geq \eta t_k \left(-\nabla J(u_k)^T d_k\right) \\ &\geq \eta \min\left\{-\nabla J(u_k)^T d_k, \frac{(1-\eta)}{L}\left(-\frac{\nabla J(u_k)^T d_k}{\|d_k\|}\right)^2, \frac{\Gamma(u_k)}{2}\left(-\frac{\nabla J(u_k)^T d_k}{\|d_k\|}\right)\right\}. \end{aligned} \tag{39}$$

On the other hand, from A3 it follows that

$$\|d_k\| = \| - B_k \nabla J(u_k)\| \leq \|B_k\| \|\nabla J(u_k)\| \leq c_1 \|\nabla J(u_k)\|,$$

and

$$-\nabla J(u_k)^T d_k = \nabla J(u_k)^T B_k \nabla J(u_k) \geq c_0 \|\nabla J(u_k)\|^2. \tag{40}$$

Hence,

$$-\frac{\nabla J(u_k)^T d_k}{\|d_k\|} \geq \frac{c_0 \|\nabla J(u_k)\|^2}{c_1 \|\nabla J(u_k)\|} = \left(\frac{c_0}{c_1}\right) \|\nabla J(u_k)\|. \tag{41}$$

Then, combining (39) with (40), (41) and (36) we obtain

$$\begin{aligned} J(u_k) - J(u_{k+1}) &\geq \eta \min\left\{c_0 \|\nabla J(u_k)\|^2, \frac{(1-\eta)c_0^2}{L c_1^2}\|\nabla J(u_k)\|^2, \left(\frac{c_0}{2c_1}\right)\Gamma(u_k)\|\nabla J(u_k)\|\right\} \\ &\geq \min\left\{\eta c_0, \frac{\eta(1-\eta)c_0^2}{L c_1^2}, \frac{\eta c_0}{2c_1}\right\} \min\left\{\|\nabla J(u_k)\|^2, \Gamma(u_k)\|\nabla J(u_k)\|\right\} \\ &= \kappa_c \min\left\{\|\nabla J(u_k)\|^2, \Gamma(u_k)\|\nabla J(u_k)\|\right\} \\ &> \kappa_c \epsilon^2, \quad \text{for } k = 0, \ldots, T-1. \end{aligned} \tag{42}$$

From A2, it follows that J has a global minimizer on $\mathbb{R}^n$. Thus, there exists J_{low} such that $J(u_k) \geq J_{low}$ for all k. Therefore,

$$J(u_0) - J_{low} \geq J(u_0) - J(u_T) = \sum_{k=0}^{T-1} J(u_k) - J(u_{k+1}) \geq \sum_{k=0}^{T-1} \kappa_c \epsilon^2 = T \kappa_c \epsilon^2$$

$$\Longrightarrow T \leq \left(\frac{J(u_0) - J_{low}}{\kappa_c}\right) \epsilon^{-2},$$

and the proof is complete. □

Remark 3 Theorem 1 means that given $\epsilon > 0$, Algorithm A takes at most $\mathcal{O}(\epsilon^{-2})$ iterations to generate a point $u_T \in X$ such that

$$\Gamma(u_T) \leq \epsilon \quad \text{or} \quad \|\nabla J(u_T)\| \leq \epsilon.$$

For $X = \mathbb{R}^n$, this bound agrees in order with known complexity bounds for first-order methods [2,4,9]. In any case, by (42) we have

$$J(u_T) < J(u_{T-1}) < \ldots < J(u_1) < J(u_0).$$

Finally, from inequality (42) we can establish the following global convergence result.

Theorem 2 *Suppose that A1-A3 hold. Then, given $u_0 \in X$, the sequence $\{u_k\} \subset X$ generated by Algorithm A from u_0 admits a subsequence that converges either to a point in the boundary of X or to a critical point of J in X.*

Proof Let us denote the closure of X by $\bar{X}$. Note that $\{u_k\} \subset L(u_0)$. Thus, by A2, sequence $\{u_k\}$ is bounded and, therefore, it admits a convergent subsequence $\{u_{k_j}\}$, with $u_k \to \bar{u} \in \bar{X}$. Since J is continuous, we also have $J(u_{k_j}) \to J(\bar{u})$ as j goes to infinity. Thus, sequence $\{J(u_k)\}$ is monotonically decreasing and admits a convergent subsequence. Hence, $\{J(u_k)\}$ must be convergent, which implies that

$$\lim_{k\to+\infty} J(u_k) - J(u_{k+1}) = 0.$$

Thus, by applying the Squeeze Theorem on inequality (42) we conclude that

$$\lim_{j\to+\infty} \nabla J(u_{k_j}) = 0 \quad \text{or} \quad \lim_{j\to+\infty} \Gamma(u_{k_j})\|\nabla J(u_{k_j})\| = 0. \tag{43}$$

On the other hand, as ∇J and Γ are continuous functions, we have

$$\lim_{j\to+\infty} \nabla J(u_{k_j}) = \nabla J(\bar{u}) \quad \text{and} \quad \lim_{j\to+\infty} \Gamma(u_{k_j}) = \Gamma(\bar{u}). \tag{44}$$

Then, combining (43) and (44) it follows that

$$\nabla J(\bar{u}) = 0 \quad \text{or} \quad \Gamma(\bar{u}) = 0,$$

that is, the limit point $\bar{u}$ is either a point in the boundary of X or a critical point of J in X. □

