SoftwareX 24 (2023) 101524

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

ImWIP: Open-source image warping toolbox with adjoints and derivatives

Jens Renders [a,c,*], Ben Jeurissen [a,b], Anh-Tuan Nguyen [a,c], Jan De Beenhouwer [a,c], Jan Sijbers [a,c]

[a] imec-Vision Lab, Department of Physics, University of Antwerp, Belgium
[b] Lab for Equilibrium Investigations and Aerospace, Department of Physics, University of Antwerp, Belgium
[c] DynXlab: Center for 4D Quantitative X-ray Imaging and Analysis, Antwerp, Belgium

ARTICLE INFO

Article history:
Received 3 February 2023
Received in revised form 3 August 2023
Accepted 6 September 2023

Keywords:
Image warping
Inverse problems
Image reconstruction

ABSTRACT

Image warping is a popular tool for modeling deformation and motion in digital images. Inversion of such models requires two related operators: adjoint and differentiated image warping. Many applications rely on these operators, often through ad hoc and approximate implementations, which leads to a suboptimal quality and convergence speed, and hinders development of and comparison across different applications.

In this work, we present an open-source image warping toolbox called ImWIP (Image Warping for Inverse Problems) that overcomes these issues. It implements differentiable image warping operators, together with their exact adjoints and derivatives (up to floating point errors). ImWIP is demonstrated on examples from X-ray computed tomography and magnetic resonance imaging, and is shown to improve both reconstruction quality and convergence speed compared to state-of-the-art warping methods.

Code metadata

Current code version	v1.2
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00085
Code Ocean compute capsule	
Legal Code License	GNU GPL v3.0
Code versioning system used	git
Software code languages, tools, and services used	Python, Numba, C++, CUDA
Compilation requirements, operating environments & dependencies	CUDA GPU, Python, NumPy, Scipy, Numba, Pylops, tqdm, Cython
If available Link to developer documentation/manual	https://imwip.readthedocs.io/
Support email for questions	jens.renders@uantwerpen.be

1. Motivation and significance

Image warping is a transformation that maps the positions in a digital image to new positions, thereby changing the geometric properties of the image rather than its pixel values [1–3]. The change in the position of pixels is dictated by a Deformation Vector Field (DVF), which is often a function of a few parameters, such as an affine or rigid transformation. Various image processing packages provide image warping functionalities, but lack the necessary tools for inversion. These tools include the adjoint action of image warping, to allow reconstruction of an image from a model that includes an image warping operator, and the derivative of image warping, to allow reconstruction of the DVF using gradient based techniques.

Image warping operators are used to model motion in various image reconstruction problems [4–9]. In these applications, adjoint image warping is used to solve for the unknown, unwarped image. It can be approximated by applying a regular image warp along a negative DVF, which is fast but inaccurate for large magnitude DVFs, or by a warp along an approximate inverse of the DVF, which is more accurate but computationally more expensive.

* Corresponding author at: imec-Vision Lab, Department of Physics, University of Antwerp, Belgium.
E-mail address: jens.renders@uantwerpen.be (Jens Renders).

https://doi.org/10.1016/j.softx.2023.101524

Applications where an inverse problem is solved for a DVF include general optical flow [10–12] and other registration problems [13,14] including rigid/affine motion estimation [15,16]. In this case, the derivative of the image warping operator towards the DVF is needed, which is usually implemented using finite difference approximations, neglecting the resampling method used during image warping. There are also applications in which both the image and the motion or distortion are unknown, and have to be reconstructed simultaneously [17–21], using both adjoint and differentiated image warping in the reconstruction.

To our knowledge, neither adjoint nor differentiated image warping operators are implemented in current image processing toolboxes. This leads to ad hoc implementations in each application, which hinder the development of new algorithms and comparisons across different fields. Moreover, the current implementations are often based on approximations. Few papers investigate how these approximations compare to the exact implementation (up to floating point errors) of the corresponding operation. In [22], it is shown that exact differentiated image warping can significantly improve 2D optical flow algorithms, yet no algorithm is provided and the work has not been extended to 3D or other applications. In [23,24], 3D image warping is applied in a deep learning framework, but is restricted to linear or nearest neighbor interpolation, for which exact derivatives do not exist.

In this work, we present ImWIP (Image Warping for Inverse Problems) [25], an open-source toolbox that provides parallelized, matrix-free implementations of approximate and exact algorithms for adjoint and differentiated image warping. ImWIP enables the modular design of algorithms that solve for unknown images or DVFs in the context of motion/deformation in 2D and 3D. The toolbox is demonstrated on examples from X-ray computed tomography and magnetic resonance imaging.

2. Software description

Image warping using multivariate splines (usually linear or cubic) is a linear operator $\mathbb{R}^N \to \mathbb{R}^N$ in terms of the image $\boldsymbol{x} \in \mathbb{R}^N$. Therefore, image warping can be represented as a matrix $\boldsymbol{M} \in \mathbb{R}^{N\times N}$, such that the warped image is given by $\boldsymbol{Mx}$. The matrix representation can be generated by ImWIP, which allows the use of linear algebra to solve certain inverse problems. However, for large 3D images, the matrix representation is inconvenient to store even though it is sparse, as the number of non-zero elements is up to 64 times the number of voxels in the warped images. Alternatively, its (adjoint) multiplication with a vector can be computed efficiently by computing the coefficients on the fly and in parallel. The coefficients of one row (column in the adjoint case) are equivalent to 64 polynomial evaluations that express the dependency of the interpolated value on a $4 \times 4 \times 4$ neighborhood.

