FEWarp – Hyper-elastic warping with FEBio

Last edited 12/3/2014

Contents

1. Introduction

Hyperelastic warping is an image registration method that tries to find the deformation map that maps a template object to a target. The template is represented by an image (2D or 3D) and by a finite element mesh that defines the region in the template image that corresponds with the object of interest. The target is defined by an image. The finite element mesh is also used as a discretized representation of an elastic, deformable body. It is assumed that the constitutive model of this body is described by a hyper-elastic strain-energy function from which the stress of the body can be determined. Hyperelastic warping proceeds by minimizing the total potential energy of the system defined by the sum of the elastic potential energy and an image-based energy.

$$W_T = \int_B w_E(\varphi)\,dV + \int_B w_I(\varphi, T, S)\,dV \tag{1.1}$$

Here, φ is the deformation map, w_E is the hyper-elastic strain-energy density, w_I is the image-based energy, *T* represents the template image and *S* the target image. For the image-based energy term, the following form is assumed.

$$w_I(\mathbf{X}, \varphi) = \frac{1}{2} k \left[T(\mathbf{X}) - S(\varphi(\mathbf{X})) \right]^2 \tag{1.2}$$

The parameter *k* is referred to as the *warping penalty* and controls the relative influence of the image energy on the overall deformation. In other words, increasing this parameter will in general result in a better registration.

Minimization of the functional defined in (1.1) corresponds to a body-force that drives to minimize the total energy.

$$F_I(\mathbf{X}, \varphi) = k\left(T(\mathbf{X}) - S(\varphi)\right)\frac{\partial S(\varphi)}{\partial \varphi} \tag{1.3}$$

Note that this force is proportional to the image overlap, as well as the target image gradient. This implies that, in order for warping to work, there has to be some initial overlap (since initially φ is the identity mapping) and the image gradient has to be well-defined (meaning, it should be relatively smooth). The warping plugin implements some tools that can help with registration of sharp images with little or no overlap (see the Image Blurring section below).

2. Hyperelastic Warping in FEBio

The *FEWarp* plugin allows users to run warping problems with FEBio. In order to use the warping plugin, the following steps need to be completed.

1. Obtain the latest version of FEBio
2. Obtain the warping plugin

3. Specify the location of the warping plugin the FEBio configuration file.

Thus, the first step is to obtain the FEBio binaries which can be downloaded from the FEBio website (www.febio.org). On the same website, you can also download the warping plugin. The download will contain several files (including documentation and example files), but the only file you really need is the plugin file. The name of this file depends on the OS: on Windows it's named *FEWarp.dll*, on Linux it's *FEWarp.so* and on Mac it's *FEWarp.dylib*. After you've obtained this file, you need to ask FEBio to load the plugin at startup. This is done by modifying the *configuration file*, which is usually called *febio.xml* and can be found in the same folder as the FEBio executable. Once you've located the file, open it in a text editing program and add the line.

```
<import>\path\to\plugin\plugin.dll</import>
```

Note that this line has to be placed between the *febio_config* tags.

When FEBio starts, it will attempt to load the plugin. It will print a message to the screen whether it was able to load the file or not.

3. The Warping Constraint

The warping algorithm is implemented in FEBio as a constraint. Therefore it must be defined in the *Constraints* section of the FEBio input file. The *type* attribute has to be set to *warp-image*.

```
<Constraints>
      <constraint type="warp-image">
      ...
      </constraint>
</Constraints>
```

The warping parameters are defined as child tags of the *constraint* tag and are discussed next. The following table shows an overview of all the warping parameters.

parameter	description
`template`	define the template image name
`target`	define the target image name
`range_min`	define the min-coordinates of the bounding box
`range_max`	define the max-coordinates of the bounding box
`penalty`	specify the penalty parameter
`blur`	specify the blur parameter
`laugon`	activate or deactivate augmented Lagrangian
`altol`	set the augmentation tolerance

Template image

The template image is defined by the *template* tag. It takes one attribute, namely the file name. Currently, only RAW image files are supported. The size of the image is defined as a child tag which contains the number of voxels in x, y, and z. (For 2D images, use 1 for the number of z-voxels).

```
<template file="template.raw">
      <size>64,64,1</size>
</template>
```

When defined like this, the template image file is assumed to be located in the current working directory. Alternatively, you could also include the full path to the image file in the template definition.