2.4 Subspace Multilevel Technique

In the standard coarse-to-fine multilevel strategy, the initial point $u_{l+1,0}$ for level $l+1$ is computed using only the solution u_l^* of the previous level. To allow the finding of a better initial point, we propose the use of all the previous solutions $u_l^*, u_{l-1}^*, \ldots, u_{N_0}^*$ by employing again the subspace technique. Given $N_0 \le z < w \le N$, let us denote by P_z^w the prolongation operator from level z to level w. We set $\hat{u}_{l+1,0} = P_l^{l+1} u_l^*$ and we compute $\nabla J_l(\hat{u}_{l+1,0})$ and $\hat{H}_{l+1,0} \approx \nabla^2 J_l(\hat{u}_{l+1,0})$. Then, we obtain a search direction $\hat{d}_{l+1,0}$ by solving the subspace quadratic problem

$$\min_d \; J_{l+1}(\hat{u}_{l+1,0}) + \nabla J_l(\hat{u}_{l+1,0})^T d + \frac{1}{2} d \hat{H}_{l+1,0} d, \tag{45}$$

$$\text{s. t.} \qquad d \in \mathcal{S}_{l+1,0} \subset \mathbb{R}^{n_{l+1}}, \tag{46}$$

where $\mathcal{S}_{l+1,0} \equiv \text{span}\left(\left\{P_{N_0}^{l+1}u_{N_0}^*, \ldots, P_l^{l+1}u_l^*\right\}\right)$. As in the Two-Step Gauss-Newton method, $\hat{d}_{l+1,0}$ can be easily computed by solving a small-scale linear system. If $J_{l+1}(\hat{u}_{l+1,0} + \hat{d}_{l+1,0}) < J_{l+1}(\hat{u}_{l+1,0})$, we define the initial point for level $l+1$ as $u_{l+1,0} = \hat{u}_{l+1,0} + \hat{d}_{l+1,0}$. Otherwise, we set $u_{l+1,0} = \hat{u}_{l+1,0}$. The corresponding modification in Algorithm 2 can be summarized in the following way.

Algorithm 6 (Subspace Multilevel GN Method)

Step 0 Set $u_{N_0,0} = (0, \ldots, 0) \in \mathbb{R}^{N_0}$ and $l := N_0$ (coarsest level).
Step 1 Compute $u_l^* = GN(l, u_{l,0})$.
Step 2 If $l = N$ (finest level), stop and return u_N^*.
Step 3 Compute $P_l^{l+1}u_l^*, \ldots, P_l^N u_l^*$, set $\hat{u}_{l+1,0} = P_l^{l+1}u_l^*$ and compute $\nabla J_{l+1}(\hat{u}_{l+1,0})$ and $\hat{H}_{l+1,0} \approx \nabla^2 J_{l+1}(\hat{u}_{l+1,0})$.
Step 4 If $l = 0$, set $u_{l+1,0} = \hat{u}_{l+1,0}$ and go to Step 6.
Step 5 Compute $\hat{d}_{l+1,0}$ by solving the subspace problem (45)-(46). If $J_{l+1}(\hat{u}_{l+1,0}+\hat{d}_{l+1,0}) < J_{l+1}(\hat{u}_{l+1,0})$, set $u_{l+1,0} = \hat{u}_{l+1,0}+\hat{d}_{l+1,0}$. Otherwise, set $u_{l+1,0} = \hat{u}_{l+1,0}$.
Step 6 Set $l := l + 1$ and go back to Step 1.

Finally, if at Step 1 of Algorithm 6 we replace Gauss-Newton method by our new Two-Step Gauss-Newton method we obtain the subspace multilevel algorithm below.

Algorithm 7 (Subspace Multilevel Two-Step GN Method)

Step 0 Set $u_{N_0,0} = (0, \ldots, 0) \in \mathbb{R}^{N_0}$ and $l := N_0$ (coarsest level).
Step 1 Compute $u_l^* = 2SGN(l, u_{l,0})$.
Step 2 If $l = N$ (finest level), stop and return u_N^*.
Step 3 Compute $P_l^{l+1}u_l^*, \ldots, P_l^N u_l^*$, set $\hat{u}_{l+1,0} = P_l^{l+1}u_l^*$ and compute $\nabla J_{l+1}(\hat{u}_{l+1,0})$ and $\hat{H}_{l+1,0} \approx \nabla^2 J_{l+1}(\hat{u}_{l+1,0})$.
Step 4 If $l = 0$, set $u_{l+1,0} = \hat{u}_{l+1,0}$ and go to Step 6.
Step 5 Compute $\hat{d}_{l+1,0}$ by solving the subspace problem (45)-(46). If $J_{l+1}(\hat{u}_{l+1,0}+\hat{d}_{l+1,0}) < J_{l+1}(\hat{u}_{l+1,0})$, set $u_{l+1,0} = \hat{u}_{l+1,0}+\hat{d}_{l+1,0}$. Otherwise, set $u_{l+1,0} = \hat{u}_{l+1,0}$.
Step 6 Set $l := l + 1$ and go back to Step 1.

3 Numerical Experiments

In order to investigate the numerical performance of the proposed methods, we have tested implementations of the following algorithms:

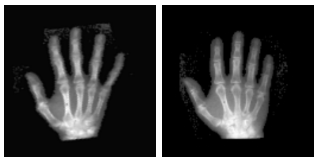
(i) the standard Multilevel Gauss-Newton Algorithm (i.e., Algorithm 2). We shall refer to this code as **GN** (from Gauss-Newton).