By taking into account that the polynomials, and by extension the matrix $\boldsymbol{M} = \boldsymbol{M}(\boldsymbol{v})$, are a function of the DVF $\boldsymbol{v}$, differentiated image warps can be computed as the matrix–vector multiplication of derivatives of $\boldsymbol{M}(\boldsymbol{v})$. With differentiated image warping, we refer to the derivative of a warped image: $\frac{d}{d\boldsymbol{v}}[\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f}]$, where $\boldsymbol{f} \in \mathbb{R}^N$ and $\boldsymbol{v} = (\boldsymbol{v}_x, \boldsymbol{v}_y, \boldsymbol{v}_z) \in \mathbb{R}^{N\times 3}$. This derivative is a linear operator from $\mathbb{R}^N$ to $\mathbb{R}^{N\times 3}$. More details on the implementation of these operations, as well as commonly used approximations, can be found in Appendix.

2.1. Software architecture

The Python interface of ImWIP contains four submodules:

- `imwip.functions`: Low level access to image warping, adjoint image warping and differentiated image warping in a functional style. Each function is implemented as a GPU-accelerated matrix–vector multiplication, where the matrix coefficients are never stored, to reduce the memory cost. The main functions are `imwip.functions.warp()` and `imwip.functions.affine_warp()`. They provide similar functionality as the warping functions in scikit-image [26], Scipy [27] and OpenCV [28]:
  - `skimage.transform.warp()`
  - `scipy.ndimage.map_coordinates()`
  - `scipy.ndimage.affine_transform()`
  - `cv2.remap()`

  with the important difference that ImWIP also implements the corresponding adjoints and derivatives.
- `imwip.operators`: High level access to image warping, adjoint image warping and differentiated image warping through the use of the `LinearOperator` class of SciPy. This allows expressions in terms of matrices, while internally calling the functions from `imwip.functions` and avoiding storing matrices and the associated memory cost.
- `imwip.matrices`: If explicit access to the matrix coefficients is required, the (sparse) matrices can be constructed using this submodule. In most cases, it is more efficient to use `imwip.operators`.
- `imwip.solvers`: A small collection of solvers, applicable to many inverse problems involving image warping.

This Python interface provides access to two backends. In these backends, image warping, adjoint image warping and differentiated image warping are implemented using different toolchains:

- The C++ CUDA back-end: this is the fastest option and is accessible from any language that has a C foreign function interface. On Linux Systems where the `nvcc` compiler is available, this back-end will be automatically compiled upon installation of the Python package, and will be used as the default back-end. This backend only accepts NumPy arrays, which will be automatically copied to the GPU for computation, and then back to the CPU.
- the Numba back-end: this is a pure Python implementation that is JIT compiled for CUDA targets using Numba. It makes ImWIP portable and easy to install, and provides more readable code than the C++ back-end. On operating systems other than Linux, or systems where `nvcc` is not available, this back-end will be selected as default. This backend accepts NumPy arrays, but also any array type that supports the CUDA Array Interface of Numba. This includes CuPy and PyTorch arrays, and avoids copies to and from the host memory.

2.2. Software functionalities

ImWIP provides the necessary operators to solve a (possibly non-linear) system of equations for unknown images and DVFs. A general inverse problem that combines these functionalities is given by

$$\boldsymbol{AM}(\boldsymbol{v})\boldsymbol{x} = \boldsymbol{b} \quad , \tag{1}$$

where

1. $\boldsymbol{x} \in \mathbb{R}^N$ is a D dimensional image with N pixels,
2. $\boldsymbol{v} \in \mathbb{R}^{N\times D}$ is a DVF,
3. $\boldsymbol{M}(\boldsymbol{v}) : \mathbb{R}^N \to \mathbb{R}^N$ is the image warping operator for DVF $\boldsymbol{v}$,

4. $\boldsymbol{A} : \mathbb{R}^N \rightarrow \mathbb{R}^M$ is a linear operator that models the imaging modality, with M measurements,
5. $\boldsymbol{b} \in \mathbb{R}^M$ is the measured data.

