The TVDMFFDVFR Algorithm

Release 0.00

Nicholas J. Tustison[1] and Brian B. Avants[2]

January 3, 2012

[1]Department of Radiology and Medical Imaging, University of Virginia
[2]Penn Image Computing and Sciences Laboratory, University of Pennsylvania

Abstract

The recent ITKv4 refactoring includes several enhancements to the existing registration framework. These additional transform classes provide access to mappings described by dense displacement fields and their corresponding optimization which complement the popular *free-form deformation* (FFD) approach already in ITK. Innovation motivated by previous work [5] and recent diffeomorphic image registration developments in which the characteristic velocity field is represented by spatiotemporal B-splines [2], resulted in a diffeomorphic B-spline-based image registration algorithm combining and extending these techniques which we make available in ITK through the gerrit system. Additionally, we include two command line tools showcasing the new elements of the registration refactoring for 1) computing mappings between two images (*antsRegistration*) including the family of transforms discussed in this article and 2) applying those transformations to images (*antsApplyTransforms*). **NB: The user must download the patch available at `http://review.source.kitware.com/#/c/3606/` in order to compile the code accompanying this article.**

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

Contents

1 Introduction

The recent major updating of ITK, denoted as ITKv4, resulted in significant modifications to the toolkit's registration capabilities. A major element of the registration contributions was the addition of dense displacement field transforms. These new displacement field transforms include invertible mappings obtained by integrating time-varying velocity fields popularized by algorithms such as the *large deformation diffeomorphic metric mapping* (LDDMM) approach described in [1]. These displacement field transforms complement the preexisting ITK implementation of the nonrigid B-spline transform used for FFD image registration [3]. Recently, the work of [2] hybridized these registration concepts into a single framework called *temporal free-form deformation* in which the time-varying velocity field characteristic of LDDMM-style algorithms is modeled using a $(d+1)$-D B-spline object (d-D + time), i.e.

$$v(\mathbf{x},t) = \sum_{i_1=1}^{X_1} \cdots \sum_{i_d=1}^{X_d} \sum_{i_t=1}^{T} v_{i_1,\ldots,i_d,i_t} B_{i_t}(t) \prod_{j=1}^{d} B_{i_j}(x_j) \tag{1}$$

where $v_{i_1,\ldots,i_d,i_t}$ is a $(d+1)$-dimensional control point lattice characterizing the velocity field and $B(\cdot)$ are the univariate B-spline basis functions separately modulating the solution in each parametric dimension. Numerical integration of the characteristic ordinary differential equation

$$\frac{d\phi(\mathbf{x},t)}{dt} = v(\phi(\mathbf{x},t),t),\ \phi(\mathbf{x},0) = \mathbf{x} \tag{2}$$

produces the diffeomorphic mapping.

In this article, we describe a modified implementation of this algorithm using B-splines which we are submitting for inclusion in the toolkit using the gerrit system available at `http://review.source.kitware.com/#/c/3606/`. It differs in several respects from the work in [2] with the major variation being an adaptation of our previous *directly manipulated free-form deformation* (DMFFD) work described in [5]. We also describe two user programs that we created called `antsRegistration` and `antsApplyTransforms` to demonstrate usage of both the proposed time-varying directly manipulated free-form deformation velocity field registration (TVDMFFDVFR) algorithm as well as major elements of the new registration framework. These programs are also available in the ANTs open source package[1] but rely solely on ITKv4 contributions (except for a couple small ANTs command line classes)

We first describe the TVDMFFDVFR algorithm and the defining classes. Note that this discussion is fairly restricted to implementation and use issues. A forthcoming conference submission will provide a more theoretical discussion as well as a more thorough and larger scale validation. Usage is facilitated with the `antsRegistration` and `antsApplyTransforms` programs. As these classes are applicable to all ITKv4 registration classes, we describe these programs in depth as their description, particularly of usage, will serve to ground the interested reader for further exploration of registration in ITKv4.

2 TVDMMFDVFR Implementation

Pseudocode for pairwise TVDMMFDVFR is given in Algorithm 1. Input consists of two d-dimensional images, I and J. The following items are selected on the command line by the user

- **Similarity metric**: Currently mutual information, neighborhood cross-correlation, and a Demons-style sum of squared differences are available for similarity metric choices using the new framework.

