Finite Strain Formulation of the Discrete Equilibrium Gap Principle: I-Mechanically Consistent Regularization for Large Motion Tracking

Martin Genet

Finite Strain Formulation of the Discrete Equilibrium Gap Principle: I-Mechanically Consistent Regularization for Large Motion Tracking

Martin Genet
Laboratoire de Mécanique des Solides, École Polytechnique/IPP/CNRS, Palaiseau, France
MΞDISIM Team, INRIA, Palaiseau, France

June 9, 2023

Contents

Abstract **2**

Keywords **2**

1 Introduction **2**

2 Methods **3**
2.1 The motion tracking problem 3
2.2 Short literature review on mechanical regularization 4
2.3 The nonlinear discrete equilibrium gap regularization 5
2.3.1 Body stresses regularization 5
2.3.2 Boundary tractions regularization 6
2.4 Inverse problem reformulation 7
2.5 Numerical resolution 8
2.6 Synthetic data 11

3 Results & Discussion **12**
3.1 Rigid body motion 12
3.2 Nonrigid Homogeneous deformation 19
3.3 Nonrigid Heterogeneous deformation 26
3.4 Impact of mesh size 30

4 Conclusion **32**

Acknowledgements **33**

References **33**

Appendix **37**

A Code for Figures 1, 2, 3 & 4 **37**

B Code for Figure 5 **41**

C Code for Figure 6 **46**

Abstract

The equilibrium gap principle offers a good trade-off between robustness & accuracy for regularizing motion tracking, as it simply enforces that the tracked motion corresponds to a body deforming under arbitrary loadings. This paper introduces an extension of the equilibrium gap principle in the large deformation setting, a novel regularization term to control surface tractions, both in the context of finite element motion tracking, and an inverse problem consistent reformulation of the tracking problem. Tracking performance of the proposed method, with displacement resolution down to the pixel size, is demonstrated on synthetic images representing various motions with various signal-to-noise ratios.

Keywords

Motion tracking; Mechanical regularization; Equilibrium gap principle; Finite element method; Inverse problems.

1 Introduction

Motion tracking is an important field of image processing, with many application domains from experimental mechanics to biomedical engineering. In experimental mechanics, especially in the content of material parameter identification, it induced a true change of paradigm, as it is no longer necessary to perform delicate experiments with simple kinematics like pure tension or compression; instead rather complex experiments can now be performed, involving potentially many deformation mechanisms, as long as the potentially complex kinematics can be tracked based on surface or volume images [Chu et al. 1985; Hild et al. 2006; Lenoir et al. 2007; Tueni et al. 2020]. In biomedical engineering, it allows for the quantitative analysis of biomedical images, hence to derive objective & quantitative biomarkers for improved diagnosis, either directly based on kinematics [Garot et al. 2000; Zou, Xi, et al. 2018], or by merging physical models and imaging data into so called digital twins [Smith et al. 2011; Patte et al. 2022].

Many approaches have been developed over the past decades, based on many variants of the many aspects of the method, such as harmonic/Fourier *vs.* intensity/features tracking, local *vs.* global approaches, *etc.* [Bornert et al. 2009; Hild et al. 2012; Sotiras et al. 2013; Tobon-Gomez et al. 2013]. In this paper, we use an intensity-based global tracking approach, which is the most natural to integrate our novel regularization approach, though other tracking approaches could have been considered as well. Existing intensity-based global approaches differ notably in their motion models (splines, finite elements, *etc.*), image similarity metrics (mean squared error, structural similarity index, mutual information, *etc.*), optimization methods (gradient descent, Gauss-Newton, Newton, *etc.*), *etc.* [Sotiras et al. 2013; Tobon-Gomez et al. 2013].

One key question in motion tracking is the regularization, *i.e.*, the *a priori* knowledge introduced in the process to improve the quality of the tracking. It is required by the intrinsic ill-posedness of the problem (we are looking for a vector field —the displacement— from an input scalar field —the image—), as well as image finite resolution, noise and bias. As in any inverse or optimization problem, efficient regularization requires a fine trade-off, here between providing enough constraint to help the tracking quality & robustness, while providing enough freedom so as to not interfere with the actual motion. In some sense regularization allows for a control, through penalization, of the function space into which the solution is sought. Many regularization terms have been proposed in the literature, such as Laplacian smoothing [Passieux et al. 2012], fluid-like mechanical regularization [Christensen et al. 1996], incompressibility [Mansi et al. 2011], hyperelastic energy [Veress et al. 2005], *etc.*, some of which will be discussed in details in this paper. An optimal trade-off is arguably reached by the so-called equilibrium gap regularization [Hild et al. 2006; Genet, Stoeck, et al. 2018], which puts no direct constraint on the kinematics, while enforcing that the motion is close to a solution of a mechanics problem, in a sense that will be specified later in the paper.

The equilibrium gap principle was originally formulated, at the discrete level and in the linear setting, in the context of material parameter identification based on full-field measurement, in [Claire et al. 2004], and was later used in the context of motion tracking notably in [Leclerc et al. 2010]. An extension to the non linear setting was proposed, at the continuous level, in [Genet, Stoeck, et al. 2018; Lee et al. 2019; Berberoğlu, Stoeck, Moireau, et al. 2019], with multiple applications to biomedical images [Xi et al. 2016; Zou, Leng, et al. 2020; Castellanos et al. 2021]. In this paper, we propose another extension, still in the nonlinear finite strain setting but at the discrete level —hence allowing to better distinguish the equilibrium gaps induced by the motion itself and the finite element discretization, as will be detailed in the paper—, and show that it performs better than all previous formulations.

As already mentioned, and discussed in details later in the paper, the equilibrium gap regularization consists in enforcing that the obtained displacement is close to a solution of a mechanics problem with generic material behavior and arbitrary imposed surface tractions [Leclerc et al. 2010; Genet, Stoeck, et al. 2018]. To actually obtain some regularization of the problem, this arbitrariness must be handled, and the surface tractions must be somehow controlled independently of the discretization of the displacement field [Leclerc et al. 2010]. A surface Laplacian of the displacement was used as an additional regularization term in [Leclerc et al. 2010], which unfortunately does not generalize to the large motion setting. One option would be to use a separate discretization for the displacement and the tractions, which however would represent a significant technical difficulty. Instead, in this paper, we propose an additional regularization term based on the surface gradient of the normal & tangential components of the surface tractions, and show that it performs as expected.

The rest of the paper is organized as follows. We first formulate the general motion tracking problem (Section 2.1), then we provide a short literature review on mechanical regularization (Section 2.2), and we describe our proposed regularization term, including body (Section 2.3.1) & boundary (Section 2.3.2) terms. Then, we give an inverse problem formulation to the regularized motion tracking problem (Section 2.4), and we describe our numerical strategy for the resolution (Section 2.5). We finish the Methods section with a description of the synthetic images that will be used for the validation of our method (Section 2.6), and then analyse tracking Results in the case of rigid motion (Section 3.1), non rigid but homogeneous motion (3.2) and non homogeneous motion (Section 3.3), as well as in the case of refined meshes (Section 3.4).

2 Methods

2.1 The motion tracking problem

Let us start by precising the problem setting and notations. We consider I_0 & I, two images (*i.e.*, image intensity fields) representing the same body $\mathcal{B}$ at two instants t_0 & t:

$$\mathrm{I}_0 : \begin{cases} \square_0 \to \mathbb{R} \\ \underline{\mathrm{X}} \mapsto \mathrm{I}_0\left(\underline{\mathrm{X}}\right) \end{cases}, \qquad \mathrm{I} : \begin{cases} \square \to \mathbb{R} \\ \underline{\mathrm{x}} \mapsto \mathrm{I}\left(\underline{\mathrm{x}}\right) \end{cases}, \tag{1}$$

where $\square_0$ & $\square$ are the image domains at t_0 & t, which are usually identical. The domains occupied by the body $\mathcal{B}$ at t_0 & t are denoted by Ω_0 & ω, respectively. The problem is to find the smooth mapping $\underline{\Phi}$, or equivalently the smooth displacement field $\underline{\mathrm{U}}$, between material points of the reference and deformed domains:

$$\underline{\Phi} : \begin{cases} \Omega_0 \to \omega \\ \underline{\mathrm{X}} \mapsto \underline{\mathrm{x}} = \underline{\Phi}\left(\underline{\mathrm{X}}\right) \end{cases}, \quad \underline{\mathrm{U}} : \begin{cases} \Omega_0 \to \mathbb{R}^3 \\ \underline{\mathrm{X}} \mapsto \underline{\mathrm{U}}\left(\underline{\mathrm{X}}\right) := \underline{\Phi}\left(\underline{\mathrm{X}}\right) - \underline{\mathrm{X}} \end{cases}, \tag{2}$$

where $\underline{\mathrm{X}}$ & $\underline{\mathrm{x}}$ denote the position of a given material point in the reference and deformed configurations, respectively. Due to its intrinsic ill-posedness, the problem is formulated as a regularized minimization problem:

$$\text{Find } \underline{\mathrm{U}^{\mathrm{sol}}} := \mathrm{argmin}_{\{\underline{\mathrm{U}}\}} \left\{ \mathrm{J}\left(\underline{\mathrm{U}}\right) := (1-\beta)\, \frac{\mathrm{J}^{\mathrm{ima}}\left(\underline{\mathrm{U}}\right)}{\mathrm{J}_0^{\mathrm{ima}}} + \beta \frac{\mathrm{J}^{\mathrm{reg}}\left(\underline{\mathrm{U}}\right)}{\mathrm{J}_0^{\mathrm{reg}}} \right\}, \tag{3}$$

where J^{ima} is the image similarity metric, or "correlation energy", J^{reg} is the regularization energy, J_0^{ima} & J_0^{reg} are normalization terms, and β defines the regularization strength. The normalization terms J_0^{ima} & J_0^{reg} allow for the consistent addition of "energies" with very different physical units, and are typically taken equal to the value of J^{ima} & J^{reg} for a chosen displacement field [Leclerc et al. 2010], for instance a plane wave displacement with a period of 10 finite element characteristic lengths and a unit magnitude. The correlation energy is assumed to be convex, at least in the neighborhood of the solution, though it is in general not quadratic.

In image intensity-based global approaches, the following correlation energy is generally used:

$$J^{ima}(\underline{U}) := \frac{1}{2}\int_{\Omega_0} \left(I(\underline{X}+\underline{U}(\underline{X})) - I_0(\underline{X})\right)^2 d\Omega_0. \tag{4}$$

Other metrics have been proposed; however, we retain this one notably because it can be differentiated straightforwardly.

We will employ the finite element method to discretize this problem, such that the displacement field is approximated as $\underline{U}(\underline{X}) \approx {}^t\underline{\underline{\mathbb{N}}}(\underline{X})\cdot\underline{\mathbb{U}}$ with $\underline{\underline{\mathbb{N}}}$ the array of shape functions. Thus, the problem becomes a finite dimensional problem:

$$\text{Find } \underline{\mathbb{U}^{sol}} := \text{argmin}_{\{\underline{\mathbb{U}}\}} \left\{ J(\underline{\mathbb{U}}) := (1-\beta)\frac{J^{ima}(\underline{\mathbb{U}})}{J_0^{ima}} + \beta\frac{J^{reg}(\underline{\mathbb{U}})}{J_0^{reg}} \right\}, \tag{5}$$

where

$$J^{ima}(\underline{\mathbb{U}}) := \frac{1}{2}\int_{\Omega_0} \left(I\left(\underline{X}+{}^t\underline{\underline{\mathbb{N}}}(\underline{X})\cdot\underline{\mathbb{U}}\right) - I_0(\underline{X})\right)^2 d\Omega_0. \tag{6}$$

Note that even without mechanical regularization, finite element discretization introduces some kind of geometrical regularization, as the richness of the approximation space is controlled by the mesh size and shape functions degree.

2.2 Short literature review on mechanical regularization

Many regularization terms have been proposed in the literature, see for instance [Sotiras et al. 2013]. Here we will briefly recall the major classes of proposals with mechanical content. All these approaches require to define a constitutive law, though it does not need to model the actual behavior of the tracked body: generic material laws can be used, and the material stiffness simply controls the strength of the regularization.

Elastic [Miller et al. 1993] & hyperelastic [Veress et al. 2005] regularizations have been proposed, which consist in penalizing the strain energy of the body:

$$J^{reg,el}(\underline{U}) := \int_{\Omega_0} \rho_0 \Psi(\underline{U})\, d\Omega_0, \tag{7}$$

where ρ_0 is the reference mass density & Ψ the reference specific free energy—in practice, generic laws such as Hooke or neohookean are usually used. This was probably inspired by the fact that the elastostatic problem can often be formulated as a minimization problem with the system potential energy that is the sum of the elastic and loading potential energies. However, by only considering the elastic energy beside the image similarity metric, and no specific loading energy, it is implicitly assumed that the only load applied to the body is a body force associated to image dissimilarity, and no boundary tractions. Moreover, this regularization penalizes strain itself, as only rigid body motions have zero elastic energy.

Another, arguably more mechanically consistent, regularization approach is based on the equilibrium gap principle [Claire et al. 2004]. It was formulated in the linear setting, and directly at the discrete level, in [Réthoré et al. 2009; Leclerc et al. 2010]. We first define $\underline{\underline{\mathbb{K}}} := \int_{\Omega_0} \underline{\underline{\mathbb{B}}} : \underline{\underline{\underline{\underline{K}}}} : {}^t\underline{\underline{\mathbb{B}}}\, d\Omega_0$ the system stiffness matrix, with $\underline{\underline{\underline{\underline{K}}}}$ the material stiffness tensor (like for elastic regularization, a

generic isotropic Hooke law is usually used) and $\underline{\underline{\underline{\mathbb{B}}}}$ the array of shape functions symmetric gradients (*i.e.*, such that $\underline{\underline{\varepsilon}}(\underline{X}) := \left(\underline{\underline{\mathrm{Grad}}}(\underline{U})(\underline{X})\right)_{\mathrm{sym}} = \frac{1}{2}\left({}^{\mathrm{t}}\underline{\underline{\mathrm{Grad}}}(\underline{U})(\underline{X}) + \underline{\underline{\mathrm{Grad}}}(\underline{U})(\underline{X})\right) \approx {}^{\mathrm{t}}\underline{\underline{\underline{\mathbb{B}}}}(\underline{X}) \cdot \underline{\mathbb{U}}$). Then the regularization is expressed as

$$\mathrm{J}^{\mathrm{reg,eq,lin}}(\underline{\mathbb{U}}) := \frac{1}{2}\,{}^{\mathrm{t}}\underline{\mathbb{U}} \cdot {}^{\mathrm{t}}\underline{\underline{\mathbb{K}^*}} \cdot \underline{\underline{\mathbb{K}^*}} \cdot \underline{\mathbb{U}}, \tag{8}$$

where $\underline{\underline{\mathbb{K}^*}}$ is a modified system stiffness matrix in which all lines associated to boundary degrees of freedom have been set to 0. The implicit assumption here is that the body is in equilibrium with some arbitrary/unknown boundary tractions (though the smoothness of the surface displacement is usually controlled by an additional surface Laplacian term [Leclerc et al. 2010]) and no body force (though known body forces such as gravity could easily be taken into account). Thus, it does not penalize strain, only deviation from equilibrium, as any equilibrium solution cancels the equilibrium gap "energy".