Such an inverse problem appears in many contexts, where the object deforms or moves during imaging [4–9,19,20,29]. The inverse problem can be solved for the image $\boldsymbol{x}$, for the DVF $\boldsymbol{v}$, or for both, by minimizing the following objective function with respect to the desired variable:

$$\text{Loss}(\boldsymbol{x}, \boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{A}\boldsymbol{M}(\boldsymbol{v})\boldsymbol{x} - \boldsymbol{b}\|^2 \quad . \tag{2}$$

This loss function can be efficiently minimized with modern gradient-based solvers such as [30,31], if the gradient with respect to both variables can be computed. These gradients can be expressed as

$$\nabla_{\boldsymbol{x}} \text{Loss}(\boldsymbol{x}, \boldsymbol{v}) = \boldsymbol{M}^T(\boldsymbol{v})\boldsymbol{A}^T[\boldsymbol{A}\boldsymbol{M}(\boldsymbol{v})\boldsymbol{x} - \boldsymbol{b}] \quad , \tag{3}$$

$$\nabla_{\boldsymbol{v}} \text{Loss}(\boldsymbol{x}, \boldsymbol{v}) = \left[\frac{d}{d\boldsymbol{v}}\boldsymbol{M}(\boldsymbol{v})\boldsymbol{x}\right]^T \boldsymbol{A}^T[\boldsymbol{A}\boldsymbol{M}(\boldsymbol{v})\boldsymbol{x} - \boldsymbol{b}] \quad . \tag{4}$$

This example shows how the adjoint image warping operator $\boldsymbol{M}^T$ appears when solving for an unknown warped image, and how differentiated image warping $\frac{d}{d\boldsymbol{v}}[\boldsymbol{M}(\boldsymbol{v})\boldsymbol{x}]$ appears when solving for an unknown DVF that is used to warp an image. Using the `imwip.operators` module, the loss function and its gradient can be efficiently implemented by using the mathematical expressions presented in Eq. (3) and Eq. (4). Next to this example, there are many objective functions in other inverse problems, or other optimization tasks such as deep learning, that contain warped images. ImWIP facilitates the development of modular algorithms for all of these tasks.

3. Illustrative examples

The documentation of ImWIP provides many examples, ranging from simple educational examples to complex real world applications. Here, we highlight two applications: one that shows the advantage of exact differentiated image warping compared to the commonly used finite difference approximation, and one that demonstrates the combination of adjoint and differentiated image warping.

3.1. Differentiated image warping: distortion correction in MRI

Context: Echo-planar imaging (EPI) is the fastest imaging method in MRI, making it a popular choice for demanding MRI applications such as diffusion and functional MRI. Unfortunately, EPI is sensitive to inhomogeneities of the MRI scanner's main magnetic field, causing significant distortions along the phase encoding direction. A popular approach to estimate these field inhomogeneities and to correct the corresponding distortions is to acquire a pair of EPI images with opposing phase-encode directions, which leads to images with distortions along the phase encoding direction of equal magnitude but in opposing directions. The distortion can then be corrected from such a pair by trying to find the field that when applied to both images in the pair produces two corrected images that are as similar as possible [13].

Acquisition: T2-weighted MRI images were acquired on a GE Discovery MR750 3T MRI system equipped with an 8-channel receiver head coil using a spin-echo EPI sequence (dataset 2 in [32]). Two $b = 0\,\text{s/mm}^2$ EPI images were acquired with opposing phase-encode directions (anterior–posterior and posterior–anterior), specifically for the purpose of EPI distortion correction. Other imaging parameters were: TR/TE: 9500/100 ms, voxel size: $2 \times 2 \times 2\ \text{mm}^3$, matrix: 120×120, slices: 68, and NEX: 1.

Objective function and gradient: Given two 3D EPI images $\boldsymbol{f}, \boldsymbol{g} \in \mathbb{R}^N$, the goal is to find a DVF $\boldsymbol{v} = (\boldsymbol{0}, \boldsymbol{v}_y, \boldsymbol{0}) \in \mathbb{R}^{N\times 3}$ such that

$$\text{diag}(\boldsymbol{1} + \nabla_y \boldsymbol{v}_y)\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f} = \text{diag}(\boldsymbol{1} - \nabla_y \boldsymbol{v}_y)\boldsymbol{M}(-\boldsymbol{v})\boldsymbol{g} \quad , \tag{5}$$

where ∇_y is an image gradient in the y direction. The factors $\text{diag}(\boldsymbol{1} + \nabla_y \boldsymbol{v}_y)$ and $\text{diag}(\boldsymbol{1} - \nabla_y \boldsymbol{v}_y)$ model the accumulation of intensity where the image is compressed, and the dilution of intensity where the image is spread out [13]. Eq. (5) can be solved using a least-squares functional with a regularization term that penalizes variation in the DVF:

$$\text{Loss}_{\text{EPI}}(\boldsymbol{v}_{\boldsymbol{y}}) = \frac{1}{2}\|\boldsymbol{r}\|^2 + \frac{a}{2}\|\nabla \boldsymbol{v}_y\|^2 \quad , \tag{6}$$

where $\boldsymbol{r}$ is the residue:

$$\boldsymbol{r} = \text{diag}(\boldsymbol{1} + \nabla_y \boldsymbol{v}_y)\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f} - \text{diag}(\boldsymbol{1} - \nabla_y \boldsymbol{v}_y)\boldsymbol{M}(-\boldsymbol{v})\boldsymbol{g} \quad . \tag{7}$$

The regularization parameter $a \in \mathbb{R}$ was empirically chosen as $a = 10^6$. The gradient of this loss function with respect to $\boldsymbol{v}_y$ is:

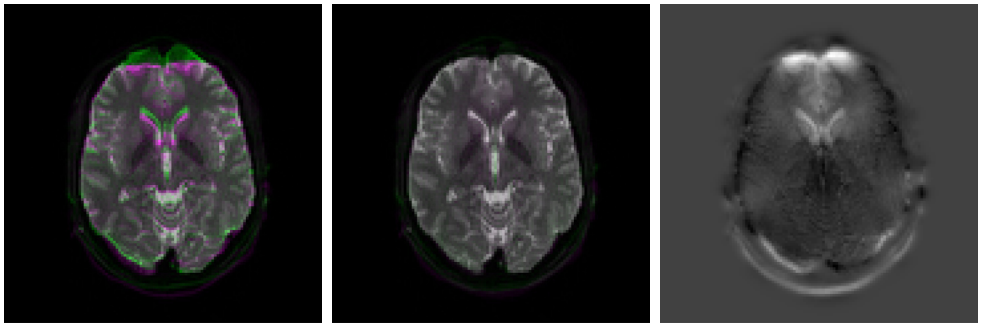
$$\begin{aligned} \nabla \text{Loss}_{\text{EPI}}(\boldsymbol{v}_{\boldsymbol{y}}) =& \nabla_y^T \text{diag}[\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f}]\boldsymbol{r} \\ &+ \text{diag}(\boldsymbol{1} + \nabla_y \boldsymbol{v}_y)\left[\frac{d}{d\boldsymbol{v}_y}\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f}\right]^T \boldsymbol{r} \\ &+ \nabla_y^T \text{diag}[\boldsymbol{M}(-\boldsymbol{v})\boldsymbol{g}]\boldsymbol{r} \\ &+ \text{diag}(\boldsymbol{1} - \nabla_y \boldsymbol{v}_y)\left[\frac{d}{d\boldsymbol{v}_y}\boldsymbol{M}(-\boldsymbol{v})\boldsymbol{g}\right]^T \boldsymbol{r} \\ &+ a\nabla_y^T \nabla_y \boldsymbol{v}_{\boldsymbol{y}} \quad . \end{aligned} \tag{8}$$

Results: In Fig. 1(a), a pair of EPI images, recorded with anterior–posterior and posterior–anterior PE directions, are visualized in green and magenta, respectively. By overlaying the images on top of each other, they add up to a grayscale image in areas where they are equal, while areas with differences are highlighted in green and magenta. After applying gradient descent with the Barzilai–Borwein step size for 600 iterations with approximate differentiated image warping, there were no visible improvements in the resulting images when iterating further. Using exact differentiated image warping, the same least-squares fit was obtained in only 200 iterations, as can be observed on the convergence plot in Fig. 3. The resulting images are shown in Fig. 1(b). The absence of green and magenta regions indicates that the distortions have been removed. Fig. 1(c) shows the y component of the DVF that corrects the deformation. Intense regions signify large distortions, which correspond to regions of transition between tissues with different susceptibility, such as the sinuses. In Fig. 2, the difference between the deformation corrected images using approximate (a) and exact (b) differentiated image warping is visualized. It can be observed that using exact differentiated image warping led to a better fit, especially around the sinuses.

3.2. Combining adjoint and differentiated image warping: Dynamic computed tomography

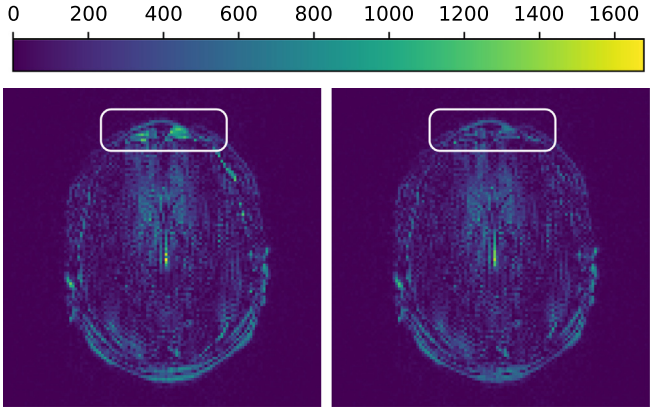
Context: X-ray computed tomography is a non-destructive technique for the 3D imaging of the interior of objects. Multiple X-ray projections of the object are acquired, along different angles. Reconstructing the image from these projections can be formulated as an inverse problem. Standard CT reconstruction techniques assume that the object is stationary during acquisition. To accommodate for possible motion or deformation during the acquisition, the static model can be extended using image warping operators [8,9,21,33].

Simulation: A dynamic CT scan was simulated using a 2D phantom representing the central slice of a bone scaffold on a



(a) Input with deformations (b) Output with deformations corrected (c) Estimated DVF using exact derivatives

Fig. 1. EPI images before (a) and after (b) deformation correction using exact derivatives. The DVF describing the deformation in the y direction is shown in (c).



(a) Approximate derivatives (b) Exact derivatives

Fig. 2. Difference images after deformation correction, using approximate derivatives (a) and exact derivatives (b). Darker colors correspond to larger differences. A region with large differences is marked with a white rectangle.