[1] http://www.picsl.upenn.edu/ANTs

- **Velocity field mesh size**: This parameter specifies the initial B-spline mesh resolution in both space and time. Note that this is not the number of control points in each parametric dimension as we find it more user-intuitive to specify the resolution in terms of a uniform, rectilinear mesh situated over the transformation domain.[2] Although it's possible to specify an arbitrary multi-resolution schedule for the control point lattice, for simplicity in interfacing with `antsRegistration`, we assume that the specified resolution is given for the base resolution and each level results in a mesh size twice that of the previous level. As an example, an initial mesh resolution of 5 elements in each of the three image directions and a temporal resolution of 4 mesh elements would be specified as '`5x5x5x4`'.
- **Number of time point samples**: For each time point sample between 0 and 1, the fixed image is warped to time point t simultaneously with an inverse warping of the moving image to the same time point. The image metric gradient is then calculated for that time point. Although more time samples provides more information for calculation of the continuous velocity field gradient, there is an accompanying increase in computational cost. We default to 4 time samples evenly spaced between 0 and 1 although this option is available on the command line.
- **Spline order**: Although typical spline order is cubic for most applications, the user can choose arbitrary order in the range $[0,5]$ with the caveat that an increase in spline order increases computational complexity and a decrease in spline order decreases the continuity of the field. Also, it might possibly be interesting to separately vary the spatial and temporal spline orders. Although that is not available on the command line of `antsRegistration`, it would be simple to do with slight modification to the classes described below.
- **Step size**: The step size characterizes the gradient descent optimization with smaller step sizes promoting a more cautious solution propagation in the direction of the current gradient. The step size is scaled based on the maximum similarity metric gradient value and voxel size so a typical choice for all registration possibilities is in the range $[0.25,1]$.

The registration algorithm has been implemented in the Insight Toolkit and is encapsulated by the following major classes (with descriptions):

- `itk::TimeVaryingVelocityFieldIntegrationImageFilter`: In [2], Eulerian integration is used to solve Eqn. (2) where the value of $\phi(\mathbf{x},t)$ at each voxel is determined from conveyance within the $(d+1)$-d time-varying velocity field. In contrast, we use 4^{th}-order Runge Kutta for integration. which is parallelized (multi-threaded) over the domain of $\phi(\mathbf{x},t')$ for a specified $t=t'$. Note that this class takes as input a $(d+1)$-d image of d-D vectors describing the time-varying velocity field. The knowledgeable reader might wonder why we did not implement a specialized integrator which takes as input the velocity field control point lattice where numerical integration is done in the continuous space of the B-spline velocity field. The problem with this approach is that it is computationally expensive. Numerical integration requires multiple sampling of the velocity field for each point in $\phi(\mathbf{x},t')$ and each sample point requires the weighted average of the proximal $\Pi_{j=1}^{d+1}(k_j+1)$ control points and corresponding polynomial evaluations. So, in the common case of cubic splines, $k=3$, with 3-D images, each sample point requires $4^4=256$ polynomial evaluations.[3]

[2] The mesh resolution in each parametric dimension is simply

$$\text{mesh size} = \text{number of control points} - \text{spline order}. \tag{3}$$

[3] If it were not so recent, the first author might find it humorous that the realization of the computationally taxing nature of this approach to numerical integration followed (instead of preceded and thereby prevented) implementation. Oh well, it was an interesting academic exercise.

Algorithm 1 The TVDMMFDVFR algorithm applied to a fixed/moving image pair

Input: images I and $\mathcal{J}$
Input: user-specified similarity metric Π
Input: velocity field mesh size and multiresolution schedule, R
Input: number of time point samples, N_t (default: 4)
Input: spline order (default: 3)
Input: gradient step size, λ (default: 0.25)
Output: velocity field control point lattice, v