(ii) the Multilevel Two-Step Gauss-Newton Algorithm (i.e., Algorithm 5). We shall refer to this code as **TS** (from Two-Step).
(iii) the Subspace Multilevel Gauss-Newton Algorithm (i.e., Algorithm 6). We shall refer to this code as **SIG** (from Subspace Initial Guess).
(iv) the Subspace Multilevel Two-Step Gauss-Newton Algorithm (i.e., Algorithm 7). We shall refer to this code as **HYBRID**, since it can be viewed as a combination of **TS** and **SIG**.

The algorithms were coded in MATLAB (R2017a) language, and the tests were performed on a PC with 3.20 GHz Intel(R) Core(TM) i5-6500 microprocessor, and with installed memory (RAM) of 8.00 GB. In all codes, the execution of the inner optimization algorithm (Gauss-Newton or Two-Step Gauss-Newton) is interrupted when all conditions (15)-(17) are satisfied or when any of the conditions (18) and (19) holds. For the latter conditions, we use $\epsilon = 10^{-16}$ and $k_{max} = 500$. Moreover, in all codes, the Gauss-Newton linear system is solved by the Conjugate Gradient (CG) Method with Diagonal Preconditioner. We stop the execution of the CG method when the residual becomes smaller 10^{-1} or when the maximum of 50 iterations is reached.

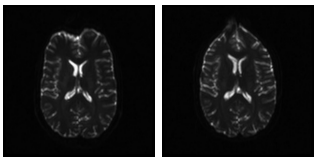
The codes were applied to Image Registration problems corresponding to 20 pairs of images (Reference, Template): ten pairs of medical images (Figures 1-10), and ten pairs of artificial images (Figures 11-20). To evaluate the performance of the codes for several problem sizes, we considered four different resolutions: 128×128, 256×256, 512×512 and 1024×1024. The choice for the objective function (3) was the one corresponding to the hyperelastic model proposed in [1]. Specifically, we use the MATLAB package FAIR as the basis for our tests (see details in [7]). In all codes, the constraint $\det \nabla y > 0$, for $y(x) = x + u(x)$, is handled within the Armijo line-search, that is, to be accepted a trial step must provide a sufficient decrease in the objective and the resulting point must be feasible with respect to the referred constraint (see Algorithm A).

The results reported below summarize more than 21 hours of numerical experimentation. Problems and results for resolution 128×128 are given in Table 1, where "TIME" represents the time in seconds taken by the code to solve the corresponding problem, "IT" represents the number of iterations perfomed to reach the solution, "FE" represents the number of function evaluations performed, and "TOTAL" provides the sum of the values in the corresponding column of the table, where the total time is given in seconds.



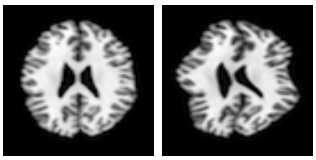
(a) Reference (b) Template

Fig. 1 Problem Hand.



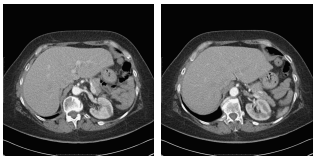
(a) Reference (b) Template

Fig. 2 Problem EPslice.



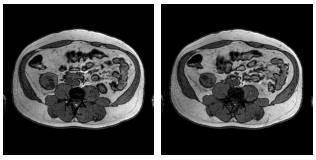
(a) Reference (b) Template

Fig. 3 Problem Brain.



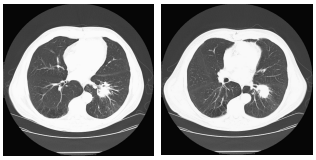
(a) Reference (b) Template

Fig. 4 Problem CT.



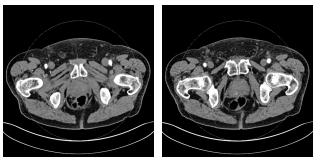
(a) Reference (b) Template

Fig. 5 Problem MRI.



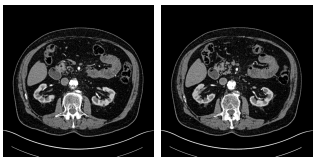
(a) Reference (b) Template

Fig. 6 Problem Lung.



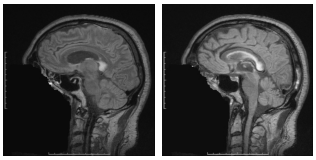
(a) Reference (b) Template

Fig. 7 Problem CT1.



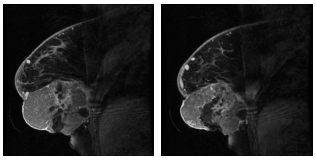
(a) Reference (b) Template

Fig. 8 Problem CT2.



(a) Reference (b) Template

Fig. 9 Problem MRI2.



(a) Reference (b) Template

Fig. 10 Problem Breast.

(a) Reference (b) Template

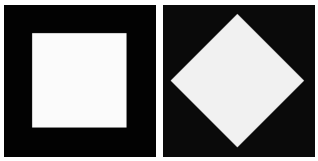
Fig. 11 Problem Circle to C.

(a) Reference (b) Template

Fig. 12 Problem C to Circle.

(a) Reference (b) Template

Fig. 13 Problem A to R.