A first attempt toward extending this principle to the large deformation nonlinear setting has been made in [Genet, Stoeck, et al. 2018]. The main idea is to use a standard finite element discretization of the problem, and use an equivalent norm (because of the stress discontinuity across finite element faces, the divergence of the stress is not defined there, and cannot be directly integrated over the mesh) to characterize the non verification of the internal linear momentum equilibrium equation:

$$\mathrm{J}^{\mathrm{reg,eq,cont}}(\underline{\mathrm{U}}) := \frac{1}{2}\sum_{\mathrm{K}}\int_{\mathrm{K}} \|\underline{\mathrm{Div}}\left(\underline{\underline{\mathrm{P}}}\right)\|^2 \mathrm{dK} + \frac{1}{2\mathrm{h}}\sum_{\mathrm{F}}\int_{\mathrm{F}} \left[\!\left[\underline{\underline{\mathrm{P}}} \cdot \underline{\mathrm{N}}\right]\!\right]^2 \mathrm{dF} \tag{9}$$

where K is the set of finite elements, F the set of interior faces, $\underline{\mathrm{N}}$ the faces normal, h a characteristic length of the finite element discretization, and $\underline{\underline{\mathrm{P}}}$ the first Piola-Kirchhoff stress tensor associated to the displacement field $\underline{\mathrm{U}}$ through the chosen constitutive law. Note that the internal angular momentum equilibrium (${}^{\mathrm{t}}\underline{\underline{\mathrm{P}}} \cdot {}^{\mathrm{t}}\underline{\underline{\mathrm{F}}}^{-1} = \underline{\underline{\mathrm{F}}}^{-1} \cdot \underline{\underline{\mathrm{P}}}$, where $\underline{\underline{\mathrm{F}}}$ is the deformation gradient associated to the displacement field $\underline{\mathrm{U}}$) is usually exactly verified through the constitutive relation. Thus, the underlying assumption is the same as with regularization (8), *i.e.*, that the body is in equilibrium with some arbitrary boundary tractions and no body force. However, the geometrically nonlinear formulation allows to correctly handle large deformations, including large rotations. Nevertheless, one limitation of this approach is that penalization (9) not only contains the equilibrium gap of the considered displacement field, but also the equilibrium gap associated to the finite element discretization. In other terms, a finite element equilibrium solution leads to a nonzero regularization energy. This calls for an improvement of the approach, which is described in the following Section.

2.3 The nonlinear discrete equilibrium gap regularization

2.3.1 Body stresses regularization

In order to avoid the limitation of the regularization term (9), we propose here a novel formulation of the equilibrium gap principle, still in the general context of nonlinear mechanics, but which allows to completely exclude the discretization-induced equilibrium gap from the regularization term. The general idea is still to penalize the non-verification of the internal linear momentum balance ($\underline{\mathrm{Div}}\left(\underline{\underline{\mathrm{P}}}\right) = 0$ in $\boldsymbol{\Omega}_0$, since the body force is neglected for the sake of simplicity, though it could be considered as well). However, after standard finite element discretization, the divergence of the stress tensor is not defined on the element faces (in 3D) or edges (in 2D) thus it is not square integrable, so we define its projection onto a space of square integrable functions, typically a standard continuous finite element space denoted V^{h}:

$$\underline{\boldsymbol{\Pi}_{\mathrm{b}}} \in \mathrm{V}^{\mathrm{h}} \mid \int_{\boldsymbol{\Omega}_0} \underline{\boldsymbol{\Pi}_{\mathrm{b}}} \cdot \underline{\boldsymbol{\Pi}_{\mathrm{b}}^*} = -\int_{\boldsymbol{\Omega}_0} \underline{\mathrm{Div}}\left(\underline{\underline{\mathrm{P}}}\right) \cdot \underline{\boldsymbol{\Pi}_{\mathrm{b}}^*} \quad \forall \underline{\boldsymbol{\Pi}_{\mathrm{b}}^*} \in \mathrm{V}_0^{\mathrm{h}}, \tag{10}$$

where $\mathrm{V}_0^{\mathrm{h}}$ denotes the space of functions of V^{h} that vanish at the boundary, *i.e.*, $\underline{\boldsymbol{\Pi}_{\mathrm{b}}^*}(\partial\boldsymbol{\Omega}_0) = 0$. Indeed, for now we are only interested in body equilibrium gap. Thus, after integration by parts, we obtain:

$$\underline{\boldsymbol{\Pi}_{\mathrm{b}}} \in \mathrm{V}^{\mathrm{h}} \mid \int_{\boldsymbol{\Omega}_0} \underline{\boldsymbol{\Pi}_{\mathrm{b}}} \cdot \underline{\boldsymbol{\Pi}_{\mathrm{b}}^*} = \int_{\boldsymbol{\Omega}_0} \underline{\underline{\mathrm{P}}} : \underline{\underline{\mathrm{Grad}}}(\boldsymbol{\Pi}_{\mathrm{b}}^*) \quad \forall \underline{\boldsymbol{\Pi}_{\mathrm{b}}^*} \in \mathrm{V}_0^{\mathrm{h}} \tag{11}$$

Similar projections, though often performed element-by-element, are used in *a posteriori* error estimation methods, to quantify the distance between the finite element solution and an actual equilibrium solution, *i.e.*, the discretization error [Ladevèze et al. 2005; Zienkiewicz et al. 2013]. However, the objective here is opposite, *i.e.*, we want to discard the discretization error and quantify the intrinsic equilibrium gap of the considered finite element field.

Nevertheless, thanks to the proposed projection, we can actually define the regularization term in a consistent manner:

$$\mathrm{J}_{\mathrm{b}}^{\mathrm{reg,eq}}\left(\underline{\mathrm{U}}\right) := \frac{1}{2}\int_{\Omega_0} \underline{\mathbf{\Pi}_{\mathrm{b}}} \cdot \underline{\mathbf{\Pi}_{\mathrm{b}}} \, \mathrm{d}\Omega_0 \tag{12}$$

A key question is the calculation of this term in the finite element context. Projection (11) simply leads to the following linear system:

$$\underline{\underline{\mathbb{M}}} \cdot \underline{\mathbb{\Pi}_{\mathrm{b}}} = \underline{\mathbb{R}_{\mathrm{b}}} \tag{13}$$

with $\underline{\underline{\mathbb{M}}} := \int_{\Omega_0} \underline{\underline{\mathbb{N}}} \cdot {}^{\mathrm{t}}\underline{\underline{\mathbb{N}}} \, \mathrm{d}\Omega_0$ the mass matrix, $\underline{\mathbb{\Pi}_{\mathrm{b}}}$ such that $\underline{\mathbf{\Pi}_{\mathrm{b}}} = {}^{\mathrm{t}}\underline{\underline{\mathbb{N}}} \cdot \underline{\mathbb{\Pi}_{\mathrm{b}}}$, and

$$\left(\underline{\mathbb{R}_{\mathrm{b}}}\right)_{\mathrm{i}} := \begin{cases} \displaystyle\int_{\Omega_0} \underline{\underline{\mathrm{P}}} : \underline{\underline{\mathrm{Grad}}}\left(\underline{\mathbb{N}_{\mathrm{i}}}\right) \, \mathrm{d}\Omega_0 & \text{if i body d.o.f.} \\ 0 & \text{if i boundary d.o.f.} \end{cases}, \tag{14}$$

where $\underline{\mathbb{N}_{\mathrm{i}}}$ is the (vector) shape function associated to the degree of freedom (d.o.f.) i, *i.e.*, the ith line of the $\underline{\underline{\mathbb{N}}}$ array. Thus, the norm (12) can be expressed as

$$\mathrm{J}_{\mathrm{b}}^{\mathrm{reg,eq}}\left(\underline{\mathbb{U}}\right) = \frac{1}{2}\,{}^{\mathrm{t}}\underline{\mathbb{\Pi}_{\mathrm{b}}} \cdot \underline{\underline{\mathbb{M}}} \cdot \underline{\mathbb{\Pi}_{\mathrm{b}}} = \frac{1}{2}\,{}^{\mathrm{t}}\underline{\mathbb{R}_{\mathrm{b}}} \cdot \underline{\underline{\mathbb{M}}}^{-1} \cdot \underline{\mathbb{R}_{\mathrm{b}}} \tag{15}$$

Note that if we linearize this expression, we obtain $\underline{\mathbb{R}_{\mathrm{b}}} \approx \underline{\underline{\mathbb{K}^*}} \cdot \underline{\mathbb{U}}$ and thus $\mathrm{J}_{\mathrm{b}}^{\mathrm{reg,eq}} \approx \frac{1}{2}\,{}^{\mathrm{t}}\underline{\mathbb{U}} \cdot {}^{\mathrm{t}}\underline{\underline{\mathbb{K}^*}} \cdot \underline{\underline{\mathbb{M}}}^{-1} \cdot \underline{\underline{\mathbb{K}^*}} \cdot \underline{\mathbb{U}}$, *i.e.*, an expression similar to (8), the mass matrix allowing to make the term consistent when refining the mesh.

2.3.2 Boundary tractions regularization

The regularization term proposed in Section 2.3.1 basically enforces that the motion solution corresponds to the motion of a body in equilibrium with some arbitrary tractions applied on its boundary. The arbitrary nature of these tractions can be problematic, especially as the computational mesh is refined and the variations of these tractions is not controlled. In [Leclerc et al. 2010], it was proposed to add a penalization term corresponding to the Laplacian of the boundary displacement, which however does not generalize to the large motion context. Thus, here we propose a new term, consistent with the body term that involves internal stresses directly, which consists in penalizing the surface tractions gradients.

It is important, however, to note that surface tractions can vary intrinsically but also because of the surface curvature. For instance, a simple homogeneous pressure applied onto a curved surface corresponds to a vector that varies in space. To avoid penalizing the surface curvature itself, we propose to penalize the surface gradient of the normal and tangential components of the boundary tractions separately. In 2D, the tangential part is scalar; in 3D we propose to take the norm of the tangential force vector.

Let us start with the regularization term associated to the normal traction, *i.e.*, $\mathrm{F_n}\left(\underline{\mathrm{U}}\right) := {}^{\mathrm{t}}\underline{\mathrm{N}} \cdot \underline{\underline{\mathrm{P}}}\left(\underline{\mathrm{U}}\right) \cdot \underline{\mathrm{N}}$ where $\underline{\mathrm{N}}$ is the body outward normal. The problem is the same as for the equilibrium equation, *i.e.*, the surface gradient of the boundary tractions associated to a standard finite element displacement field is not defined on the boundary elements edges (in 3D) or points (in 2D) thus it is not integrable. Hence we propose to use the same technique, *i.e.*, we first define its projection onto a space of square integrable functions, again a typical finite element space, continuous on the domain boundary and denoted $\partial\mathrm{V}^{\mathrm{h}}$:

$$\underline{\mathbf{\Pi}_{\mathrm{n}}} \in \partial\mathrm{V}^{\mathrm{h}} \mid \int_{\partial\Omega_0} \underline{\mathbf{\Pi}_{\mathrm{n}}} \cdot \underline{\mathbf{\Pi}_{\mathrm{n}}^*} = -\int_{\partial\Omega_0} \underline{\mathrm{Grad_s}}\left(\mathrm{F_n}\right) \cdot \underline{\mathbf{\Pi}_{\mathrm{n}}^*} \quad \forall \underline{\mathbf{\Pi}_{\mathrm{n}}^*} \in \partial\mathrm{V}^{\mathrm{h}}, \tag{16}$$

where $\underline{\mathrm{Grad}_\mathrm{s}}\left(\mathrm{F_n}\right) = \underline{\underline{\mathbf{\Pi}}} \cdot \underline{\mathrm{Grad}}\left(\mathrm{F_n}\right)$ with $\underline{\underline{\mathbf{\Pi}}} := \underline{\underline{1}} - \underline{\mathrm{N}} \otimes \underline{\mathrm{N}}$ the projection operator onto the domain boundary [Brandner et al. 2021]. After integration by parts of the right hand side we obtain:

$$\underline{\mathbf{\Pi}_\mathrm{n}} \in \partial \mathrm{V}^\mathrm{h} \mid \int_{\partial\mathbf{\Omega}_0} \underline{\mathbf{\Pi}_\mathrm{n}} \cdot \underline{\mathbf{\Pi}_\mathrm{n}^*} = \int_{\partial\mathbf{\Omega}_0} \mathrm{F_n} \cdot \mathrm{Div_s}\left(\underline{\mathbf{\Pi}_\mathrm{n}^*}\right) \quad \forall \underline{\mathbf{\Pi}_\mathrm{n}^*} \in \partial \mathrm{V}^\mathrm{h}, \tag{17}$$

where $\mathrm{Div_s}\left(\underline{\mathbf{\Pi}_\mathrm{n}^*}\right) = \mathrm{tr}\left(\underline{\underline{\mathbf{\Pi}}} \cdot \underline{\underline{\mathrm{Grad}}}\left(\underline{\mathbf{\Pi}_\mathrm{n}^*}\right) \cdot \underline{\underline{\mathbf{\Pi}}}\right)$ [Brandner et al. 2021]. Then, the regularization term is actually defined as:

$$\mathrm{J_n^{reg,eq}}\left(\underline{\mathrm{U}}\right) := \frac{1}{2} \int_{\partial\mathbf{\Omega}_0} \underline{\mathbf{\Pi}_\mathrm{n}} \cdot \underline{\mathbf{\Pi}_\mathrm{n}} \, \mathrm{d}\partial\mathbf{\Omega}_0 \tag{18}$$