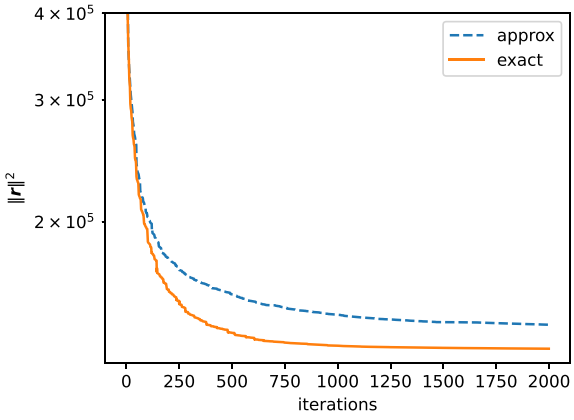


Fig. 3. Convergence plot of the MRI experiment. Exact differentiated image warping leads to a faster drop in residue, and a lower stagnation point.

512 × 512 pixel grid. A total of 105 parallel beam projections were simulated, along angles uniformly distributed in the range $[0, 2\pi[$ radians, on a virtual detector with 640 pixels. Scaling motion was simulated by applying a constant scaling factor every fifth projection, which results in 21 different scales across the duration of the scan. Variants of this example, with other motion parameters and images, can be found in [21]. A simplified variant is available in the documentation of ImWIP.

Objective function and gradient: Let $\boldsymbol{W}_{\text{stationary}}$ represent the static CT operator mapping images to projection space according to the specified CT geometry. This operator can be written as

$$\boldsymbol{W}_{\text{stationary}} = \begin{bmatrix} \boldsymbol{W}_1 \\ \boldsymbol{W}_2 \\ \vdots \\ \boldsymbol{W}_n \end{bmatrix} \quad , \tag{9}$$

where the blocks $\boldsymbol{W}_1, \dots \boldsymbol{W}_n, n = 21$ correspond to the projection operators of the subscans. Following [8,33], the CT model can be extended to include a motion model $\boldsymbol{M}(\boldsymbol{p})$, where $\boldsymbol{p} \in \mathbb{R}^n$ are the scaling parameters, as follows:

$$\boldsymbol{W}\boldsymbol{M}(\boldsymbol{p}) = \begin{bmatrix} \boldsymbol{W}_1 & 0 & 0 & 0 \\ 0 & \boldsymbol{W}_2 & 0 & 0 \\ & & \ddots & \\ 0 & 0 & 0 & \boldsymbol{W}_n \end{bmatrix} \begin{bmatrix} \boldsymbol{M}(p_1) \\ \boldsymbol{M}(p_2) \\ \vdots \\ \boldsymbol{M}(p_n) \end{bmatrix} \quad . \tag{10}$$

The CT projection operators $(\boldsymbol{W}_i)_{i=1}^n$ are provided by the ASTRA toolbox [34], and the affine image warping operators $(\boldsymbol{M}(p_i))_{i=1}^n$ are provided by ImWIP, all matrix-free. These matrix-free operators can be combined in block operators using PyLops [35]. The inverse problem can now be formulated as

$$\boldsymbol{W}\boldsymbol{M}(\boldsymbol{p})\boldsymbol{x} = \boldsymbol{b} \quad , \tag{11}$$

where the unknowns are the motion parameters $\boldsymbol{p}$ and the unknown image $\boldsymbol{x} \in \mathbb{R}^{512^2}$. The vector $\boldsymbol{b} \in \mathbb{R}^{105 \cdot 640}$ is the concatenation of all 105 projections. The least-squares functional corresponding to Eq. (11) is

$$\text{Loss}_{\text{CT}}(\boldsymbol{p}, \boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{r}\|^2 \quad , \tag{12}$$

where $\boldsymbol{r}$ is the residue

$$\boldsymbol{r} = \boldsymbol{W}\boldsymbol{M}(\boldsymbol{p})\boldsymbol{x} - \boldsymbol{b} \quad . \tag{13}$$

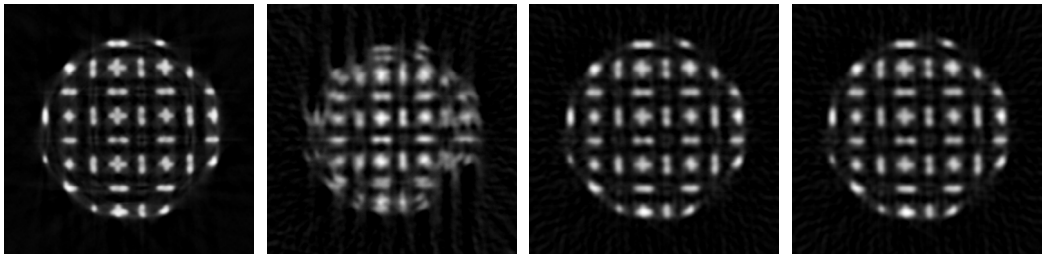
The gradient of this objective function is

$$\nabla \text{Loss}_{CT} = \left(\nabla_{\boldsymbol{p}} \text{Loss}_{\text{CT}}^T, \nabla_{\boldsymbol{x}} \text{Loss}_{\text{CT}}^T\right)^T \quad , \tag{14}$$

where

$$\nabla_{\boldsymbol{p}} \text{Loss}_{\text{CT}}(\boldsymbol{p}, \boldsymbol{x}) = \left[\frac{d}{d\boldsymbol{p}}\boldsymbol{M}(\boldsymbol{p})\boldsymbol{x}\right]^T \boldsymbol{W}^T\boldsymbol{r} \quad , \tag{15}$$

$$\nabla_{\boldsymbol{x}} \text{Loss}_{\text{CT}}(\boldsymbol{p}, \boldsymbol{x}) = \boldsymbol{M}^T(\boldsymbol{p})\boldsymbol{W}^T\boldsymbol{r} \quad . \tag{16}$$