(a) Reference (b) Template

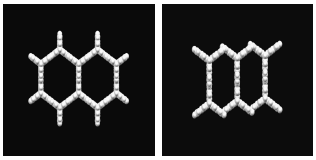
Fig. 14 Problem Square to Square.

(a) Reference (b) Template

Fig. 15 Problem Lena.

(a) Reference (b) Template

Fig. 16 Problem Circle to Square.



(a) Reference (b) Template

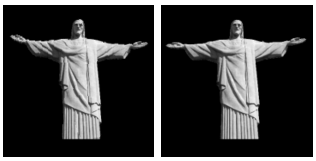
Fig. 17 Problem Molecule.

(a) Reference (b) Template

Fig. 18 Problem F to F.

(a) Reference (b) Template

Fig. 19 Problem Circle to I.



(a) Reference (b) Template

Fig. 20 Problem Rio.

	GN			SIG			TS			HYBRID		
PROBLEM	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**
1. Hand	2.1	31	75	1.8	31	80	2.1	30	120	1.6	27	114
2. EPLslice	11.4	82	216	8.6	71	190	5.0	52	214	4.7	47	201
3. Brain	1.5	52	116	1.5	49	116	1.3	29	114	1.3	29	110
4. CT	5.9	51	116	5.6	48	116	3.9	42	166	3.7	40	162
5. MRI	3.3	56	123	3.3	56	129	2.7	38	152	2.7	38	156
6. Lung	3.3	40	95	3.7	45	112	2.5	30	123	3.5	36	153
7. CT1	8.9	82	202	7.4	79	200	4.7	51	206	3.9	47	194
8. CT2	4.1	36	82	3.1	33	82	2.7	28	112	2.2	25	106
9. MRI2	10.2	95	206	9.4	91	204	5.3	57	228	5.5	58	238
10. Breast	5.1	53	117	5.0	53	123	3.5	37	147	3.7	38	157
11. Circle to C	1.4	45	103	1.6	48	114	1.4	37	131	1.3	37	146
12. C to Circle	2.7	46	119	2.8	49	129	4.5	87	356	2.5	40	175
13. A to R	0.8	39	88	0.7	41	99	0.8	23	94	0.6	19	84
14. Square to Square	0.6	19	46	0.6	19	54	0.5	14	60	0.5	12	61
15. Lena	0.6	18	46	0.7	18	52	0.7	17	68	0.6	16	70
16. Circle to Square	0.2	12	34	0.4	14	44	0.2	11	44	0.3	12	56
17. Molecule	0.9	23	56	1.0	22	60	1.1	22	87	1.1	20	84
18. F to F	0.9	41	95	1.0	43	104	0.8	25	102	0.8	25	106
19. Circle to I	1.5	25	60	1.2	28	73	1.3	22	90	1.3	19	80
20. Rio	0.6	15	40	0.4	14	44	0.5	14	56	0.5	13	58
TOTAL	65.6	861	2035	59.7	852	2125	45.5	666	2670	**42.5**	598	2511

Table 1 Results for resolution 128×128.

	GN			SIG			TS			HYBRID		
PROBLEM	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**
1. Hand	6.4	35	85	7.1	35	92	9.9	36	144	9.2	33	140
2. EPLslice	110.1	172	480	120.6	176	500	88.0	123	548	83.2	120	534
3. Brain	4.6	61	136	4.9	56	134	5.1	35	128	4.9	34	132
4. CT	175.9	201	470	169.8	185	440	79.4	146	577	84.8	146	588
5. MRI	45.4	92	212	37.2	84	200	24.3	63	254	22.4	59	246
6. Lung	29.2	61	155	41.9	74	196	50.2	66	289	31.1	61	264
7. CT1	54.8	115	286	53.8	112	287	35.6	81	330	32.9	73	305
8. CT2	40.5	89	192	38.3	73	168	25.5	60	240	28.2	61	250
9. MRI2	60.2	136	305	60.1	132	304	42.4	97	391	43.7	99	407
10. Breast	36.6	77	176	28.8	75	177	26.8	59	241	20.4	56	233
11. Circle to C	7.0	51	117	6.3	54	130	7.2	42	148	4.9	41	166
12. C to Circle	9.8	53	135	10.9	54	142	10.1	91	374	8.3	44	193
13. A to R	2.5	41	94	3.3	44	109	4.5	28	114	2.9	22	99
14. Square to Square	1.9	21	52	6.6	26	73	3.1	17	72	4.4	16	79
15. Lena	2.2	20	52	2.6	20	60	2.6	19	76	2.8	18	80
16. Circle to Square	2.3	15	42	2.5	19	56	0.4	12	48	0.9	13	62
17. Molecule	5.2	32	76	5.7	27	74	6.1	26	101	5.2	24	101
18. F to F	3.8	46	107	3.7	47	116	3.4	28	116	3.9	28	122
19. Circle to I	6.6	32	76	5.9	32	83	5.5	26	102	5.7	23	96
20. Rio	1.4	17	46	1.7	16	52	1.8	16	64	2.0	15	68
TOTAL	606.5	1367	3294	611.6	1341	3393	432.0	1071	4367	**401.9**	986	4165

Table 2 Results for resolution 256×256.

From Table 1 we see that **TS**, **SIG** and **HYBRID** were better than **GN** in terms of the total time. The fastest code was **HYBRID**, which outperformed **GN** on 16 problems (9 of them corresponding to medical images).