The discretization procedure is similar to the bulk term:

$$\mathrm{J_n^{reg,eq}}\left(\underline{\mathbb{U}}\right) = \frac{1}{2} \, {}^\mathrm{t}\underline{\mathbb{R}_\mathrm{n}} \cdot \underline{\underline{\mathbb{M}_{\partial\mathbf{\Omega}_0}^{-1}}} \cdot \underline{\mathbb{R}_\mathrm{n}} \tag{19}$$

where $\underline{\underline{\mathbb{M}_{\partial\mathbf{\Omega}_0}}} := \int_{\partial\mathbf{\Omega}_0} \underline{\underline{\mathbb{N}}} \cdot {}^\mathrm{t}\underline{\underline{\mathbb{N}}} \, \mathrm{d}\partial\mathbf{\Omega}_0$ is the "mass" matrix of the domain boundary, and

$$\left(\mathbb{R}_\mathrm{n}\right)_\mathrm{i} := \begin{cases} 0 & \text{if i body d.o.f.} \\ \displaystyle\int_{\partial\mathbf{\Omega}_0} \mathrm{F_n} \cdot \mathrm{Div_s}\left(\underline{\mathbb{N}_\mathrm{i}}\right) \, \mathrm{d}\partial\mathbf{\Omega}_0 & \text{if i boundary d.o.f.} \end{cases}. \tag{20}$$

Regarding the regularization term associated to tangential tractions, in 2D the tangential force is a scalar, defined as $\mathrm{F_{t,2D}}\left(\underline{\mathrm{U}}\right) := {}^\mathrm{t}\underline{\mathrm{T}} \cdot \underline{\underline{\mathrm{P}}}\left(\underline{\mathrm{U}}\right) \cdot \underline{\mathrm{N}}$ where $\underline{\mathrm{T}}$ is the body tangential vector, so it is the same formulation as the normal traction:

$$\mathrm{J_{t,2D}^{reg,eq}}\left(\underline{\mathbb{U}}\right) = \frac{1}{2} \, {}^\mathrm{t}\underline{\mathbb{R}_\mathrm{t,2D}} \cdot \underline{\underline{\mathbb{M}_{\partial\mathbf{\Omega}_0}^{-1}}} \cdot \underline{\mathbb{R}_\mathrm{t,2D}} \tag{21}$$

where

$$\left(\mathbb{R}_\mathrm{t,2D}\right)_\mathrm{i} := \begin{cases} 0 & \text{if i body d.o.f.} \\ \displaystyle\int_{\partial\mathbf{\Omega}_0} \mathrm{F_{t,2D}} \cdot \mathrm{Div_s}\left(\underline{\mathbb{N}_\mathrm{i}}\right) \, \mathrm{d}\partial\mathbf{\Omega}_0 & \text{if i boundary d.o.f.} \end{cases}. \tag{22}$$

In 3D the tangential force is a vector, defined as $\underline{\mathrm{F_{t,3D}}}\left(\underline{\mathrm{U}}\right) := \underline{\underline{\mathbf{\Pi}}} \cdot \left(\underline{\underline{\mathrm{P}}}\left(\underline{\mathrm{U}}\right) \cdot \underline{\mathrm{N}}\right)$, and we propose to simply penalize the gradient of its norm, which leads to:

$$\mathrm{J_{t,3D}^{reg,eq}}\left(\underline{\mathbb{U}}\right) = \frac{1}{2} \, {}^\mathrm{t}\underline{\mathbb{R}_\mathrm{t,3D}} \cdot \underline{\underline{\mathbb{M}_{\partial\mathbf{\Omega}_0}^{-1}}} \cdot \underline{\mathbb{R}_\mathrm{t,3D}} \tag{23}$$

where

$$\left(\mathbb{R}_\mathrm{t,3D}\right)_\mathrm{i} := \begin{cases} 0 & \text{if i body d.o.f.} \\ \displaystyle\int_{\partial\mathbf{\Omega}_0} \|\underline{\mathrm{F_{t,3D}}}\| \cdot \mathrm{Div_s}\left(\underline{\mathbb{N}_\mathrm{i}}\right) \, \mathrm{d}\partial\mathbf{\Omega}_0 & \text{if i boundary d.o.f.} \end{cases}. \tag{24}$$

2.4 Inverse problem reformulation

Because the proposed regularization has a strong mechanical sense, the regularized tracking problem can be reformulated as an inverse problem, where the unknown is the traction field applied on the domain boundary, and which can be formulated as a constrained optimization problem:

$$\text{Find } \underline{\mathrm{T^{sol}}} := \mathrm{argmin}_{\{\underline{\mathrm{T}}\}} \left\{ \mathrm{J}\left(\underline{\mathrm{T}}\right) := (1-\beta) \, \|\mathrm{I} \circ \underline{\mathbf{\Phi}}\left(\underline{\mathrm{T}}\right) - \mathrm{I_0}\|_{\mathbf{\Omega}_0}^2 + \beta \, \|\underline{\mathrm{T}}\|_{\partial\mathbf{\Omega}_0}^2 \right\}, \tag{25}$$

where $\underline{\mathbf{\Phi}}\left(\underline{\mathrm{T}}\right)$ is the mapping solution associated to the traction field $\underline{\mathrm{T}}$, *i.e.*, which verifies constitutive and equilibrium equations, and β is the regularization strength. Norms will be specified later on. The meaning of this formulation is that we search for a traction field which generates a displacement field (through a chosen generic mechanical behavior and standard mechanical equilibrium) that allows to match the two image intensities, the regularization term allowing to control the smoothness of the traction field. In optimization, such constraints can be enforced

strongly through Lagrange multipliers, or approximately through penalization [Allaire 2007]. Here we propose to simply use penalization, such that the problem becomes:

$$\text{Find } \left(\underline{\mathrm{T}}^{\mathrm{sol}}, \underline{\mathrm{U}}^{\mathrm{sol}}\right) := \mathrm{argmin}_{\{\underline{\mathrm{T}},\underline{\mathrm{U}}\}} \Big\{ \mathrm{J}\left(\underline{\mathrm{T}}, \underline{\mathrm{U}}\right) := (1-\beta) \left\| \mathrm{I} \circ \underline{\Phi} - \mathrm{I}_0 \right\|^2_{\Omega_0} + \beta \left\| \underline{\mathrm{T}} \right\|^2_{\partial\Omega_0} + \gamma \left(\left\| \underline{\mathrm{Div}}\left(\underline{\underline{\mathrm{P}}}\right) \right\|^2_{\Omega_0} + \left\| {}^{\mathrm{t}}\underline{\underline{\mathrm{P}}} \cdot {}^{\mathrm{t}}\underline{\underline{\mathrm{F}}}^{-1} - \underline{\underline{\mathrm{F}}}^{-1} \cdot \underline{\underline{\mathrm{P}}} \right\|^2_{\Omega_0} + \left\| \underline{\underline{\mathrm{P}}} \cdot \underline{\mathrm{N}} - \underline{\mathrm{T}} \right\|^2_{\partial\Omega_0} \right) \Big\}, \quad (26)$$

where γ is the penalization coefficient, while the three additional terms represent the balance of linear momentum (without imposed force, though it could be introduced straightforwardly), the balance of angular momentum (which is usually verified exactly thanks to the constitutive framework), and the balance with applied tractions, respectively. In this formulation, $\underline{\Phi}$, $\underline{\underline{\mathrm{F}}}$ & $\underline{\underline{\mathrm{P}}}$ are the mapping, deformation gradient & first Piola-Kirchhoff stress tensor associated to the displacement field $\underline{\mathrm{U}}$. Again, the norms will be specified later on. To simplify the formulation, we can remove the traction field variable by enforcing strongly the last balance term, *i.e.*, $\underline{\mathrm{T}} = \underline{\underline{\mathrm{P}}}\left(\underline{\mathrm{U}}\right) \cdot \underline{\mathrm{N}}$, leading to the following formulation:

$$\text{Find } \underline{\mathrm{U}}^{\mathrm{sol}} := \mathrm{argmin}_{\{\underline{\mathrm{U}}\}} \Big\{ \mathrm{J}\left(\underline{\mathrm{U}}\right) := (1-\beta) \left\| \mathrm{I} \circ \underline{\Phi} - \mathrm{I}_0 \right\|^2_{\Omega_0} + \beta \left\| \underline{\underline{\mathrm{P}}} \cdot \underline{\mathrm{N}} \right\|^2_{\partial\Omega_0} + \gamma \left\| \underline{\mathrm{Div}}\left(\underline{\underline{\mathrm{P}}}\right) \right\|^2_{\Omega_0} \Big\}. \quad (27)$$

This formulation corresponds formally (*i.e.*, using the right norms) to the original formulation (3), when using $\mathrm{J}^{\mathrm{reg,eq}}_{\mathrm{b}}$ for the bulk regularization and $\mathrm{J}^{\mathrm{reg,eq}}_{\mathrm{n/t}}$ for the surface terms. This gives another point of view on each term, *i.e.*, that the bulk term can be seen as a penalization term, while the actual regularization comes from the surface terms. Also, it is important to notice that this reformulation only makes sense for "equilibrium gap" regularization terms; plugging instead the elastic regularization terms would lead to an equivalent inverse problem without proper physical meaning, where notably the deformation is driven by a body force generated by the image mismatch.

2.5 Numerical resolution

After describing existing and new regularization terms, we recall that the general motion tracking problem formulation was given by Equation (5), with the image correlation term $\mathrm{J}^{\mathrm{ima}}$ given by Equation (6), and where the regularization term $\mathrm{J}^{\mathrm{reg}}$ may take the form $\mathrm{J}^{\mathrm{reg,el}}$ (Equation (7)), $\mathrm{J}^{\mathrm{reg,eq,cont}}$ (Equation (9)), $\mathrm{J}^{\mathrm{reg,eq}}_{\mathrm{b}}$ (Equation (12)), $\mathrm{J}^{\mathrm{reg,eq}}_{\mathrm{n}}$ (Equation (18)), $\mathrm{J}^{\mathrm{reg,eq}}_{\mathrm{t}}$ (Equation (22) in 2D, (24) in 3D), or combinations of such terms. Indeed, as already discussed, it is often necessary to combine bulk and boundary terms. In such cases one might want to use different weights for each regularization term [Leclerc et al. 2010]; however, to simplify the analysis of the performance of the various terms considered here, we propose to simply sum them, such that they have the same weight, and there is only one parameter controlling the regularization strength, namely β.

Minimization problem (5) is actually formulated as a root finding problem:

$$\underline{\mathbb{U}} \mid \underline{\nabla \mathbb{J}}\left(\underline{\mathbb{U}}\right) = (1-\beta)\, \underline{\nabla \mathbb{J}^{\mathrm{ima}}}\left(\underline{\mathbb{U}}\right) + \beta \underline{\nabla \mathbb{J}^{\mathrm{reg}}}\left(\underline{\mathbb{U}}\right) = 0 \quad (28)$$

where the gradient of the image term is simply:

$$\underline{\nabla \mathbb{J}^{\mathrm{ima}}}\left(\underline{\mathbb{U}}\right) := \int_{\Omega_0} \left(\mathrm{I}\left(\underline{\mathrm{X}} + {}^{\mathrm{t}}\underline{\underline{\mathbb{N}}} \cdot \underline{\mathbb{U}}\right) - \mathrm{I}_0 \right) \frac{\partial \mathrm{I}}{\partial \underline{\mathrm{X}}} \left(\underline{\mathrm{X}} + {}^{\mathrm{t}}\underline{\underline{\mathbb{N}}} \cdot \underline{\mathbb{U}}\right) \cdot \underline{\underline{\mathbb{N}}}\, \mathrm{d}\Omega_0, \quad (29)$$

and the gradients of the proposed regularization terms are:

$$\underline{\nabla \mathbb{J}^{\mathrm{reg,eq}}_{\mathrm{b}}}\left(\underline{\mathbb{U}}\right) := {}^{\mathrm{t}}\underline{\underline{\mathrm{d}\mathbb{R}_{\mathrm{b}}}}\left(\underline{\mathbb{U}}\right) \cdot \underline{\underline{\mathbb{M}}}^{-1} \cdot \underline{\mathbb{R}_{\mathrm{b}}}\left(\underline{\mathbb{U}}\right) \quad (30)$$

with

$$\left(\mathrm{d}\mathbb{R}_{\mathrm{b}}\right)_{\mathrm{ij}} := \begin{cases} \displaystyle\int_{\Omega_0} \underline{\underline{\mathrm{Grad}}}\left(\underline{\mathbb{N}_{\mathrm{i}}}\right) : \frac{\partial \underline{\underline{\mathrm{P}}}}{\partial \underline{\underline{\mathrm{F}}}} : \underline{\underline{\mathrm{Grad}}}\left(\underline{\mathbb{N}_{\mathrm{j}}}\right)\, \mathrm{d}\Omega_0 & \text{if i body d.o.f.} \\ 0 & \text{if i boundary d.o.f.} \end{cases}, \quad (31)$$

and

$$\underline{\nabla \mathbb{J}^{\mathrm{reg,eq}}_{\mathrm{n/t}}}\left(\underline{\mathbb{U}}\right) := {}^{\mathrm{t}}\underline{\underline{\mathrm{d}\mathbb{R}_{\mathrm{n/t}}}}\left(\underline{\mathbb{U}}\right) \cdot \underline{\underline{\mathbb{M}_{\partial\Omega_0}}}^{-1} \cdot \underline{\mathbb{R}_{\mathrm{n/t}}}\left(\underline{\mathbb{U}}\right) \quad (32)$$

with

$$\left(\mathrm{d}\mathbb{R}_{\mathrm{n/t}}\right)_{\mathrm{ij}} := \begin{cases} 0 & \text{if i body d.o.f.} \\ \displaystyle\int_{\partial\Omega_0} \mathrm{Div_s}\left(\underline{\mathbb{N}_\mathrm{i}}\right) \frac{\partial \mathrm{F_{n/t}}}{\partial \underline{\underline{\mathrm{F}}}} : \underline{\underline{\mathrm{Grad}}}\left(\underline{\mathbb{N}_\mathrm{j}}\right) \mathrm{d}\partial\Omega_0 & \text{if i boundary d.o.f.} \end{cases}. \tag{33}$$