(a) Ground truth (b) Without motion correction (c) Correction with unknown motion (d) Correction with known motion

Fig. 4. Reconstructions of the bone scaffold, without and with motion correction.

This gradient features a combination of adjoint image warping ($\boldsymbol{M}^T(\boldsymbol{p})$) and differentiated image warping ($\frac{d}{d\boldsymbol{p}}\boldsymbol{M}(\boldsymbol{p})\boldsymbol{x}$).

Results: The static and dynamic models (with and without known motion parameters) were inverted using 600 gradient descent with the Barzilai–Borwein step size. Fig. 4(a) shows the ground truth for this experiment. In Fig. 4(b), the reconstruction based on the static CT model is shown, in which the effect of the scaling motion is clearly visible in the form motion artifacts. The dynamic reconstruction with unknown motion is shown in Fig. 4(c), where it can be visually observed that the motion was correctly estimated and compensated. The dynamic reconstruction with known motion is shown in Fig. 4(d) as a reference.

4. Impact

Applications of adjoint and differentiated image warping are widely spread among different domains. ImWIP is purposefully designed with a modular approach, focusing on motion and deformation modeling only, such that it can be combined with domain specific packages such as (but not restricted to) the ASTRA-Toolbox [34] for CT, or the Fast Fourier Transform for MRI and PyTorch [36] for deep learning. For this reason, ImWIP will facilitate the development of motion/deformation compensated imaging techniques, and lead to more accurate results, across different domains.

Currently, ImWIP has provided the basis for a study on adjoint image warping in 4D-CT [37], and for the ongoing development of algorithms that combine image reconstruction and motion estimation in dynamic CT [21,38]. ImWIP has also supported the development of novel MRI reconstruction method that directly computes deformation between longitudinal MRI scans [14]. Furthermore, it is being applied to extensions of a motion compensated super-resolution technique [29], with the goal of accelerating this method.

5. Conclusions

IMWIP, an open-source, GPU-accelerated image warping toolbox, was presented and demonstrated on several applications. The toolbox includes common approximate implementations of adjoint and differentiated image warping, as well as novel algorithms that compute these operators exactly and that are independent of the choice of splines. The toolbox allows for modular design of algorithms in which the implementation of adjoint or differential image warping can be easily altered, for comparison. It was shown that, when estimating a DVF, the exact differentiated image warping leads to substantial improvement in convergence speed, compared to the implementation based on finite differences.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

Jens Renders is an SB PhD fellow at the Research Foundation Flanders, Belgium (Grant: 1SA2920N). Ben Jeurissen gratefully acknowledges support from Research Foundation Flanders, Belgium (Grant: G090020N) and Belgian Science Policy Prodex (Grant: ISLRA 2009–1062). Anh-Tuan Nguyen is supported by the Research Foundation Flanders, Belgium (Grant: S007219N).

Appendix. Implementation details

The sparse matrix representation of the image warping operators of ImWIP are computed as in [37]. In the case of tricubic image warping, this matrix $\boldsymbol{M}$ has 64 non-zero coefficients per row, as each voxel of the warped image depends on 64 voxels of the input image. The computation of these coefficients boils down to the evaluation of 64 polynomials, denoted $b_{i,j,k}(x, y, z)$ for $(i, j, k) \in \{-1, 0, 1, 2\}^3$. As in [37], these polynomials are found using SageMath and compiled into ImWIP.

Next to the construction of the sparse matrix representation, the polynomials are also used for the matrix-free adjoint, differentiated and regular image warping implementations, where the matrix coefficients are computed on the fly. Regular image warping is implemented as a row-wise matrix–vector multiplication (Algorithm 1). Adjoint image warping is implemented as a column-wise matrix–vector multiplication with the same polynomials (Algorithm 2) and differentiated image warping is implemented as a row-wise matrix–vector multiplication with differentiated polynomials (Algorithm 3, without loss of generality differentiated to the first component of the vector field). The memory footprint of these algorithms is dominated by the size of the input and output images. All other allocations are local variables with small footprint.

ImWIP also provides commonly used approximations to adjoint and differentiated image warping. Adjoint image warping can be approximated using a regular image warp along an approximate inverse DVF. The effect of this approximation compared to the exact implementation is investigated in [37] . Differentiated image warping (without loss of generality to the first

component of the DVF) can be approximated by applying an image gradient to the warped image:

$$\frac{d}{d\boldsymbol{v}_x}\left(\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f}\right) \approx \mathrm{diag}(\nabla_x\left(\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f}\right)) \quad . \tag{A.1}$$

This approach appears in the implementation of many optical flow algorithms [11]. It is inexact, since it neglects the interpolation method used in the warp and uses an image gradient operator that is based on finite differences (central differences in ImWIP).