Table 2 shows the results for resolution 256×256. Codes **TS** and **HYBRID** were better than **GN** in terms of the total time. The fastest code was **HYBRID**, which outperformed **GN** on 11 problems (7 of them corresponding to medical images).

	GN			SIG			TS			HYBRID		
PROBLEM	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**
1. Hand	37.6	39	95	32.8	38	102	25.9	38	152	26.3	35	150
2. EPLslice	220.3	190	525	332.3	207	586	286.2	149	666	214.0	137	617
3. Brain	18.0	64	144	15.1	58	142	15.0	37	148	15.7	36	144
4. CT	2827.4	524	1430	1700	370	988	824.3	253	1065	446.2	202	832
5. MRI	303.3	132	309	298.5	117	286	209.3	96	391	286.9	105	436
6. Lung	175.8	77	201	134.3	93	246	425.7	109	500	243.8	88	389
7. CT1	385.1	158	394	346.1	155	396	224.8	110	443	248.2	109	454
8. CT2	1083.4	208	537	1044.4	189	511	480.1	125	533	576.9	137	607
9. MRI2	339.4	167	388	270.0	157	371	204.3	125	507	249.0	130	545
10. Breast	221.2	100	234	230.6	98	239	195.5	85	347	151.1	77	319
11. Circle to C	31.2	54	125	57.2	68	162	60.9	51	172	58.9	51	197
12. C to Circle	37.9	56	144	39.9	57	150	47.3	97	401	47.4	49	210
13. A to R	26.2	49	112	13.3	46	117	17.8	30	121	14.5	24	111
14. Square to Square	16.7	25	62	17.9	27	78	26.3	20	84	20.7	18	86
15. Lena	17.2	22	58	19.1	22	68	18.3	21	84	19.7	20	90
16. Circle to Square	11.4	20	54	13.3	23	66	2.6	13	52	3.7	14	66
17. Molecule	74.3	40	95	69.7	46	116	74.5	39	145	64.7	37	150
18. F to F	12.2	48	113	7.1	48	122	6.6	29	122	7.6	29	128
19. Circle to I	32.4	43	100	53.1	42	105	45.3	33	128	64.3	31	125
20. Rio	9.5	19	52	9.6	18	60	11.8	18	72	12.5	17	78
TOTAL	5880.7	2035	5172	4704.6	1879	4911	3202.6	1478	6133	**2772.3**	1346	5734

Table 3 Results for resolution 512×512.

	GN			SIG			TS			HYBRID		
PROBLEM	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**
1. Hand	119.9	41	101	121.2	40	110	105.3	40	160	58.2	36	156
2. EPLslice	539.9	198	550	589.5	216	612	518.2	155	693	654.2	150	677
3. Brain	49.8	65	148	91.7	63	156	52.9	38	152	46.7	37	152
4. CT	3884.3	556	1504	2299.4	390	1035	2478.2	297	1257	1526.3	235	965
5. MRI	1048.2	166	384	1094.8	136	337	908.2	120	486	696.0	118	489
6. Lung	929.0	102	258	1074.6	125	318	1615.3	140	650	2520.2	144	653
7. CT1	1246.3	181	446	1040.6	174	440	906.2	133	532	1050.3	143	585
8. CT2	6148.7	330	899	4924.8	287	796	3758.3	198	881	3865.9	215	982
9. MRI2	974.9	189	442	809.6	170	409	998.3	152	621	1088	154	652
10. Breast	515.3	107	254	477.5	105	259	760.1	100	414	1120.6	104	440
11. Circle to C	235.5	61	141	538.6	93	214	126.1	53	178	192.4	56	210
12. C to Circle	245.3	61	156	239.8	65	168	181.1	100	414	204.9	54	226
13. A to R	124.2	58	132	117.9	51	129	89.2	34	132	105.3	29	123
14. Square to Square	57.8	26	66	106.7	29	84	68.0	21	88	64.2	19	90
15. Lena	140.3	26	68	109.3	25	78	135.8	24	96	92.2	22	100
16. Circle to Square	69.6	23	62	57.1	27	76	43.4	14	56	34.1	15	70
17. Molecule	364.3	47	111	374.9	53	132	830.6	69	242	662.0	59	223
18. F to F	60.4	51	121	63.2	51	132	70.5	31	130	81.3	32	142
19. Circle to I	73.2	44	104	139.8	44	111	181.7	39	146	234.1	38	149
20. Rio	90.9	22	60	69.2	20	68	70.6	20	80	78.9	19	88
TOTAL	16917.9	2354	6007	14340	2164	5664	**13898**	1778	7408	14376.6	1679	7172

Table 4 Results for resolution 1024×1024.

Table 3 shows the results for resolution 512×512. Codes **TS**, **SIG** and **HYBRID** were better than **GN** in terms of the total time. In this case, the fastest code was **HYBRID**, which outperformed **GN** on 13 problems (9 of them corresponding to medical images).

Finally, Table 4 shows the results for resolution 1024×1024. Once again, **TS**, **SIG** and **HYBRID** were better than **GN** in terms of the total time. The fastest code was **TS**, which outperformed **GN** on 12 problems (6 of them corresponding to medical images).

The improved performance of **TS** and **HYBRID** over **GN** is better highlighted in Tables 5, 6 and 7, which shows the reduction in the total time provided by the new methods.