We propose to solve this problem using a Gauss-Newton method. Details were given in [Genet, Stoeck, et al. 2018] in the context of the regularization term $\mathrm{J^{reg,eq,cont}}$ (Equation (9)). The key point is that the Jacobian associated to the image term only contains the image gradient product term, not the image hessian term which usually degrades the convergence due to the double derivative of the noise; in principle the second derivation of the regularization terms could be computed exactly, however it is tedious and not necessary, so we employ the same approach as for the image term. Thus, at each Newton iteration we solve the following linear system:

$$\underline{\Delta\mathbb{U}} \mid \left((1-\beta)\,\underline{\underline{\nabla\nabla\mathbb{J}^{\mathrm{ima}}}} + \beta\underline{\underline{\nabla\nabla\mathbb{J}^{\mathrm{reg}}}}\right)\cdot\underline{\Delta\mathbb{U}} = -\left((1-\beta)\,\underline{\nabla\mathbb{J}^{\mathrm{ima}}} + \beta\underline{\nabla\mathbb{J}^{\mathrm{reg}}}\right), \tag{34}$$

where

$$\underline{\underline{\nabla\nabla\mathbb{J}^{\mathrm{ima}}}} := \int_{\Omega_0} \underline{\underline{\mathbb{N}}}\cdot\left(\frac{\partial \mathrm{I}}{\partial \underline{\mathrm{x}}}\cdot{}^{\mathrm{t}}\frac{\partial \mathrm{I}}{\partial \underline{\mathrm{x}}}\right)\cdot{}^{\mathrm{t}}\underline{\underline{\mathbb{N}}}\,\mathrm{d}\Omega_0, \tag{35}$$

which corresponds to a mass matrix weighted by the deformed image gradients, and

$$\underline{\underline{\nabla\nabla\mathbb{J}^{\mathrm{reg,eq}}_{\mathrm{b/n/t}}}} := {}^{\mathrm{t}}\underline{\underline{\mathrm{d}\mathbb{R}_{\mathrm{b/n/t}}}}\cdot\underline{\underline{\mathbb{M}_{\Omega_0/\partial\Omega_0}}}^{-1}\cdot\underline{\underline{\mathrm{d}\mathbb{R}_{\mathrm{b/n/t}}}}. \tag{36}$$

As a stopping criterion for the Newton iterations, we propose to simply use the relative displacement, which is well adapted for the motion tracking problem.

Nevertheless, in practice, we found that such Gauss-Newton iterations do not always converge toward a solution for such a highly stiff problem. Indeed, they often lead to inverted elements, for which the mechanical model cannot be evaluated—this is a situation the regularization is supposed to prevent through the use of a proper energy potential with an infinite energy barrier (for instance, the Ogden-Ciarlet-Geymonat has a $-\ln(\mathrm{J})$ term, where J denotes the volume change [Ogden 1972; Ciarlet et al. 1982]), but the Newton iterations can sometimes pass this energy barrier (interestingly, it is possible, and it is actually the case for the Ogden-Ciarlet-Geymonat potential, that even though the potential is only defined for non-inverted elements, the expression of its derivative is well defined for inverted elements). This is a common issue in large deformation computation, which is usually solved using adaptation time stepping [Le Tallec 1994; Genet 2019]; however, it is not possible here because the "time" increment is controlled by the image temporal discretization. Hence, we augment the Newton iterations with a backtracking line search [Press et al. 2007] that prevents the passing of energy barriers.

Finally, the image series integrator is described in Algorithm 1, the nonlinear solver in Algorithm 2, and the line search in Algorithm 3.

Initialisation
  Read initial image I_0
  Instantiate finite element solution: $\underline{\mathbb{U}} \leftarrow 0$
foreach frame index $\mathrm{t} = 1, 2, \ldots$ **do**
  Read current image $\mathrm{I_t}$
  Compute current displacement: $\underline{\mathbb{U}} \leftarrow$ `nonlinear_solver` $(\mathrm{I}_0, \mathrm{I_t})$
end

Algorithm 1: Image series integrator. The `nonlinear_solver` is detailed Algorithm 2. Note that in this basic version, the Newton iterations at frame t are naturally initialized with the converged solution at frame $\mathrm{t}-1$. A multi-level version is presented in Algorithm 4.

The richness of the displacement solution space is controlled by the finite element mesh and interpolation degree. When using a coarse mesh, with many pixels per element, the mesh itself acts

while err $>$ tol **do**
  Assemble residual, Jacobian: $\underline{\nabla \mathbb{J}}$, $\underline{\underline{\nabla \nabla \mathbb{J}}}$
  Compute solution increment: $\underline{\Delta \mathbb{U}} \leftarrow$ `linear_solver` $\left(\underline{\nabla \mathbb{J}}, \underline{\underline{\nabla \nabla \mathbb{J}}}\right)$
  Update error: err $\leftarrow \frac{\|\underline{\Delta \mathbb{U}}\|}{\|\underline{\mathbb{U}}\|}$
  Compute relaxation: $\alpha \leftarrow$ `line_search` $(\underline{\mathbb{U}}, \underline{\Delta \mathbb{U}})$
  Update solution: $\underline{\mathbb{U}} \leftarrow \underline{\mathbb{U}} + \alpha\, \underline{\Delta \mathbb{U}}$
end

Algorithm 2: Nonlinear solver. In practice we use a tolerance of tol $= 1\%$.

Initialisation
  Compute initial energy: $J_0 \leftarrow J(\underline{\mathbb{U}})$
  Initialise relaxation counter: $k \leftarrow 0$
while $J > J_0$ **do**
  Define current relaxation: $\alpha \leftarrow f^k$
  Compute current energy: $J \leftarrow J(\underline{\mathbb{U}} + \alpha\, \underline{\Delta \mathbb{U}})$
  Update relaxation counter: $k \leftarrow k + 1$
end

Algorithm 3: Backtracking line search. In practice we use a backtracking factor of $f = 0.5$.

as some kind of geometrical regularization, limiting the size of the displacement solution space. Conversely, when solving for very fine meshes, for instance when elements reach the size of the image pixels, convergence becomes problematic as the displacement solution space becomes huge and there is very little information per element; in this case mechanical regularization compensates for the lack of information, but it might not be enough to obtain a robust convergence of the nonlinear iterations. Thus, multi-resolution is often necessary to perform motion tracking on very fine meshes [Bornert et al. 2009; Leclerc et al. 2010], which consists in performing tracking on successively refined meshes, initiating the nonlinear iterations of a given frame and given refinement level by the converged solution obtained at the same frame but at the previous refinement level, instead of the converged solution obtained at the same refinement level but at the previous frame, as it is naturally the case in the single-refinement-level Algorithm 1. Since two triangulations of the same geometric domain might not overlap, for instance if the domain has a curved boundary, and especially if the triangulations have different characteristic sizes, one cannot simply interpolate the displacement field from the coarse grid to the fine one. Instead we propose to use a projection. The multi-resolution frame integrator is detailed in Algorithm 4.

foreach refinement level $k = 1, 2, \ldots$ **do**
  Initialisation
    Read initial image I_0
    Instantiate finite element solution $\underline{\mathbb{U}^k}$
  foreach frame index $t = 1, 2, \ldots$ **do**
    Read current image I_t
    if $k > 1$ **then**
      Initialize displacement: $\underline{\mathbb{U}^k} \leftarrow \underline{\mathbb{U}_t^{k-1}}$
    Compute current displacement: $\underline{\mathbb{U}^k} \leftarrow$ `nonlinear_solver` (I_0, I_t)
    Save displacement: $\underline{\mathbb{U}_t^k} \leftarrow \underline{\mathbb{U}^k}$
  end
end

Algorithm 4: Image series integrator with multi-resolution. Compared to the single-refinement-level Algorithm 1, here the Newton iterations at frame t of refinement level k are initialized with the converged solution at frame t of refinement level $k-1$.

These algorithms have been implemented in an open-source library [Genet 2023a] written in python

and based on the FEniCS [Logg et al. 2012; Alnæs et al. 2015] and VTK [Schroeder et al. 2006] libraries. It is currently freely available online at `https://gitlab.inria.fr/mgenet/dolfin_warp`. We also provide the code to reproduce the results of this paper under the form of jupyter notebooks [Genet 2023b]: static versions are given in the appendix of the paper while interactive versions are currently available online at `https://mgenet.gitlabpages.inria.fr/N-DEG-paper-demos/index.html`.

2.6 Synthetic data

In order to establish the tracking performance of the proposed method, we generated synthetic images corresponding to various objects (simple square, cardiac-like ring) and motions (rigid translation & rotation, homogeneous compression & shear, cardiac-like contraction & twist) with various noise levels. To focus on the regularization term itself, we limited ourselves to highly resolved (though the impact of image resolution could be investigated as well [Berberoğlu, Stoeck, Kozerke, et al. 2022]), and textured (with used a tagged-MRI-like pattern, see [Rutz et al. 2008]) images. The images occupy the spatial domain $[0;1]^2$ (arbitrary unit), the temporal domain $[0;1]$ (arbitrary unit), are discretized with 100×100 pixels spatially and 21 frames temporally.

For the simple (analytical) motion of the square, we define the initial domain as $\Omega_0^{\mathrm{t}} := [0.1;0.7] \times [0.2;0.8]$ for the translation case and $\Omega_0^{\mathrm{r,c,s}} := [0.2;0.8]^2$ for the rotation, compression & shear cases. The motion model are given by

$$\begin{cases} \underline{\mathrm{x}}^{\mathrm{t}}(\underline{\mathrm{X}},\mathrm{t}) := \underline{\mathrm{X}} + \mathrm{t}\,\underline{\mathrm{D}} & \text{with } \underline{\mathrm{D}} := \begin{pmatrix} 0.2 \\ 0 \end{pmatrix} \\ \underline{\mathrm{x}}^{\mathrm{r}}(\underline{\mathrm{X}},\mathrm{t}) := \underline{\mathrm{X}}_0 + \underline{\underline{\mathrm{R}}} \cdot (\underline{\mathrm{X}} - \underline{\mathrm{X}}_0) & \text{with } \underline{\mathrm{X}}_0 := \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix},\ \underline{\underline{\mathrm{R}}} := \begin{pmatrix} +\cos(\theta) & -\sin(\theta) \\ +\sin(\theta) & +\cos(\theta) \end{pmatrix},\ \theta = \frac{\mathrm{t}\pi}{4} \\ \underline{\mathrm{x}}^{\mathrm{c}}(\underline{\mathrm{X}},\mathrm{t}) := \underline{\mathrm{X}}_0 + \underline{\underline{\mathrm{F}}}^{\mathrm{c}} \cdot (\underline{\mathrm{X}} - \underline{\mathrm{X}}_0) & \text{with } \underline{\mathrm{X}}_0 := \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix},\ \underline{\underline{\mathrm{F}}}^{\mathrm{c}} := \begin{pmatrix} \sqrt{1-2\mathrm{t}0.2} & 0 \\ 0 & 1 \end{pmatrix} \\ \underline{\mathrm{x}}^{\mathrm{s}}(\underline{\mathrm{X}},\mathrm{t}) := \underline{\mathrm{X}}_0 + \underline{\underline{\mathrm{F}}}^{\mathrm{s}} \cdot (\underline{\mathrm{X}} - \underline{\mathrm{X}}_0) & \text{with } \underline{\mathrm{X}}_0 := \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix},\ \underline{\underline{\mathrm{F}}}^{\mathrm{s}} := \begin{pmatrix} 1 & 0.2 \\ 0 & 1 \end{pmatrix} \end{cases}, \tag{37}$$

which represent a 0.2 (arbitrary unit) translation, a $\frac{\pi}{4}$ rad rotation, a 20% compression and a 20% shear, respectively.

Then, for each case, the image generation consisted in the following steps: for each frame (time t), for each pixel (position $\underline{\mathrm{x}}$), the initial (*i.e.*, first time frame) position of the pixel ($\underline{\mathrm{X}}(\underline{\mathrm{x}},\mathrm{t})$) was determined through inversion of the mappings (37) and the intensity was given by

$$\mathrm{I}(\underline{\mathrm{x}},\mathrm{t}) := \mathrm{I}_0(\underline{\mathrm{X}}(\underline{\mathrm{x}},\mathrm{t})) \quad \text{with} \quad \mathrm{I}_0(\underline{\mathrm{X}}) := \sqrt{|\sin\left(\frac{\pi \mathrm{X}}{\mathrm{s}}\right)||\sin\left(\frac{\pi \mathrm{Y}}{\mathrm{s}}\right)|} \tag{38}$$

Note that a more complex imaging model [Berberoğlu, Stoeck, Kozerke, et al. 2022] could have been considered as well.

For the cardiac-like case, a ring(center $\underline{\mathrm{X}}_0 := \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$, internal radius 0.2, external radius 0.4 (arbitrary units)) was defined, and an hyperelastic (neo-Hookean & Ogden-Ciarlet-Geymonat potentials with unit Young modulus and 0.3 Poisson coefficient) finite element model with very fine mesh (element size equal to the image pixel size) was run with prescribed displacement applied to the internal (inward displacement of 0.1 and rotation of $-\frac{\pi}{4}$ rad) and external (inward displacement of 0.05 and rotation of $-\frac{\pi}{8}$ rad) edges, mimicking the in-plane motion of a cardiac slice. And to generate the images, we employed the following approach: for each time frame (time t) the reference mesh was warped by applying the computed displacement field, and the displacement field was projected onto the image, such that for each pixel (position $\underline{\mathrm{x}}$) the reference position of the pixel was computed as $\underline{\mathrm{X}}(\underline{\mathrm{x}},\mathrm{t}) = \underline{\mathrm{x}} - \underline{\mathrm{U}}(\underline{\mathrm{x}},\mathrm{t})$. Finally, the texture model (38) was applied.

Noise was eventually added to the images, characterized by the signal-to-noise ratio (SNR). The magnitude of the signal here is 1, and we added random Gaussian noise with zero mean and standard deviation of 0.1, 0.2 & 0.3, corresponding to SNR of 10, 5 & 3.3, respectively. Note that, like for the imaging model, more complex noise models, for instance including spatial and/or temporal correlations [Berberoğlu, Stoeck, Kozerke, et al. 2022], could have been considered.