Algorithm 1 Tricubic image warping

Input Image $\boldsymbol{f}$, DVF $\boldsymbol{v}$
Output Warped image $\boldsymbol{g}$

$\boldsymbol{f} = \boldsymbol{0}$
for each voxel position (p_1, p_2, p_3) **do in parallel**
  $(q_1, q_2, q_3) = (p_1, p_2, p_3) + v(p_1, p_2, p_3)$
  $(i', j', k') = (\lfloor q_1 \rfloor, \lfloor q_2 \rfloor, \lfloor q_3 \rfloor)$
  for $(i, j, k) \in \{-1, 0, 1, 2\}^3$ **do**
    $g(p_1, p_2, p_3)$ += $b_{ijk}(i' - q_1, j' - q_2, k' - q_3) f(i' + i, j' + j, k' + k)$
  end for
end for

Algorithm 2 Adjoint tricubic image warping

Input Image $\boldsymbol{g}$, DVF $\boldsymbol{v}$
Output Adjoint warped image $\boldsymbol{f}$

$\boldsymbol{f} = \boldsymbol{0}$
for each voxel position (p_1, p_2, p_3) **do in parallel**
  $(q_1, q_2, q_3) = (p_1, p_2, p_3) + v(p_1, p_2, p_3)$
  $(i', j', k') = (\lfloor q_1 \rfloor, \lfloor q_2 \rfloor, \lfloor q_3 \rfloor)$
  for $(i, j, k) \in \{-1, 0, 1, 2\}^3$ **do**
    $f(i' + i, j' + j, k' + k)$ += $b_{ijk}(i' - q_1, j' - q_2, k' - q_3) g(p_1, p_2, p_3)$
  end for
end for

Algorithm 3 Differentiated tricubic image warping

Input Image $\boldsymbol{f}$, DVF $\boldsymbol{v}$
Output Diagonal matrix $\frac{d}{d\boldsymbol{v}_x}\left(\boldsymbol{M}(\boldsymbol{v})\boldsymbol{f}\right) = \mathrm{diag}(\boldsymbol{d})$

$\boldsymbol{d} = \boldsymbol{0}$
for each voxel position (p_1, p_2, p_3) **do in parallel**
  $(q_1, q_2, q_3) = (p_1, p_2, p_3) + v(p_1, p_2, p_3)$
  $(i', j', k') = (\lfloor q_1 \rfloor, \lfloor q_2 \rfloor, \lfloor q_3 \rfloor)$
  for each $(i, j, k) \in \{-1, 0, 1, 2\}^3$ **do**
    $d(p_1, p_2, p_3)$ +=
    $(\frac{d}{dx} b_{ijk})(i' - q_1, j' - q_2, k' - q_3) f(i' + i, j' + j, k' + k)$
  end for
end for