$v \leftarrow \mathbf{0}$ // Initialize velocity field
for number of resolution levels **do**
  for number of iterations for current level **do**
    $G \leftarrow []$ // Initialize dense gradient storage array
    for $t = 1 \rightarrow N_t$ **do**
      $t' \leftarrow \frac{t-1}{N_t-1}$ // Scale time point to $[0,1]$
      $I' \leftarrow I \circ \phi(\mathbf{x}, t')$ // Warp I to the current time point
      $\mathcal{J}' \leftarrow \mathcal{J} \circ \phi^{-1}(\mathbf{x}, 1-t')$ // Warp $\mathcal{J}$ to the current time point
      $G[t] \leftarrow \frac{\partial \Pi(I', \mathcal{J}')}{\partial \mathbf{x}}$ // Store dense similarity metric gradient
    end for
    $\delta v \leftarrow B(G)$ // Calculate velocity control point lattice gradient
    $v \leftarrow v + \lambda \delta v$ // Take a step in the gradient direction
  end for
  $v \leftarrow R(v)$ // Refine lattice based on schedule
end for

- `itk::TimeVaryingBSplineVelocityFieldTransform`: This transform class, derived from the `itk::DisplacementFieldTransform`, coordinates the integration of the sampled DMFFD velocity field and uses the generated displacement field and its inverse to map geometric primitives (consistent with all the transform classes).

- `itk::TimeVaryingBSplineVelocityFieldImageRegistrationMethod`: This class is derived from `itk::ImageRegistrationMethodv4` which is the generic image registration handling method class. This base class coordinates the major elements of a standard image registration scenario: a fixed/moving image pair, optimization strategy, image metric, transformation, and some multi-resolution choices and determines the optimal transformation parameters. We refer to this collection of parameters as an image registration *stage*. Each image registration stage can be concatenated to another registration stage through the `itk::CompositeTransform`. For most transforms this class is sufficient. However, these time-varying velocity field registration strategies require specialization of the optimization or the image metric classes. We decided to handle these specializations in this separate image registration method class.

- `itk::TimeVaryingBSplineVelocityFieldTransformParametersAdaptor`: Traditional FFD algorithm often employ multiresolution strategies often increase the resolution of the control point lattice propagating the solution from the lower resolution to the higher resolution. Similarly, we allow the user to change the velocity field control point lattice resolution derived from our earlier contribution `itk::TransformParametersAdaptor` for a generic handling of adapting the transform parameters.

- `itk::BSplineScatteredDataPointSetToImageFilter`: Although this was a previous contribution intended for generalized B-spline m-dimensional parametric fitting of n-dimensional data [4], we

list it here as it yields the preconditioned control point lattice gradient during DMFFD optimization. Further details are given in [5].

3 antsRegistration and antsApplyTransforms

With the inputs and implementation explained, we can demonstrate how it is all put together using `antsRegistration`. Note that this is still a work in progress and will be updated quite frequently in the ANTs repository.

The basic idea behind `antsRegistration` is that a savvy user is going to perform image registration by specifying multiple image registration stages where each registration stage has its own options and outputs its own optimized transform. Each optimized transform corresponding to its own stage is going to be linked via the `itk::CompositeTransform`. During the running of the program, after each stage is finished, the optimized transform is written to disk. We show the help menu (Listing 1) which can be invoked by typing `antsRegistration --help` at the command prompt and describe certain options which have not been discussed in previous sections (although the help menu should provide sufficient information to get started).