Target image

The target image is specified similarly as the template image, except that the tag is named *target.* It takes the file name as an attribute and the size as a child tag.

```
<target file="target.raw">
      <size>64,64,1</size>
</target>
```

Image Range

The image range defines the coordinates and dimensions of the bounding box that the images correspond to in physical space. It is assumed that the units are the same as the units in which the nodal coordinates of the mesh are defined in. Note that although the template and target images may have different pixel resolutions, it is assumed that they occupy the same domain in physical space. In other words, they have the same range.

The range is defined with two tags. The *range_min* defines the coordinates of the lower-left coordinates of the box, and *range_max* defines the upper-right coordinates of the box.

```
<range_min>0,0,0,</range_min>
```

```
<range_max>64,64,1</range_max>
```

Warping penalty

The warping penalty parameter is used to control the strength of the warping forces that drive the deformation of the mesh. Since these forces could initially be really large, it is often better to ramp the warping forces up. This can be done by ramping up the penalty factor via a loadcurve.

```
<penalty lc="1">1.0</penalty>
```

Here, the *lc* attribute defines the load curve ID that defines the load curve that will be used to ramp up the penalty. This load curve needs to be defined in the *LoadData* section of the FEBio input file.

Image Blurring

The warping forces (see equation (1.3)) are proportional to the overlap of the template and target as well as to the target image gradient. When the initial overlap is small or when the target contains sharp transitions (i.e. edges), the warping algorithm may have trouble converging. To facilitate the process, the user can blur the images. Blurring the images can increase the overlap, as well as smooth the image gradients. The blurring parameter defines a blurring range (in units of pixels) and an optional loadcurve parameter.

```
<blur lc="2">1.0</blur>
```

The loadcurve parameter can be used to decrease the amount of blurring as a function of time. The recommended use would be to define a relatively large amount of blurring at the start of the simulation, and decrease the blur parameter to zero as the solutions progresses. When using a loadcurve to control blurring, the loadcurve must be defined in the LoadData section of the FEBio input file.

Augmented Lagrangian

In the context of constraint enforcement, in general, the Lagrange multipliers have to be sought that enforce the constraints exactly. In the context of warping, the Lagrange multipliers are the forces that generate a perfect match between template and target. Usually, finding the exact Lagrange multipliers is difficult and costly in terms of computational effort and therefore approximation methods are often used. Two popular approximation methods are the penalty method and the augmented Lagrange method. In the penalty method, the constraint is enforced loosely by penalizing any deviation from the constraint. This is essentially the method described in the introduction, where the forces are proportional to the mismatch between template and target. Note that in the penalty method there has to be some mismatch in order for the warping force to be non-zero. Although the mismatch can be minimized by increasing the penalty parameter, increasing the penalty too much may destabilize the model and create convergence problems. To circumvent these issues, the warping plugin also implements an augmented Lagrangian formulation of warping where the forces are ramped up during each augmentation.

$$\lambda^{k+1} = \lambda^{k} + F_I\left(\varphi^{k}\right) \tag{1.4}$$

As can be seen from this equation, the Lagrange multipliers are calculated iteratively. In the first iteration, $\lambda^1 = 0$, and the algorithm proceeds as in the penalty method. After this first iteration converges, the multipliers are updated according to (1.4) and the solution is repeated with the updated warping forces. Each iteration is called an augmentation and the process continues until the Lagrange multipliers have converged. The augmented Lagrangian method is often capable of finding a better registration than the penalty method for a given penalty factor. The obvious disadvantage is the need for repeated solutions of the same time step.

The augmented Lagrangian method is controlled by two parameters.

```
<laugon>1</laugon>
<altol>0.1</altol>
```

The *laugon* parameter activates the augmented Lagrangian method (omit this parameter or set it to zero for the penalty method). The *altol* sets the convergence norm for the augmentations.

4. Warping Plot Fields

The warping plugin defines several plot fields that can be written to the FEBio plotfile. The following table shows the available fields.

name	description
warp-template	The template image evaluated on the mesh
warp-target	The target image evaluated on the mesh
warp-energy	The image energy evaluated on the mesh
warp-force	The warping force evaluated on the mesh

These fields are defined in the *plotfile* section of the *Output* section in the FEBio input file. The following example outputs the stress, displacement, and the warping fields to the plot file.

```
<Output>
      <plotfile type="febio">
            <var type="displacement"/>
            <var type="stress"/>
            <var type="warp-template"/>
            <var type="warp-target"/>
            <var type="warp-energy"/>
            <var type="warp-force"/>
      </plotfile>
</Output>
```