References

[1] Wolberg G. Digital image warping, vol. 10662. IEEE Computer Society Press Los Alamitos, CA; 1990.
[2] Glasbey CA, Mardia KV. A review of image-warping methods. J Appl Stat 1998;25(2):155–71.
[3] Markussen B. Large deformation diffeomorphisms with application to optic flow. Comput Vis Image Underst 2007;106(1):97–105.
[4] Feng T, Wang J, Fung G, Tsui B. Non-rigid dual respiratory and cardiac motion correction methods after, during, and before image reconstruction for 4D cardiac PET. Phys Med Biol 2015;61(1):151.
[5] Odstrcil M, Holler M, Raabe J, Sepe A, Sheng X, Vignolini S, et al. Ab initio nonrigid X-ray nanotomography. Nature Commun 2019;10(1):2600.
[6] Rueckert D, Sonoda LI, Hayes C, Hill DLG, Leach MO, Hawkes DJ. Non-rigid registration using free-form deformations: Application to breast MR images. IEEE Trans Med Imaging 1999;18(8):712–21.
[7] van Heeswijk RB, Bonanno G, Coppo S, Coristine A, Kober T, Stuber M. Motion compensation strategies in magnetic resonance imaging. Crit Rev Biomed Eng 2012;40(2).
[8] Van Nieuwenhove V, De Beenhouwer J, Vlassenbroeck J, Brennan M, Sijbers J. MoVIT: A tomographic reconstruction framework for 4D-CT. Opt Express 2017;25(16):19236–50.
[9] Zang G, Idoughi R, Tao R, Lubineau G, Wonka P, Heidrich W. Warp-and-project tomography for rapidly deforming objects. ACM Trans Graph 2019;38(4):86.
[10] Horn BKP, Schunck BG. Determining optical flow. Artificial Intelligence 1981;17(1–3):185–203.
[11] Pérez JS, Meinhardt-Llopis E, Facciolo G. TV-L1 optical flow estimation. Image Process Line 2013;2013:137–50.
[12] Agarwal A, Gupta S, Singh DK. Review of optical flow technique for moving object detection. In: 2016 2nd International conference on contemporary computing and informatics. IEEE; 2016, p. 409–13.
[13] Andersson JL, Skare S, Ashburner J. How to correct susceptibility distortions in spin-echo echo-planar images: Application to diffusion tensor imaging. Neuroimage 2003;20(2):870–88.
[14] Renders J, Shafieizargar B, Verhoye M, De Beenhouwer J, den Dekker AJ, Sijbers J. DELTA-MRI: Direct deformation estimation from LongiTudinally acquired k-space data. 2023, arXiv:2301.09455.
[15] Chumchob N, Chen K. A robust affine image registration method. Int J Numer Anal Model 2009;6(2).
[16] Lakshmanan AG, Swarnambiga A, Vasuki S, Raja AA. Affine based image registration applied to MRI brain. In: 2013 International conference on information communication and embedded systems. IEEE; 2013, p. 644–9.
[17] Ramos-Llordén G, den Dekker AJ, Van Steenkiste G, Jeurissen B, Vanhevel F, Van Audekerke J, et al. A unified maximum likelihood framework for simultaneous motion and T_1 estimation in quantitative MR T_1 mapping. IEEE Trans Med Imaging 2017;36(2):433–46.
[18] Beirinckx Q, Jeurissen B, Nicastro M, Poot DH, Verhoye M, den Dekker AJ, et al. Model-based super-resolution reconstruction with joint motion estimation for improved quantitative MRI parameter mapping. Comput Med Imaging Graph 2022;100:102071.
[19] Bousse A, Bertolli O, Atkinson D, Arridge S, Ourselin S, Hutton BF, et al. Maximum-likelihood joint image reconstruction/motion estimation in attenuation-corrected respiratory gated PET/CT using a single attenuation map. IEEE Trans Med Imaging 2015;35(1):217–28.
[20] Burger M, Dirks H, Frerking L, Hauptmann A, Helin T, Siltanen S. A variational reconstruction method for undersampled dynamic X-ray tomography based on physical motion models. Inverse Problems 2017;33(12):124008.
[21] Nguyen A, Renders J, Soete J, Wevers M, Sijbers J, De Beenhouwer J. An accelerated motion-compensated iterative reconstruction technique for dynamic computed tomography. In: Developments in X-ray tomography XIV, vol. 12242, SPIE; 2022.
[22] Le Besnerais G, Champagnat F. B-spline image model for energy minimization-based optical flow estimation. IEEE Trans Image Process 2006;15(10):3201–6.
[23] Balakrishnan G, Zhao A, Sabuncu MR, Guttag J, Dalca AV. VoxelMorph: A learning framework for deformable medical image registration. IEEE Trans Med Imaging 2019;38(8):1788–800.
[24] Van Houtte J, Audenaert E, Zheng G, Sijbers J. Deep learning-based 2D/3D registration of an atlas to biplanar X-ray images. Int J Comput Assist Radiol Surg 2022;1–10.
[25] Renders J, De Beenhouwer J, Sijbers J. ImWIP: Image warping for inverse problems. 2023, http://dx.doi.org/10.5281/zenodo.5910755.
[26] Van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, et al. Scikit-image: Image processing in Python. PeerJ 2014;2:e453.
[27] Virtanen P, Gommers R, Oliphant TE, SciPy 10 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 2020;17:261–72.
[28] Bradski G. The OpenCV library. Dr. Dobb's J Software Tools 2000.
[29] Beirinckx Q, Ramos-Llordén G, Jeurissen B, Poot DH, Parizel PM, Verhoye M, et al. Joint maximum likelihood estimation of motion and T1 parameters from magnetic resonance images in a super-resolution framework: A simulation study. Fund Inform 2020;172(2):105–28.
[30] Barzilai J, Borwein JM. Two-point step size gradient methods. IMA J Numer Anal 1988;8(1):141–8.
[31] Fletcher R. Practical methods of optimization. John Wiley & Sons; 2013.

[32] Jeurissen B, Tournier J-D, Dhollander T, Connelly A, Sijbers J. Multi-tissue constrained spherical deconvolution for improved analysis of multi-shell diffusion MRI data. NeuroImage 2014;103:411–26.
[33] Van Eyndhoven G, Sijbers J, Batenburg J. Combined motion estimation and reconstruction in tomography. In: European conference on computer vision, vol. 7583. Springer; 2012, p. 12–21.
[34] van Aarle W, Palenstijn WJ, Cant J, Janssens E, Bleichrodt F, Dabravolski A, et al. Fast and flexible X-ray tomography using the ASTRA toolbox. Opt Express 2016;24(22):25129–47.
[35] Ravasi M, Vasconcelos I. Pylops—A linear-operator python library for scalable algebra and optimization. SoftwareX 2020;11:100361.
[36] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32. Curran Associates, Inc.; 2019, p. 8024–35.
[37] Renders J, Sijbers J, De Beenhouwer J. Adjoint image warping using multivariate splines with application to four-dimensional computed tomography. Med Phys 2021;48(10):6362–74.
[38] Nguyen A-T, Renders J, Sijbers J, De Beenhouwer J. Region-based motion-compensated iterative reconstruction technique for dynamic computed tomography. 2023, arXiv:2301.11029.