- **-d, –dimensionality**: `antsRegistration` tries to infer the image dimensionality by reading the fixed image but the user can force a dimensionality choice using this option.
- **-u, –useHistogramMatching**: Boolean to histogram match the fixed and moving images using `itk::HistogramMatchingImageFilter` with 12 match points and 256 histogram bins.
- **-r, –initialTransform**: Option to specify the initial transform between the fixed and moving images. Otherwise identity is used to initialize the composite transform.
- **-o, –output**: The principal output is an optimized transform or set of optimized transforms mapping the moving image to the fixed image. The value for this option is the string prefix attached to each of the transforms. For example, if the user performs a
- **-s, –smoothingSigmas**: Using `itk::DiscreteGaussianImageFilter` the fixed and moving images are smoothed using the specified sigma values for each multi-resolution level.
- **-f, –shrinkFactors**: Using `itk::ShrinkImageFilter` the fixed and moving images are resized in each dimension by the specified shrink factors.

```
[ntustison@Nietzschean-Numerics] $ antsRegistration --help

COMMAND:
     antsRegistration
          This program is a user-level registration application meant to utilize
          ITKv4-only classes. The user can specify any number of "stages" where a stage
          consists of a transform; an image metric; and iterations, shrink factors, and
          smoothing sigmas for each level.

OPTIONS:
     -d, --dimensionality 2/3
          This option forces the image to be treated as a specified-dimensional image. If
          not specified, N4 tries to infer the dimensionality from the input image.

     -o, --output [outputTransformPrefix,<outputWarpedImage>,<outputInverseWarpedImage>]
          Specify the output transform prefix (output format is .nii.gz ).Optionally, one
          can choose to warp the moving image to the fixed space and, if the inverse
          transform exists, one can also output the warped fixed image.
```

```
-r, --initialTransform [initialTransform,useInverse]
     Specify the initial transform(s) which get immediately incorporated into the
     composite transform. The order of the transforms is stack-esque in that the last
     transform specified onthe command line is the first to be applied. See
     antsApplyTransformsfor additional information.


-m, --metric CC[fixedImage,movingImage,metricWeight,radius]
             MI[fixedImage,movingImage,metricWeight,numberOfBins]
             Demons[fixedImage,movingImage,metricWeight]
     Three image metrics are available--- CC: ANTS neighborhood cross correlation,
     MI: Mutual information, and Demons: Thirion's Demons (modified mean-squares).
     Note that the metricWeight is currently not used. Rather, it is a temporary
     place holder until multivariate metrics are available for a single stage.

-t, --transform Rigid[gradientStep]
                Affine[gradientStep]
                Similarity[gradientStep]
                Translation[gradientStep]
                BSpline[gradientStep,meshSizeAtBaseLevel]
                GaussianDisplacementField[gradientStep,updateFieldSigmaInPhysicalSpace,
                                          totalFieldSigmaInPhysicalSpace]
                BSplineDisplacementField[gradientStep,updateFieldMeshSizeAtBaseLevel,
                                         totalFieldMeshSizeAtBaseLevel,<splineOrder=3>]
                TimeVaryingVelocityField[gradientStep,numberOfTimeIndices,
                                         updateFieldSigmaInPhysicalSpace,
                                         updateFieldTimeSigma,totalFieldSigmaInPhysicalSpace,
                                         totalFieldTimeSigma]
                TimeVaryingBSplineVelocityField[gradientStep,velocityFieldMeshSize,
                                                <numberOfTimePointSamples=4>,<splineOrder=3>]
     Several transform options are available. The gradientStep or learningRate
     characterizes the gradient descent optimization and is scaled appropriately for
     each transform using the shift scales estimator. Subsequent parameters are
     transform-specific and can be determined from the usage.

-i, --iterations MxNx0...
     Specify the number of iterations at each level.

-s, --smoothingSigmas MxNx0...
     Specify the amount of smoothing at each level.

-f, --shrinkFactors MxNx0...
     Specify the shrink factor for the virtual domain (typically the fixed image) at
     each level.

-u, --useHistogramMatching
     Histogram match the images before registration.

-h
     Print the help menu (short version).
     <VALUES>: 0

--help
     Print the help menu.
     <VALUES>: 1, 0
```

Listing 1: antsRegistration long command line menu which is invoked using the '`--help`' option. The short command line menu is obtained by typing '`-h`'.

Once the registration algorithm is finished, the user will have a set of transforms corresponding to each specified stage. For example, suppose the user selects to perform the following series of registration stages with the output prefix `myRegistration`:

- **Stage 0**: Rigid

- **Stage 1**: Affine
- **Stage 2**: GaussianDisplacementField

This will result in the following output files

- **Stage 0**: `myRegistration0Rigid.mat`
- **Stage 1**: `myRegistration1Affine.mat`
- **Stage 2**: `myRegistration2Warp.nii.gz`

where the `.mat` files are ITK transform files and the displacement field transforms are specified by displacement field images. It should be noted that, where applicable, an inverse displacement field transform will be specified by the file name '`${outputPrefix}_${currentStage}InverseWarp.nii.gz`'.