3 Results & Discussion

To establish the tracking performance of the proposed method (described in Section 2.3), we now present tracking results on various synthetic images (described in Section 2.6), for various regularization terms, namely elastic (Equation (7), continuous version of the equilibrium gap (Equation (9)), discrete version of the equilibrium gap (Equation (12)) and discrete version of the nonlinear equilibrium gap including surface traction regularization terms (Equations (12), (18), (22) and/or (24))). For each regularization term, we considered both a small strain approximation with a Hooke strain energy potential, and a large strain formulation with the neo-Hookean & Ogden-Ciarlet-Geymonat potentials. For all models, we considered a unit Young modulus and null Poisson ratio.

The tracking performance is evaluated in terms of a normalized tracking error defined as

$$\text{err} := \frac{\sqrt{\frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} \frac{1}{|\Omega_0|}\int_{\Omega_0} \|\underline{\mathrm{U}} - \underline{\mathrm{U}^{\mathrm{ex}}}\|^2}}{\sqrt{\frac{1}{\mathrm{T}}\int_0^{\mathrm{T}} \frac{1}{|\Omega_0|}\int_{\Omega_0} \|\underline{\mathrm{U}^{\mathrm{ex}}}\|^2}}, \tag{39}$$

where $\underline{\mathrm{U}}$ is the tracked displacement & $\underline{\mathrm{U}^{\mathrm{ex}}}$ is the ground truth used to generate the images.

3.1 Rigid body motion

We start with simple rigid motions, namely pure translation and pure rotation. Figures 1 & 2 show normalized tracking error as a function of regularization strength $\beta \in [0; 1]$, for various levels of image SNR and for various regularization terms. In the plots, dots represent noise realizations, and lines represent their average.

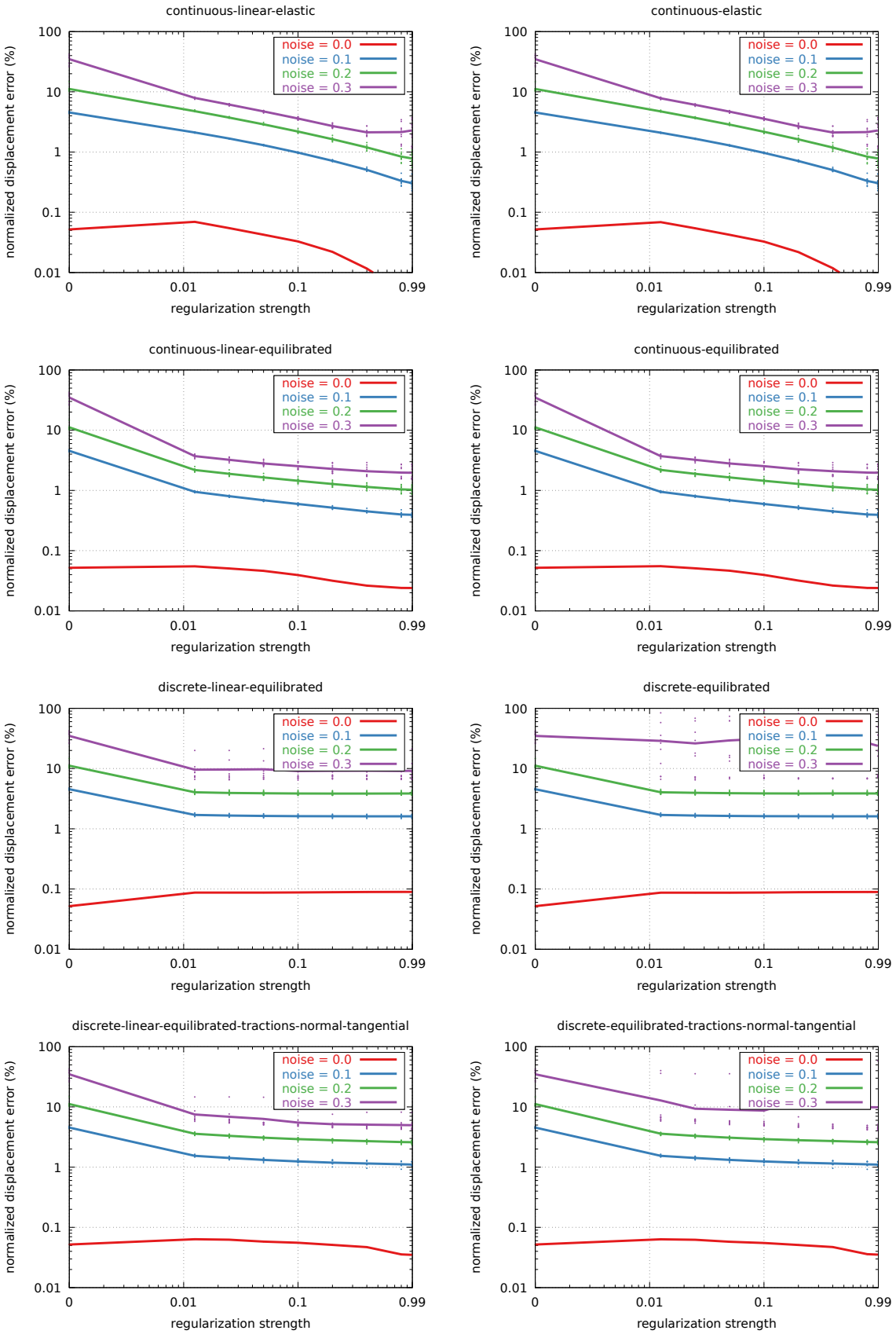
Let us first discuss the pure translation case, *i.e.*, Figure 1. For noiseless images, the tracking error is below 0.1% for all regularization terms and all regularization strengths. Increasing noise level leads to an increase of both the tracking error mean and dispersion. For all considered regularization terms, increasing the regularization strength decreases the mean error, though the dispersion is little impacted. Basically, for such a simple motion, all regularization terms perform rather well. This is explained by the fact that a rigid translation cancels all considered regularization terms, allowing to filter noise-induced spurious motions without interfering with the tracking itself.

One can see that elastic (first row in Figure 1a) and continuous equilibrium gap (second row in Figure 1a) terms behave better than the discrete equilibrium gap terms (third & fourth rows in Figure 1a). This is due to the fact that these terms basically prevent the mesh from deforming, or from deforming in a non affine way, which is compatible with the exact solution here, hence the good tracking performance. However, as we will see later, this constraint will prove problematic for more complex motions.

Focusing on the regularization term introduced in this paper (third & fourth rows in Figure 1a), one can see that the error dispersion becomes significant for images with low SNR. This is due to the fact that these terms represent a much lighter constraint on the displacement field, solely enforcing that it is close to an equilibrium solution; this is, however, the very reason why it performs well on basically any motion as we will see later. Nevertheless, one can also see that adding the boundary traction terms (fourth row in Figure 1a), which penalizes the non smoothness of the normal and tangential tractions at the edges, helps decreasing the tracking error mean and dispersion compared to bulk terms only (third row in Figure 1a).

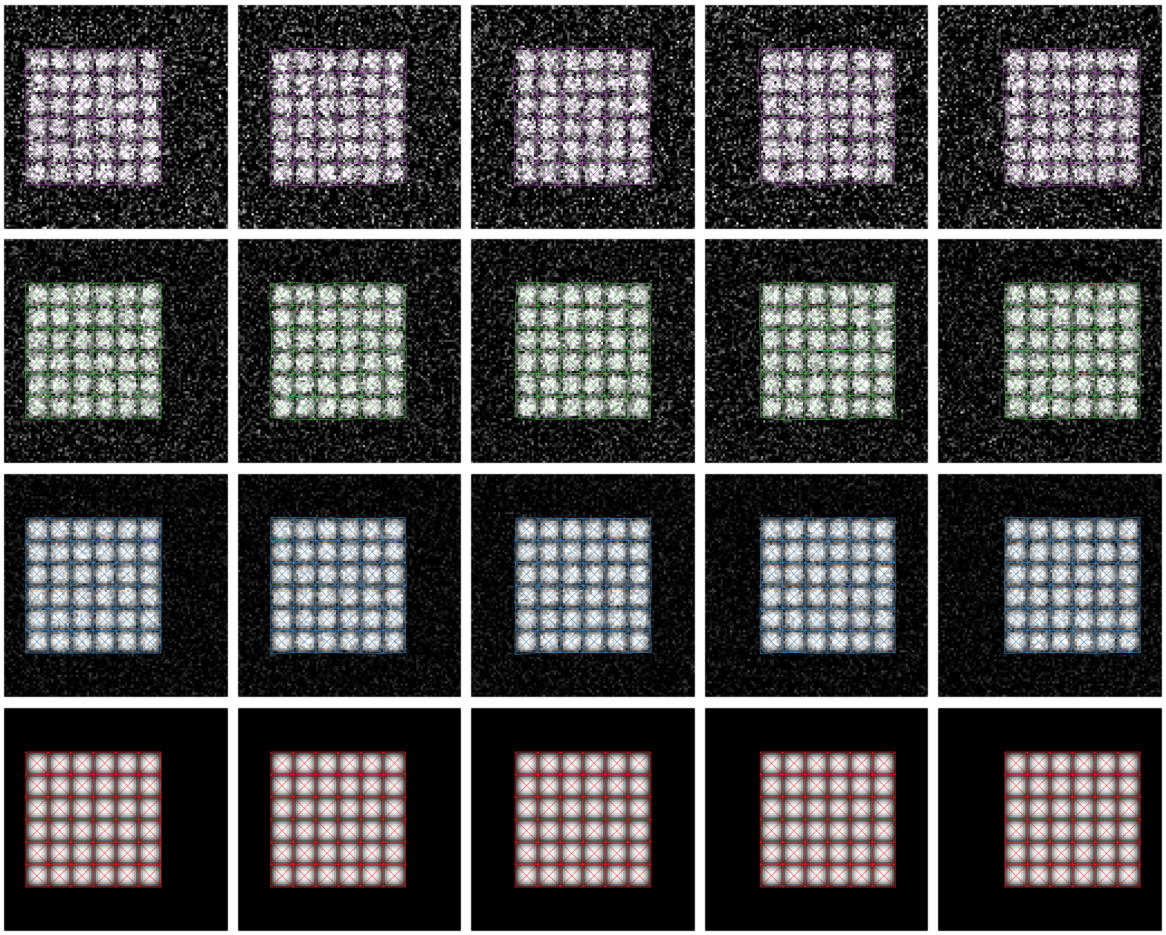
Let us now discuss the pure rotation case, *i.e.*, Figure 2. For the nonlinear elastic (upper right in Figure 2a) and all equilibrated (second, third & fourth rows in Figure 2a) regularization terms, the conclusions are the same as for the pure translation case. For the linear elastic regularization term (upper left in Figure 2a), however, the result is quite different. Indeed, if increasing slightly the regularization strength allows to decrease the tracking error, increasing it further completely degrades the tracking. This is due to the fact that finite rotations generate nonzero infinitesimal

strains, thus nonzero elastic energy; hence the algorithm, which tries to minimize this energy, tends to underestimate rotations (see Figure 2c). This is not the case for hyperelastic regularization (upper right in Figure 2a), as in the finite strain setting, rigid rotations do not generate any strain. Interestingly, this is also not the case for other regularization terms with linearized kinematics and behaviors (left column, second to fourth rows in Figure 2a), because here the spurious strain induced by the finite rotation is homogeneous, hence the associated stress is also homogeneous and thus equilibrated, *i.e.*, it does not generate any spurious equilibrium gap.



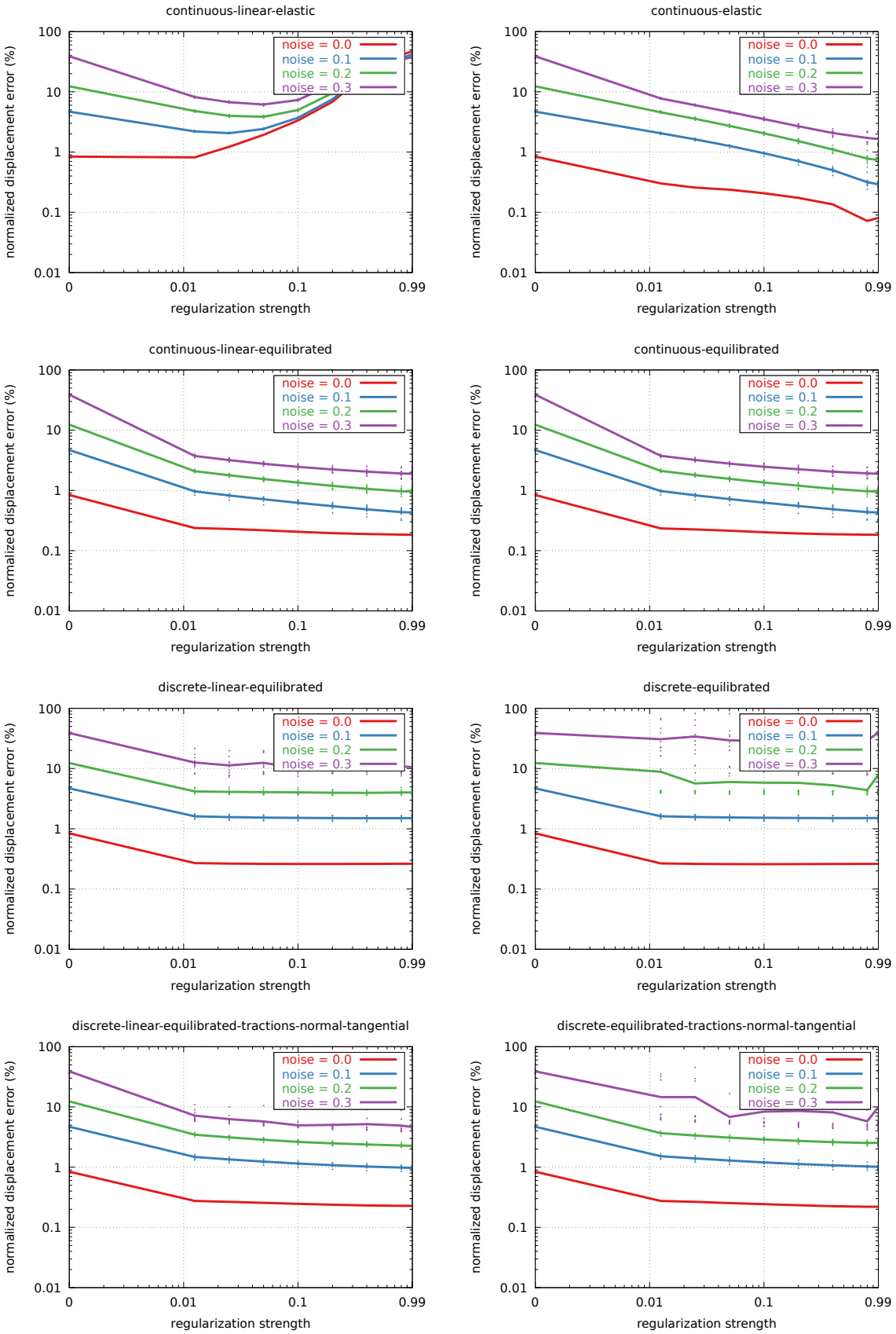
(a) Normalized displacement error (39) as a function of regularization strength (parameter β), for various levels of image noise, for various regularization terms (first row: elastic (7); second row: continuous version of the equilibrium gap (9); third row: discrete version of the equilibrium gap (12); fourth row: discrete version of the equilibrium gap including surface traction regularization (18) & (22)), and for various constitutive laws (left column: small strain, Hooke law; right column: large strain, neo-Hookean & Ogden-Ciarlet-Geymonat law). For such a simple motion, basically all regularization terms allow to reduce the tracking error.