Resolution	Time GN	Time SIG	Reduction
128×128	65.6	59.7	9.0%
256×256	606.5	611.6	–
512×512	5880.7	4704.6	20.0%
1024×1024	16917.9	14340	15.2%

Table 5 Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and SIG.

Resolution	Time GN	Time TS	Reduction
128×128	65.6	45.5	30.6%
256×256	606.5	432.0	28.7%
512×512	5880.7	3202.6	45.5%
1024×1024	16917.9	13898	**17.8**%

Table 6 Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and TS.

Resolution	Time GN	Time HYBRID	Reduction
128×128	65.6	42.5	**35.2**%
256×256	606.5	401.9	**33.7**%
512×512	5880.7	2772.3	**52.8%**
1024×1024	16917.9	13898	15.0%

Table 7 Comparison of the total time (in seconds) to solve all 20 problems for each resolution between GN and HYBRID.

As mentioned above, codes **TS** and **HYBRID** behave much better when we consider only medical images. In terms of the total time, this difference of performance is shown on Tables 8, 9 and 10.

Resolution	Time GN	Time SIG	Reduction
128×128	55.7	49.4	11.3%
256×256	563.6	562.6	0.2%
512×512	5611.6	4404.5	21.5%
1024×1024	15456	12524	18.9%

Table 8 Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and SIG.

Resolution	Time GN	Time TS	Reduction
128×128	55.7	33.7	39.4%
256×256	563.6	387.1	31.3%
512×512	5611.6	2891.1	48.4%
1024×1024	15456	12100.9	**21.7**%

Table 9 Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and TS.

Resolution	Time GN	Time HYBRID	Reduction
128×128	55.7	32.9	**41.0**%
256×256	563.6	360.8	**36.0**%
512×512	5611.6	2458.3	**56.2%**
1024×1024	15456	12100.9	18.3%

Table 10 Comparison of the total time (in seconds) to solve all 10 problems with medical images for each resolution between GN and HYBRID.

Additional information about the codes can be obtained by using the Performance Profile, which is a tool for benchmarking and comparing optimization software [3]. More specifically, let $t_{p,s}$ denote the time to solve problem p by solver s. The performance ratio is defined as $r_{p,s} = \dfrac{t_{p,s}}{t_p^*}$, where t_p^* is the lowest time required to solve problem p among all solvers that are being compared. Clearly, $r_{p,s} \geq 1$ for all p and s. The performance profile for each code s is defined as

$$\rho_s(\tau) = \frac{\text{number of problems for which } r_{p,s} \leq \tau}{\text{total number of problems}}.$$

Therefore, the value $\rho_s(\tau)$ represents the percentage of problems solved by algorithm s with a cost at most τ times worse than that of the best algorithm. This means that, for a given value of τ, the best solver is the one with the highest value of $\rho_s(\tau)$. In particular, $\rho_s(1)$ gives the percentage of problems for which solver s is the best.

Figures 21, 22 e 23 show the performance profiles for codes **GN** and **HYBRID** taking as reference all 30 problems with medical images and resolutions of 128×128, 256×256 and 512×512 (combined results of Tables 1, 2 e 3). As expected, we can see that in this set of test problems, code **HYBRID** is significantly more efficient than **GN** in terms of CPU time and number of Iterations. It is interesting to notice that **GN** outperforms **HYBRID** in terms of the number of function evaluations. However, this effect is compensated by the time that **HYBRID** saves in the solution of a smaller number of Gauss-Newton linear systems (which is equal to the number of iterations).

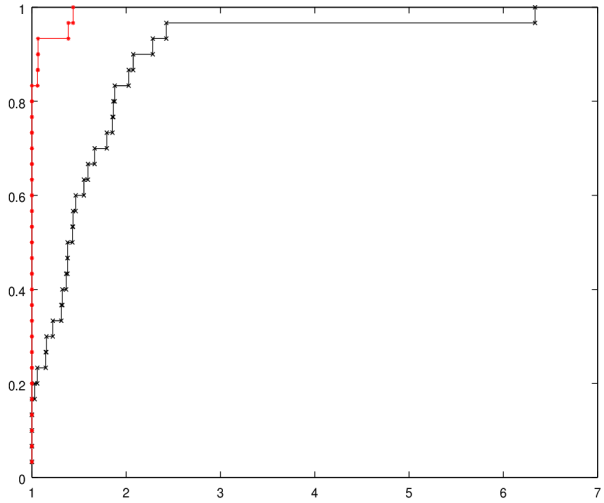


Fig. 21 Performance Profile based on CPU Time for the set of 20 problems with medical images and resolutions of 128×128 and 256×256.

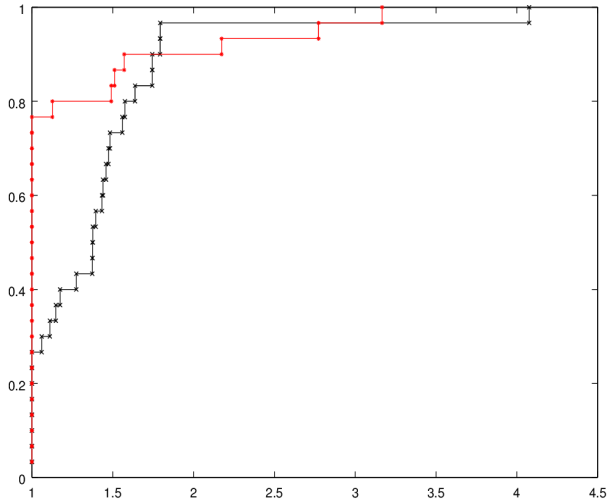


Fig. 22 Performance Profile based on Number of Iterations for the set of 20 problems with medical images and resolutions of 128×128, 256×256 and 512×512.