If the user wants to apply these transforms to warp a separate image (or set of images), she can use `antsApplyTransforms` with the help menu given below. The help menu describes how the transforms are applied in stack fashion. Several interpolators are possible including a couple new interpolators written by Paul Yushkevich. Note that these are not in the toolkit yet but are intended to be considered for inclusion.

```
[ntustison@Nietzschean-Numerics] $ antsApplyTransforms --help

COMMAND:
     antsApplyTransforms
          antsApp is a sweet piece of code.

OPTIONS:
     -d, --dimensionality 2/3
          This option forces the image to be treated as a specified-dimensional image. If
          not specified, antsWarp tries to infer the dimensionality from the input image.

     -i, --input inputFileName
          Currently, the only input objects supported are image objects. However, the
          current framework allows for warping of other objects such as meshes and point
          sets.

     -r, --reference-image imageFileName
          For warping input images, the reference image defines the spacing, origin, size,
          and direction of the output warped image.

     -o, --output warpedOutputFileName
          The warped object---currently only valid for images.

     -n, --interpolation Linear
                         NearestNeighbor
                         MultiLabel[<sigma=imageSpacing>,<alpha=4.0>]
                         Gaussian[<sigma=imageSpacing>,<alpha=1.0>]
                         BSpline[<order=3>]
                         CosineWindowedSinc
                         WelchWindowedSinc
                         HammingWindowedSinc
                         LanczosWindowedSinc
          Several interpolation options are available in ITK. These have all been made
          available.

     -t, --transform transformFileName
                     [transformFileName,useInverse]
          Several transform options are supported including all those defined in the ITK
          library in addition to a deformation field transform. The ordering of the
          transformations follows the ordering specified on the command line. An identity
          transform is pushed onto the transformation stack. Each new transform
```

```
          encountered on the command line is also pushed onto the transformation stack.
          Then, to warp the input object, each point comprising the input object is warped
          first according to the last transform pushed onto the stack followed by the
          second to last transform, etc. until the last transform encountered which is the
          identity transform. Also, it should be noted that the inverse transform can be
          accommodated with the usual caveat that such an inverse must be defined by the
          specified transform class

     -v, --default-value value
          Default voxel value to be used with input images only. Specifies the voxel value
          when the input point maps outside the output domain

     -h
          Print the help menu (short version).
          <VALUES>: 0

     --help
          Print the help menu.
          <VALUES>: 1, 0
```

Listing 2: antsApplyTransforms long command line menu which is invoked using the '`--help`' option. The short command line menu is obtained by typing '`-h`'.

4 Example Usage

We show a simple 2-D example using two images available in ITK shown in Fig. 1.