Figure 1: Translation case.



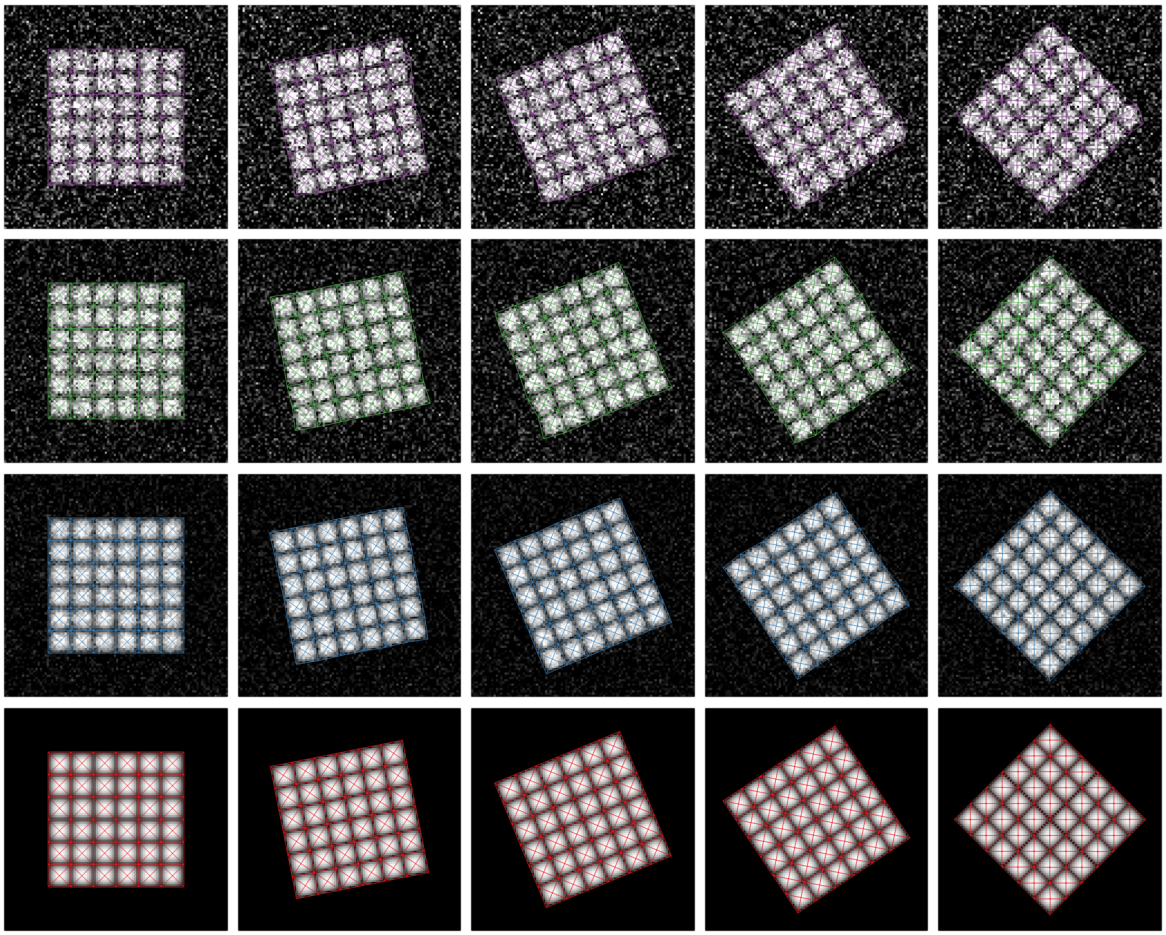
(b) Tracking solutions for the discrete version of the nonlinear equilibrium gap (12) including surface traction regularization (18) & (22), with regularization strength $\beta = 0.1$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). Tracking is visually satisfying for all noise levels.

Figure 1: Translation case.



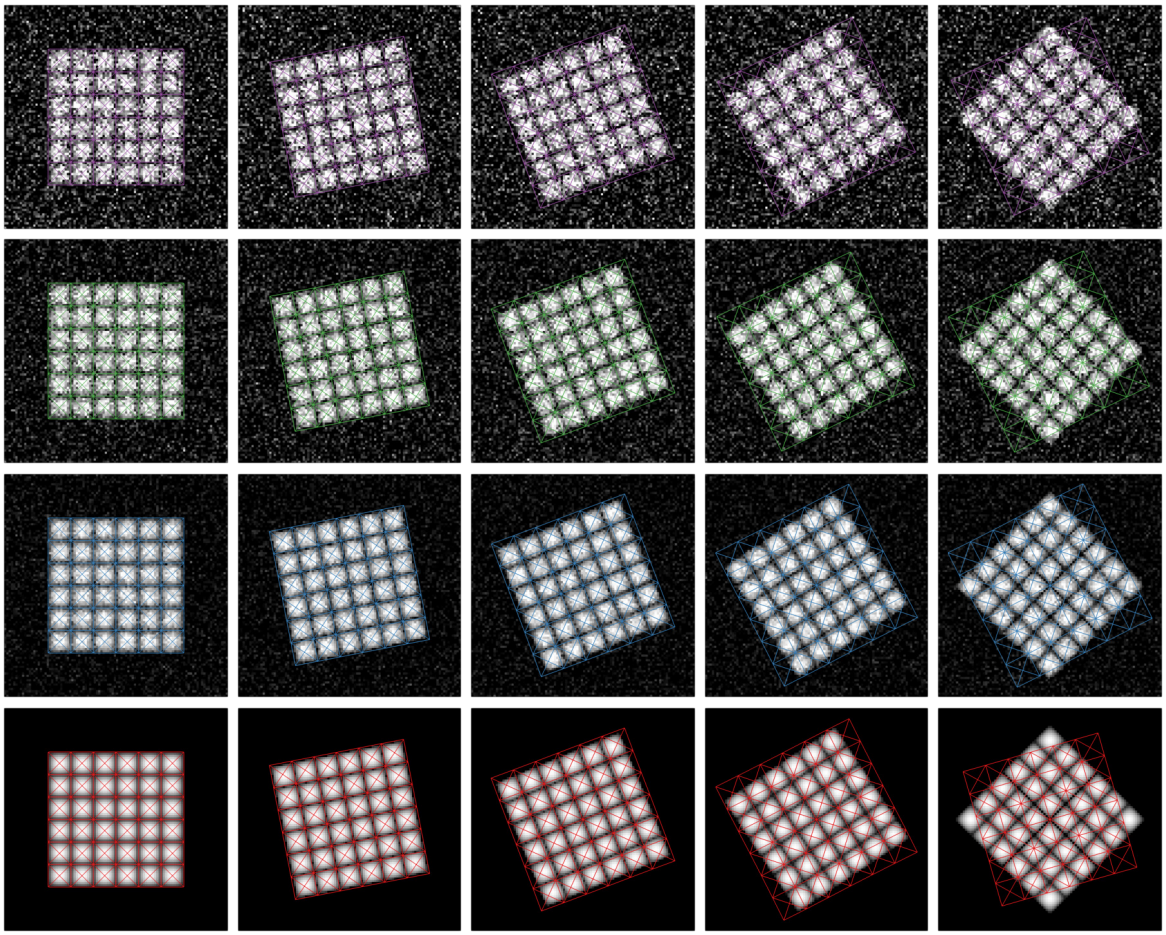
(a) Normalized displacement error (39) as a function of regularization strength (parameter β), for various levels of image noise, for various regularization terms (first row: elastic (7); second row: continuous version of the equilibrium gap (9); third row: discrete version of the equilibrium gap (12); fourth row: discrete version of the equilibrium gap including surface traction regularization (18) & (22)), and for various constitutive laws (left column: small strain, Hooke law; right column: large strain, neo-Hookean & Ogden-Ciarlet-Geymonat law). The linear elastic regularization term interferes with the tracking because finite rotation lead to nonzero infinitesimal strain, thus generating spurious elastic energy.

Figure 2: Rotation case.



(b) Tracking solutions for the discrete version of the nonlinear equilibrium gap (12) including surface traction regularization (18) & (22), with regularization strength $\beta = 0.1$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). Tracking is visually satisfying for all noise levels.

Figure 2: Rotation case.



(c) Tracking solutions for the linear elastic regularization term (7), with regularization strength $\beta = 0.8$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). The linear elastic regularization term tends to reduce rotations because they generate spurious elastic energy, which degrades the tracking.

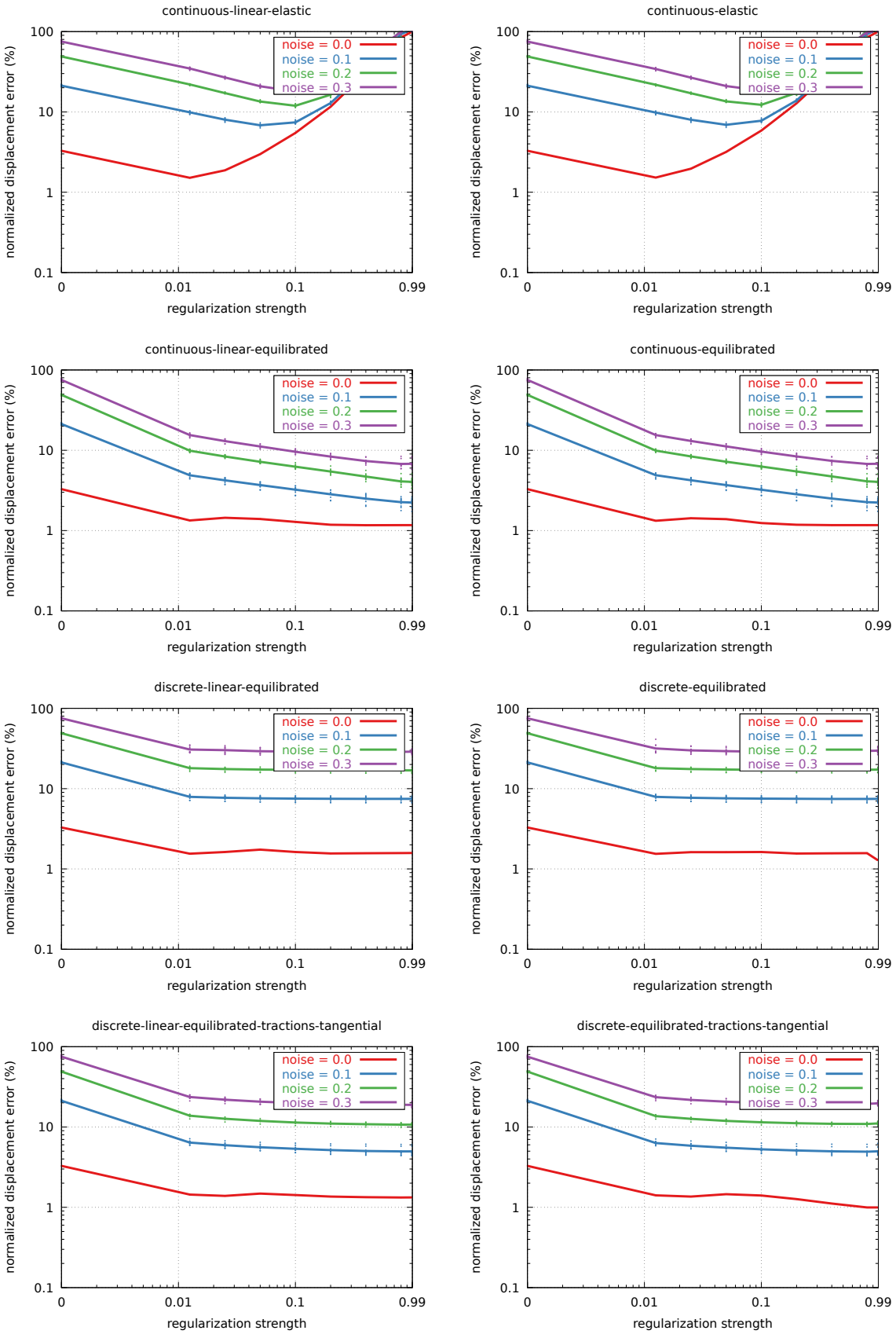
Figure 2: Rotation case.

3.2 Nonrigid Homogeneous deformation

We now consider nonrigid —though still homogeneous— deformations, namely compression (Figure 3) and shear (Figure 4). Conclusions are the same as for rigid transformations for all regularization terms based on the equilibrium gap principle (*i.e.*, second, third & fourth rows in Figures 3a & 4a), which perform very well. Regularization terms based on the elastic energy (*i.e.*, first row in Figures 3a & 4a) are, on the contrary, very problematic as they tend to prevent the mesh from deforming. This illustrates very well the fact that such regularization terms should not be used when tacking nonrigid deformations, as they interfere pathologically with the tracking.