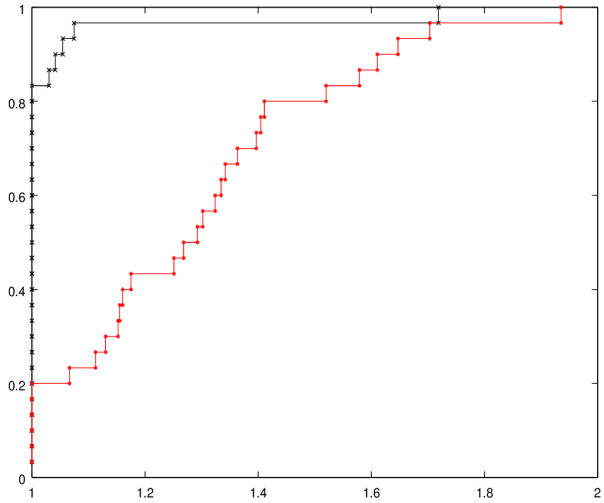


Fig. 23 Performance Profile based on Number of Function Evaluations for the set of 20 problems with medical images and resolutions of 128×128, 256×256 and 512×512.

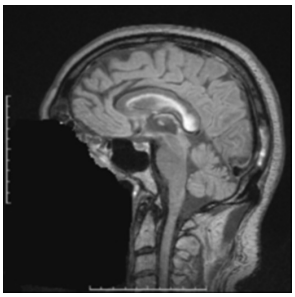
As an example, Figure 24 shows the registered images obtained by all codes applyed to problem MRI2 with resolution 512×512.

We also tested the codes **GN** and **HYBRID** on four 3D problems from [7] (such as the Brain Problem illustrated on Figures 25 and 26). The results are in Table 11.

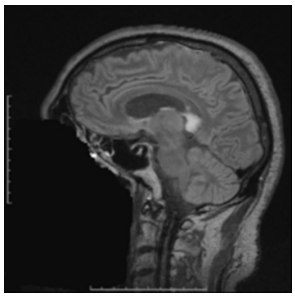
	GN			**HYBRID**		
PROBLEM	**TIME**	**IT**	**FE**	**TIME**	**IT**	**FE**
1. Brain	1435	26	60	1249	25	97
2. Knee	937	16	40	698	13	56
3. Phantom	105	15	37	243	16	68
4. Mice	62	28	65	48	17	68
TOTAL	2539	85	202	2238	71	289

Table 11 Results for 3D problems.

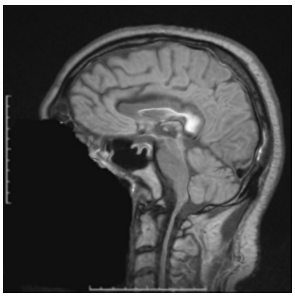
Once again, **HYBRID** outperformed **GN**. However, it seems that the gain of **HYBRID** over **GN** deteriorates when the problems become larger. One possible explanation is that for larger problems, the computational cost to compute function and grandient evaluations becomes comparable with the cost to solve the Gauss-Newton problem. In this case, the saving obtained by performing a smaller number of iterations may be not enough to compensate the additional time used to evaluate the objective function and its gradients.



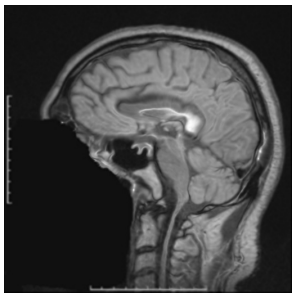
(a) Template Image.



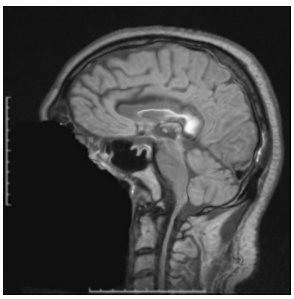
(b) Reference Image.



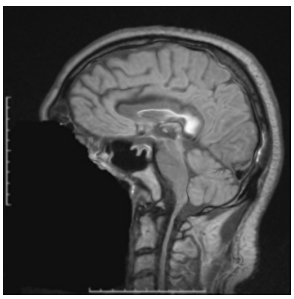
(c) $T(\mathbf{y})$ obtained by code **GN** in 468.11 seconds.



(d) $T(\mathbf{y})$ obtained by code **SIG** in 326.04 seconds.



(e) $T(\mathbf{y})$ obtained by code **TS** in 250.82 seconds.



(f) $T(\mathbf{y})$ obtained by code **HYBRID** in 314.29 seconds.

Fig. 24 Registered images for problem MRI2 with resolution 512×512.