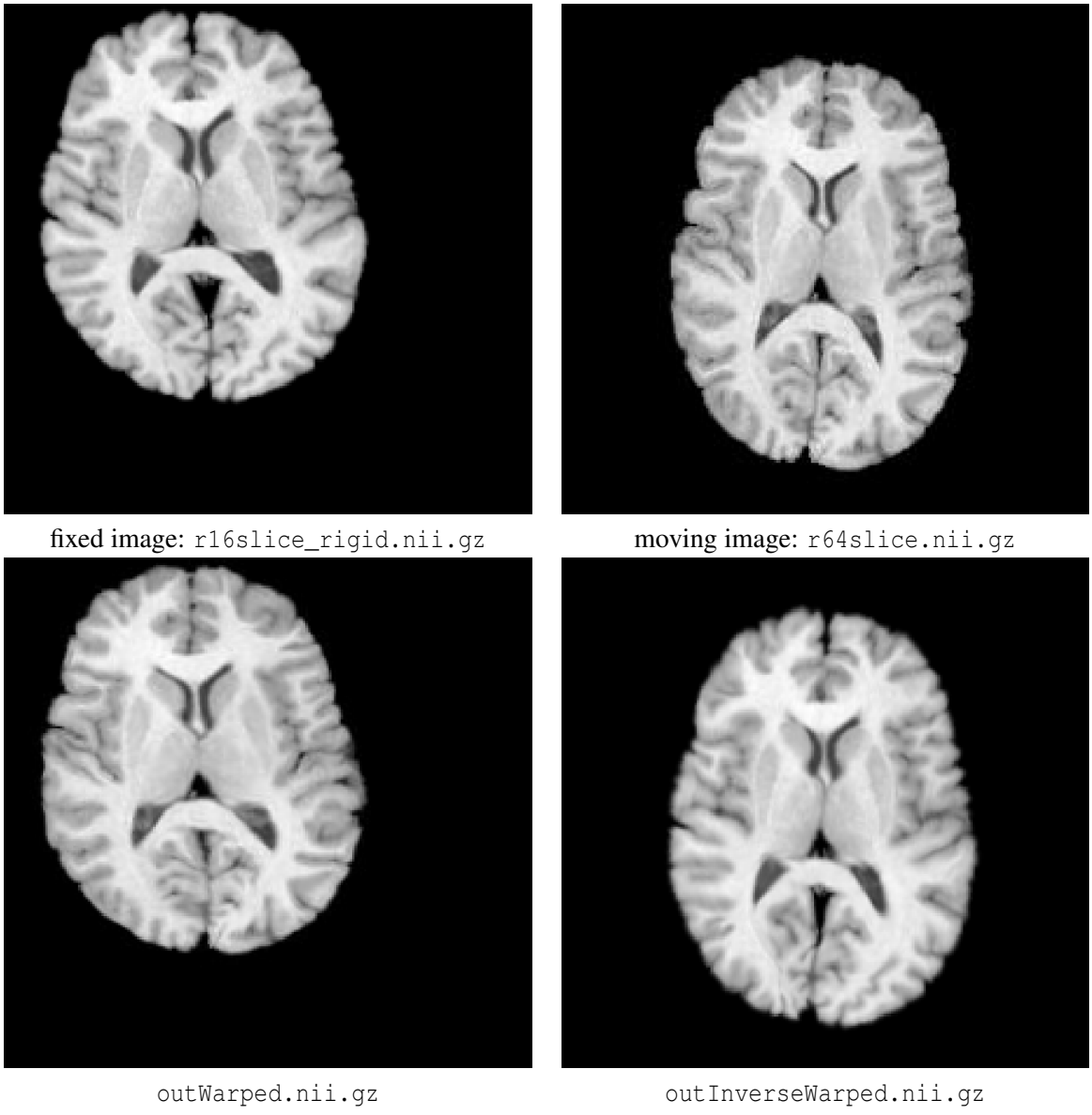
```
$ antsRegistration -d 2 -o [out,outWarped.nii.gz,outInverseWarped.nii.gz] \
                   -m MI[r16slice_rigid.nii.gz,r64slice.nii.gz,1,32] \        # Affine stage
                   -t Affine[0.5] \
                   -i 100x100 \
                   -s 2.0x1.0 \
                   -f 3x2 \
                   -u 1 \
                   -m CC[r16slice_rigid.nii.gz,r64slice.nii.gz,1,4] \        # TVDMFFDVR stage
                   -t TimeVaryingBSplineVelocityField[0.95,4x4x2,4,3] \
                   -i 30x20 \
                   -s 1.0x0.5 \
                   -f 2x1 \
                   -u 1
```

References

[1] M. Faisal Beg, Michael I. Miller, Alain Trouvé, and Laurent Younes. Computing large deformation metric mappings via geodesic flows of diffeomorphisms. *Int. J. Comput. Vision*, 61:139–157, February 2005. 1

[2] Mathieu De Craene, Gemma Piella, Oscar Camara, Nicolas Duchateau, Etelvino Silva, Adelina Doltra, Jan D'hooge, Josep Brugada, Marta Sitges, and Alejandro F Frangi. Temporal diffeomorphic free-form deformation: Application to motion and strain estimation from 3d echocardiography. *Med Image Anal*, Nov 2011. (document), 1, 1, 2

[3] D Rueckert, L I Sonoda, C Hayes, D L Hill, M O Leach, and D J Hawkes. Nonrigid registration using free-form deformations: application to breast mr images. *IEEE Trans Med Imaging*, 18(8):712–21, Aug 1999. 1



fixed image: `r16slice_rigid.nii.gz` moving image: `r64slice.nii.gz`

`outWarped.nii.gz` `outInverseWarped.nii.gz`

Figure 1: Top row: Fixed and moving sample images to demonstrate application of the TVDMFFDVFR algorithm. Bottom row: Warped and inversely warped images.

[4] Nicholas Tustison and James Gee. Generalized n-d c^k B-spline scattered data approximation with confidence values. In Guang-Zhong Yang, TianZi Jiang, Dinggang Shen, Lixu Gu, and Jie Yang, editors, *Medical Imaging and Augmented Reality*, volume 4091 of *Lecture Notes in Computer Science*, pages 76–83. Springer Berlin / Heidelberg, 2006. 2

[5] Nicholas J Tustison, Brian B Avants, and James C Gee. Directly manipulated free-form deformation image registration. *IEEE Trans Image Process*, 18(3):624–35, Mar 2009. (document), 1, 2