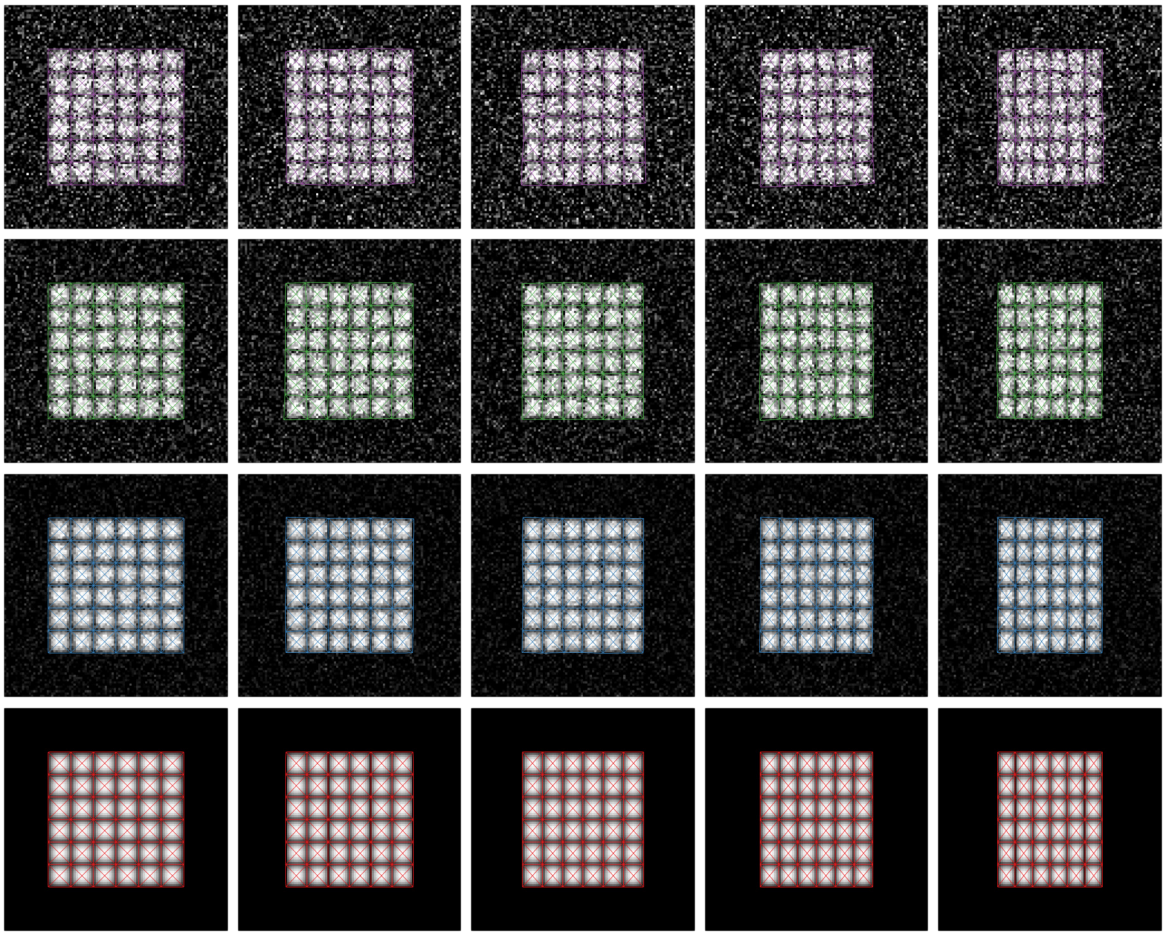
One can also notice that the continuous formulation of the equilibrium gap regularization (second row in Figures 3a & 4a) seems to perform better than the discrete formulation (third & fourth rows in Figures 3a & 4a); as we will see later, this is actually due to the fact that term (9) contains the equilibrium gap induced by the discretization in addition to the equilibrium gap induced by the images, and tends to minimize it, *i.e.*, to maintain the stress as homogeneous as possible, which helps the tracking here because the exact solution has indeed an homogeneous stress field, but which will prove highly pathological for non homogeneous cases.

It is important to notice also that for the compression case, only the tangential term (22) was included in the boundary traction terms (fourth row in Figure 3a). Indeed, the exact solution has highly non-smooth tractions (which are nonzero on left & right edges, zero on bottom & top edges), so that penalizing the surface gradient of the normal surface traction would not make sense. Similarly, in the shear case, only the normal term (18) was included in the boundary tractions terms (fourth row in Figure 4a).



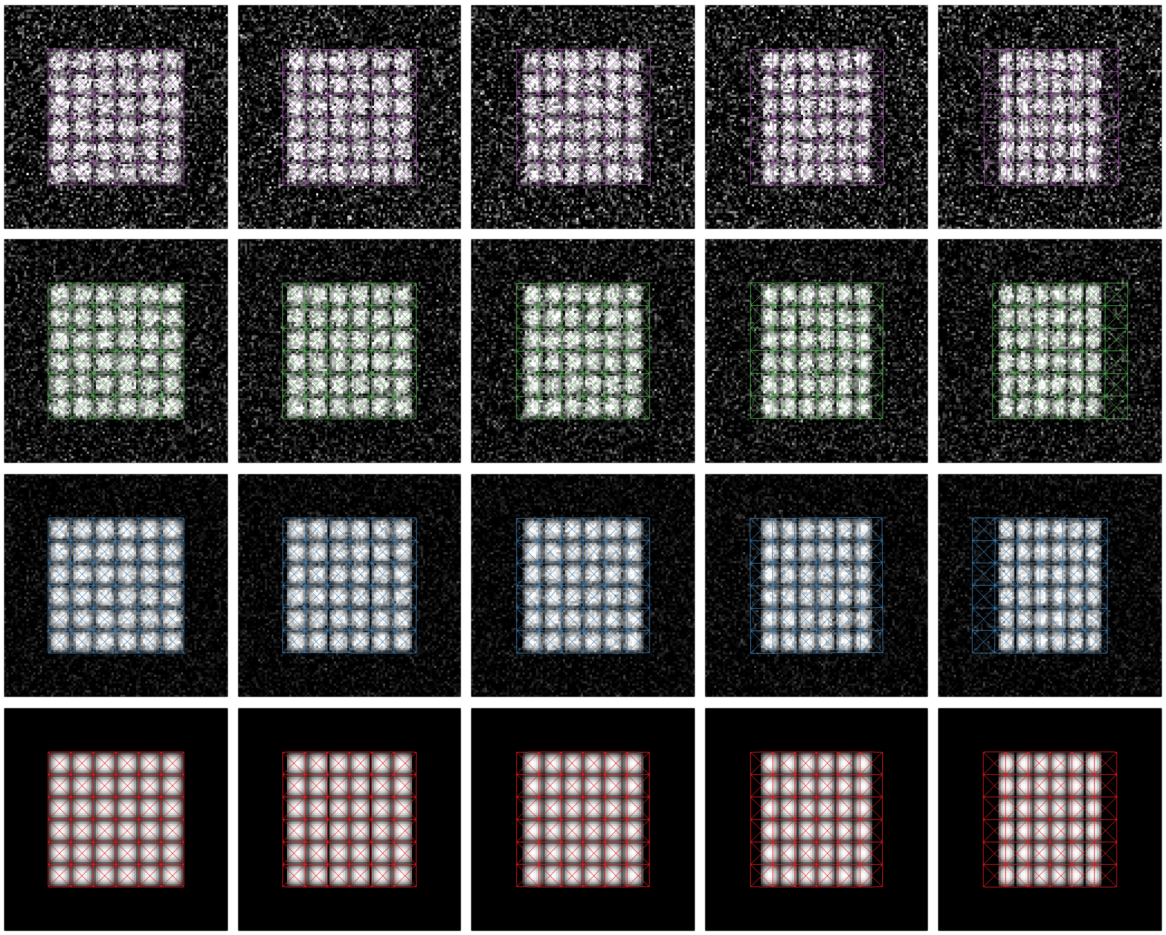
(a) Normalized displacement error (39) as a function of regularization strength (parameter β), for various levels of image noise, for various regularization terms (first row: elastic (7); second row: continuous version of the equilibrium gap (9); third row: discrete version of the equilibrium gap (12); fourth row: discrete version of the equilibrium gap including surface traction regularization (22)), and for various constitutive laws (left column: small strain, Hooke law; right column: large strain, neo-Hookean & Ogden-Ciarlet-Geymonat law). The elastic regularization terms interfere with the tracking because they basically prevent the mesh from deforming.

Figure 3: Compression case.



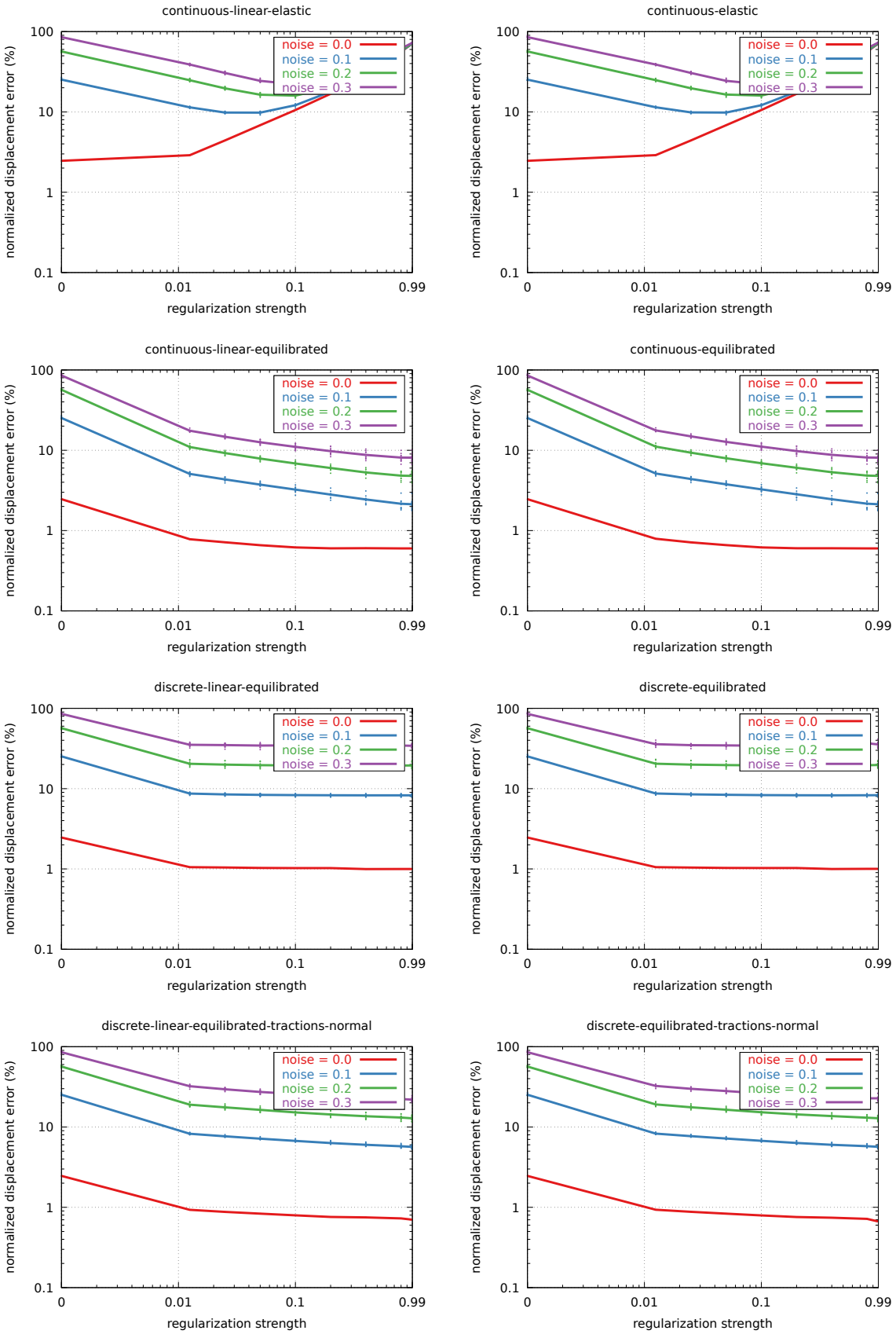
(b) Tracking solutions for the discrete version of the nonlinear equilibrium gap (12) including surface traction regularization (22), with regularization strength $\beta = 0.1$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). Tracking is visually satisfying for all noise levels.

Figure 3: Compression case.



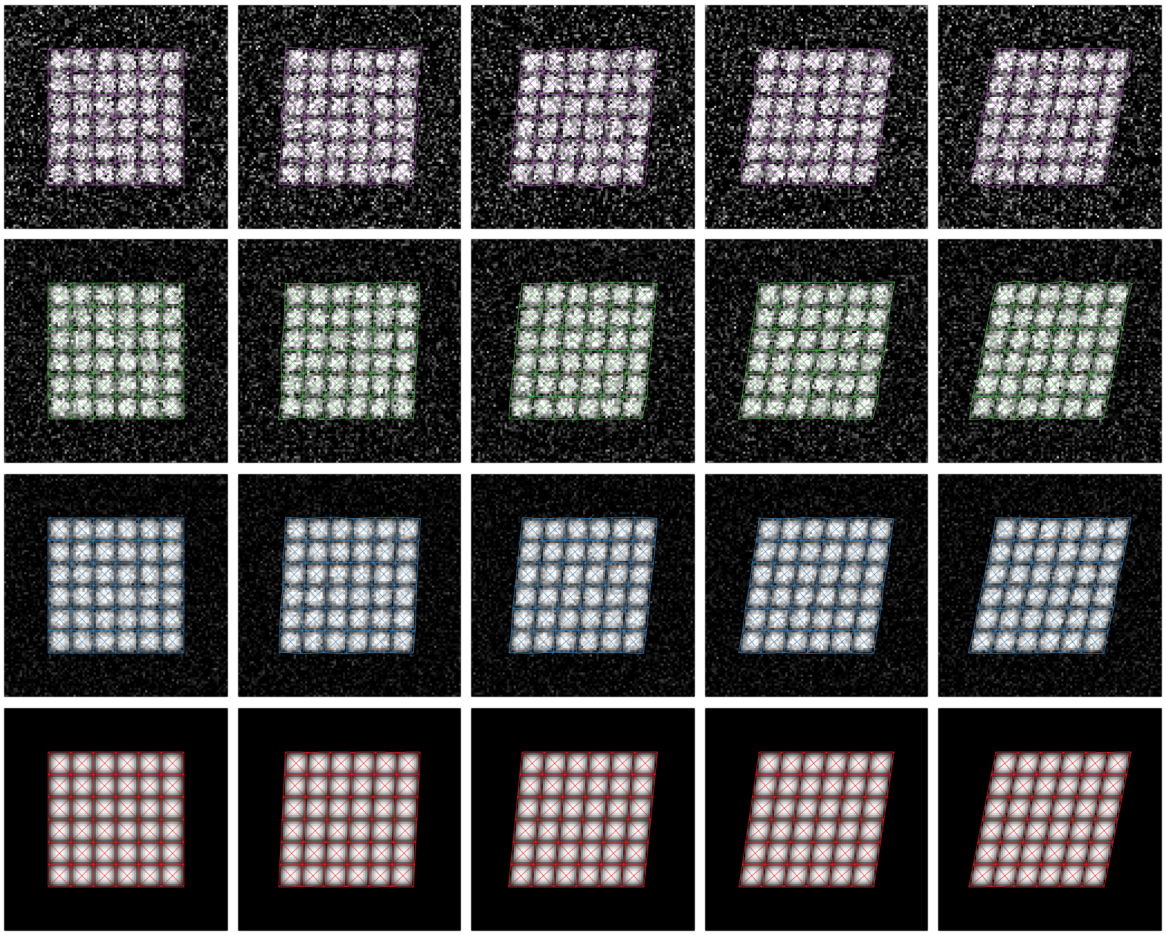
(c) Tracking solutions for the elastic regularization term (7), with regularization strength $\beta = 0.8$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). The elastic regularization terms tends to prevent the mesh from deforming, which degrades the tracking.