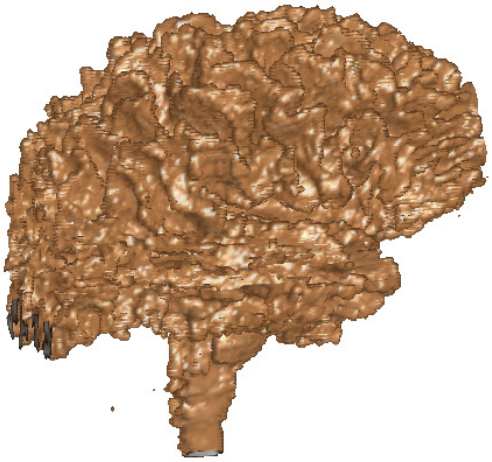
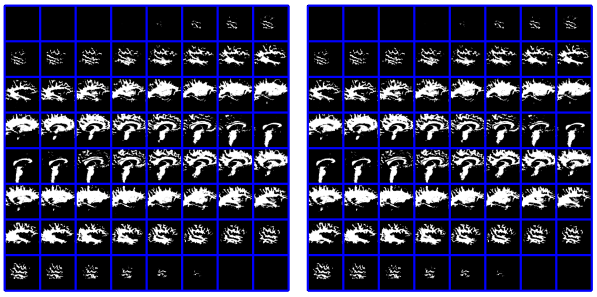


Fig. 25 3D Brain Problem.



(a) Template Image. (b) Reference Image.

Fig. 26 Template and Reference for the 3D Brain Problem.

Finally, it is worth to mention that the methods proposed in this work can be applied to general smooth optimization problems. Notice that the key component of the codes **HYBRID** and **TS** is the Algorithm 3 embedded on them. To evaluate the performance of this algorithm on a different class of problems, we applied it on a set of 10 test problems from [8] (without the multilevel step). The results on Table 12 show that the gain obtained with Algorithm 3 over the standard Gauss-Newton method is not restricted to Image Registration Problems.

	Gauss-Newton		Algorithm 3	
PROBLEM(n,m)	**IT**	**FE**	**IT**	**FE**
1. Extended Rosenbrock(100,100)	50	180	32	129
2. Extended Rosenbrock(500,500)	49	177	27	110
3. Extended Powell Singular(100,100)	23	47	26	79
4. Extended Powell Singular(500,500)	24	49	26	79
5. Penalty I(100,101)	99	667	25	154
6. Penalty I(500,501)	101	670	27	154
7. Variably Dimensioned(100,100)	48	93	28	81
8. Discrete Integral Equation(100,100)	15	31	02	06
9. Broyden Tridiagonal(100,100)	20	41	11	32
10. Broyden Banded(100,100)	24	49	15	45
TOTAL	453	2004	219	869

Table 12 Results for MGH problems.

4 Conclusion

In this paper we propose a Two-Step Gauss-Newton method for smooth unconstrained optimization and a modified coarse-to-fine multilevel scheme. Both methods rely on very simple subspace techniques and they aim the solution of Image Registration problems by the discretize-then-optimize approach. Numerical experiments were performed on a diverse set of 20 pairs of images (Reference, Template) considering four different resolutions. The results obtained correspond to more than 21 hours of numerical experimentation. For registration problems with resolutions of 128×128, 256×256 and 512×512, a hybrid of our two new subspace methods outperformed the standard multilevel Gauss-Newton method, reducing the total running time in 52.8% for problems with resolution of 512×512. The advantage of the new methods over the Gauss-Newton scheme is even bigger when we consider the registration of medical images. For example, in our set of 10 problems from medical images with resolution of 512×512, our hybrid method was faster than the multilevel Gauss-Newton method on 9 problems, reducing the total running time in 56.2%. These results are very encouraging. As a future work, we intend to investigate other choices for the subspace used in the Two-Step Gauss-Newton method.

References

1. Burger, M., Modersitzki, J., Ruthotto, L.: A Hyperelastic Regularization Energy for Image Registration. SIAM Journal on Scientific Computing **35**, B132-B148 (2013)
2. Cartis, C., Gould, N.I.M., Toint, Ph.L.: On the Complexity of Steepest Descent, Newton's and Regularized Newton's Methods for Nonconvex Unconstrained Optimization Problems. SIAM Journal on Optimization **20**: 2833-2852 (2010)
3. Dolan, E., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201-213 (2002)
4. Grapiglia, G.N., Yuan, J., Yuan, Y.: Nonlinear Stepsize Control Algorithms: Complexity Bounds for First-and Second-Order Optimality. Journal of Optimization Theory and Applications **171**: 980-997 (2016)

5. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. Mathematical Programming **45**, 503-528 (1989)
6. Modersitzki, J.: Numerical Methods for Image Registration. Oxford University Press, New York, 2014
7. Modersitzki, J.: FAIR: Flexible Algorithms for Image Registration. SIAM, Philadelphia, 2009
8. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing Unconstrained Optimization Software. ACM Transactions on Mathematical Software **7**, 17-41 (1981)
9. Nesterov, Yu.: Introductory Lectures on Convex Optimization: A Basic Course. Applied Optimization. Kluwer, Dordrecht (2004)
10. Nocedal, J.: Updating quasi-Newton matrices with limited storage. Mathematics of Computation **35**, 773-782 (1980)
11. Oliveira, F.P.M., Tavares, J.M.R.S.: Medical image registration: a review. Computer Methods in Biomechanics and Biomedical Engineering **17**, 73-93 (2014)
12. Sun, W., Yuan, Y.: Optimization Theory and Methods: Nonlinear Programming. Springer, Berlin (2006)
13. Yuan, Y.: A review on subspace methods for nonlinear optimization. In: Proceedings of International Congress of Mathematicians, Seoul, 2014
14. Zhang, J., Chen, K.: A new curvature-based image registration model and its fast algorithm. International Journal of Numerical Analysis and Modeling **13**, 969–985 (2016)