Figure 3: Compression case.



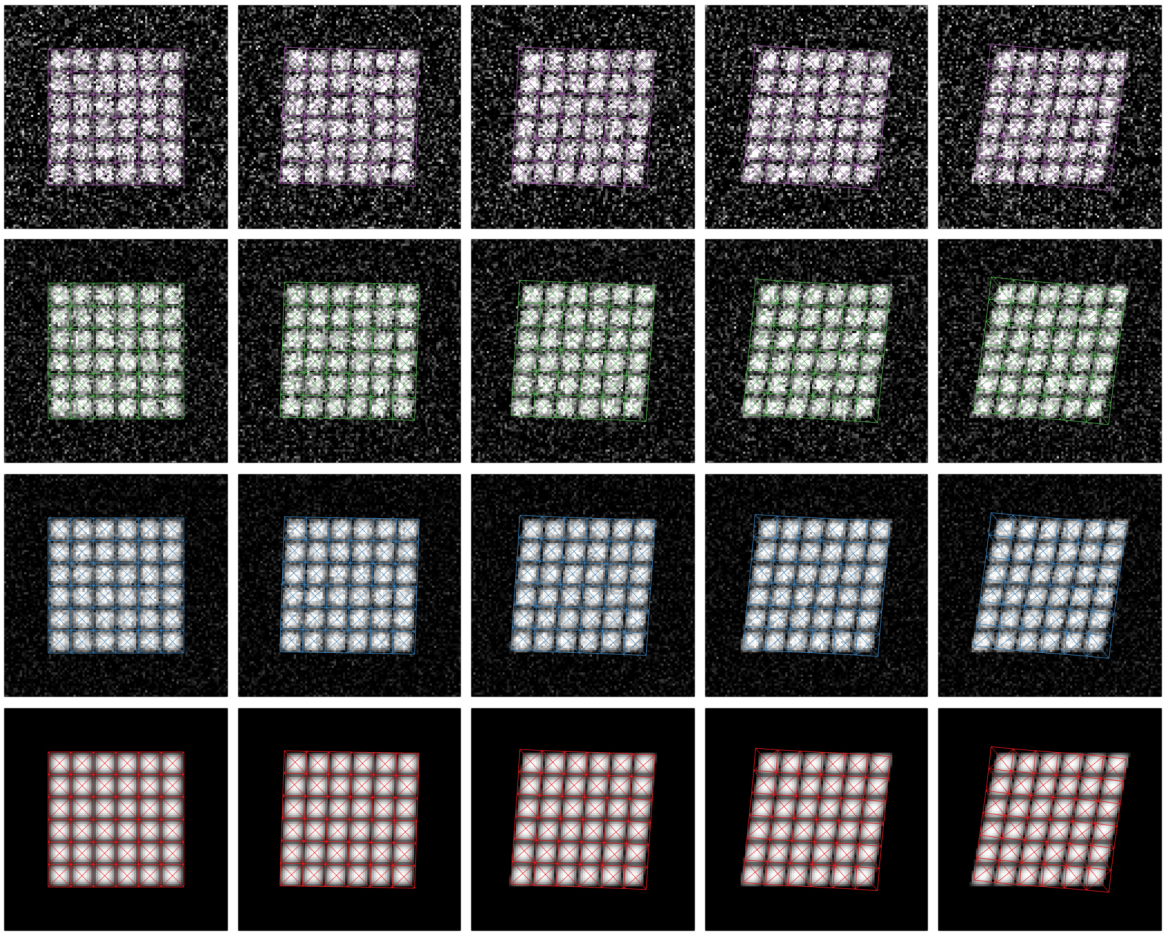
(a) Normalized displacement error (39) as a function of regularization strength (parameter β), for various levels of image noise, for various regularization terms (first row: elastic (7); second row: continuous version of the equilibrium gap (9); third row: discrete version of the equilibrium gap (12); fourth row: discrete version of the equilibrium gap including surface traction regularization (18)), and for various constitutive laws (left column: small strain, Hooke law; right column: large strain, neo-Hookean & Ogden-Ciarlet-Geymonat law). The elastic regularization terms interfere with the tracking because they basically prevent the mesh from deforming.

Figure 4: Shear case.



(b) Tracking solutions for the discrete version of the nonlinear equilibrium gap (12) including surface traction regularization (18), with regularization strength $\beta = 0.1$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). Tracking is visually satisfying for all noise levels.

Figure 4: Shear case.

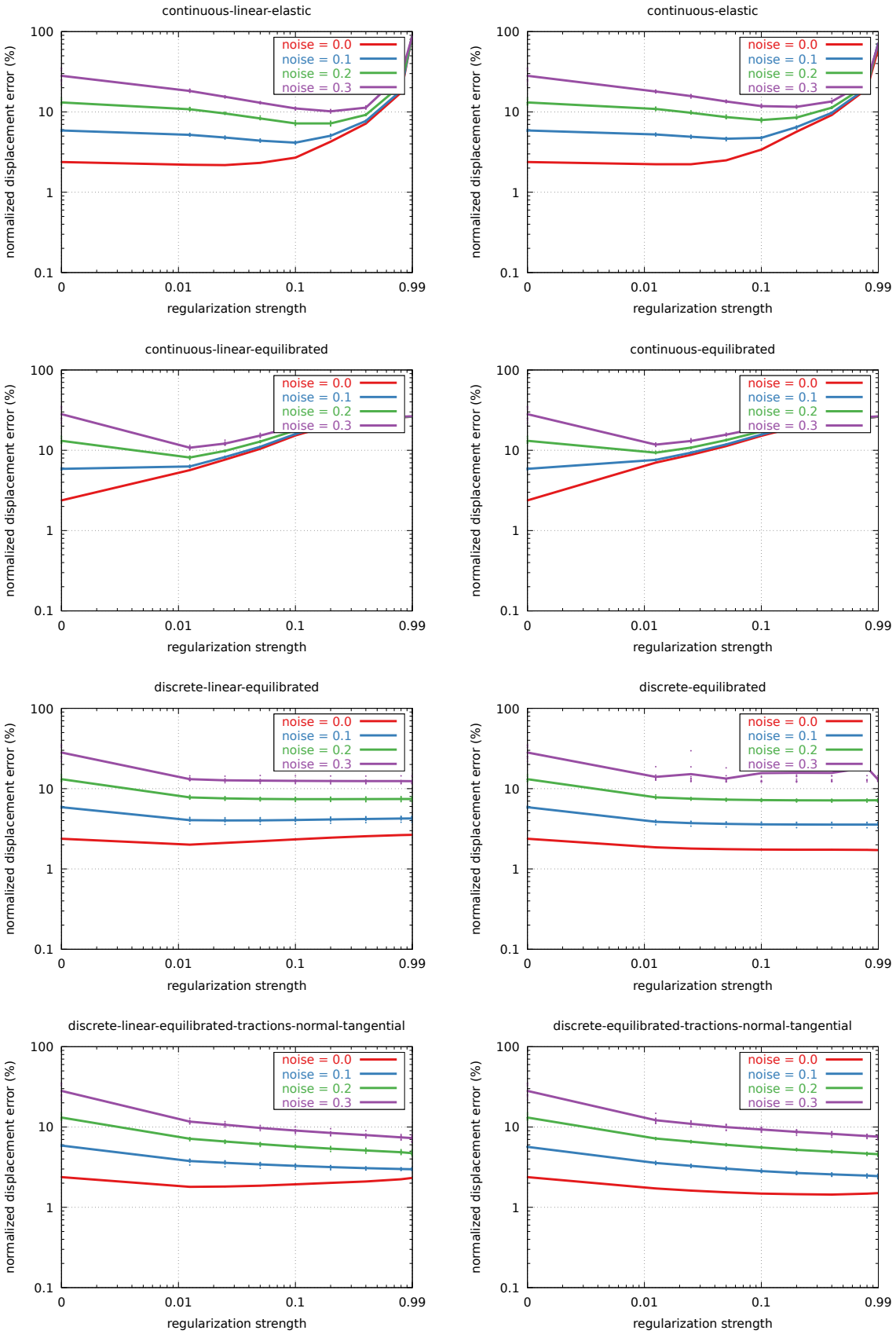


(c) Tracking solutions for the elastic regularization term (7), with regularization strength $\beta = 0.8$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). The elastic regularization terms tends to prevent the mesh from deforming, which degrades the tracking.

Figure 4: Shear case.

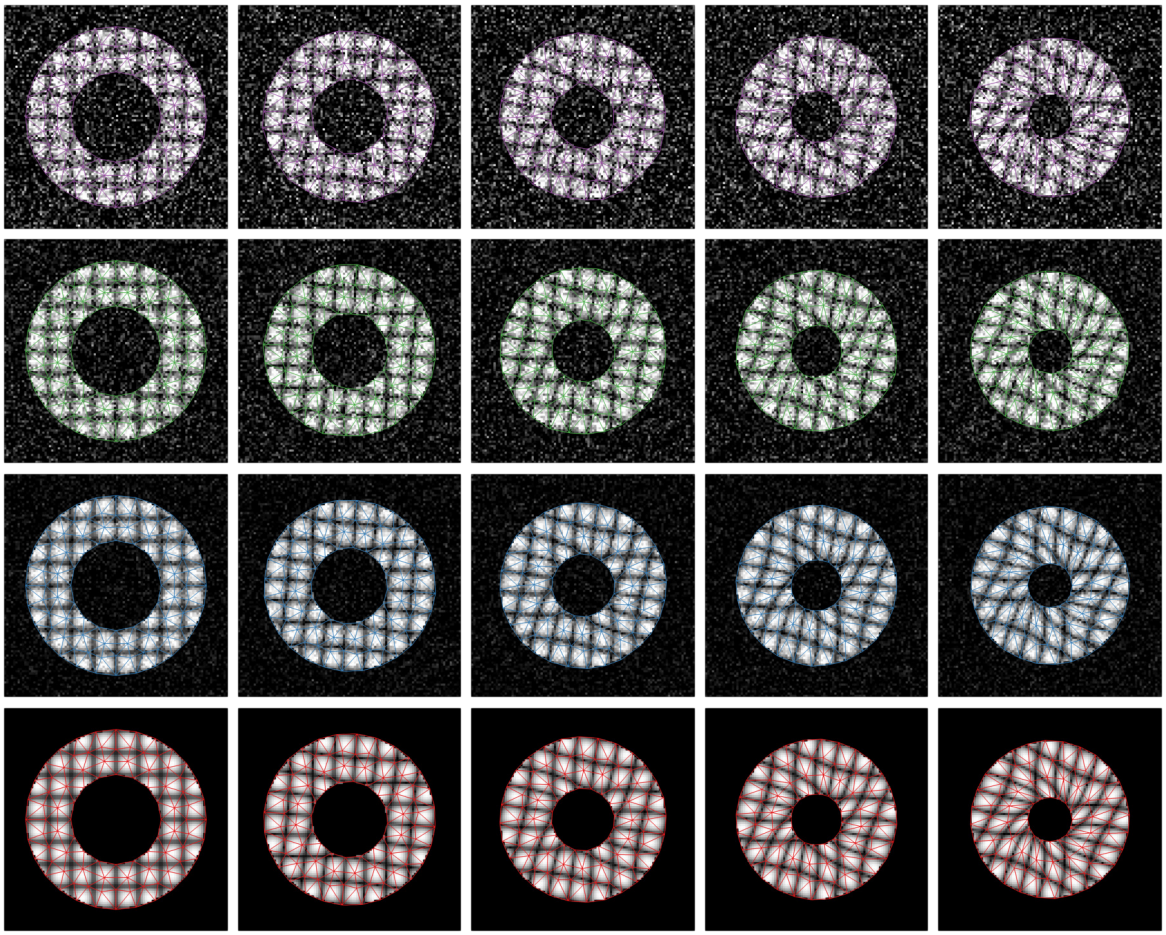
3.3 Nonrigid Heterogeneous deformation

Let us now consider more realistic images, mimicking cardiac tagged magnetic resonance imaging slices. Tracking results are presented in Figure 5. Elastic regularization (first row of Figure 5a) has the same limitation as for homogeneous deformations, in that it tends to prevent the mesh from deforming and thus interferes with the tracking. Interestingly, the continuous formulation of the equilibrium gap principle (second row of Figure 5a) presents similar limitations, though somewhat lighter for large regularization strengths. This can be explained by the fact that this "energy" contains the equilibrium gap induced by the finite element discretization, thus it tends to minimize the discretization error, *i.e.*, it tends to force the mesh to deform homogeneously, which was compatible with the ground truth of the simple examples of Sections 3.1 & 3.2, but not in more realistic cases. The discrete version of the equilibrium gap principle (third & fourth rows of Figure 5a), on the other hand, which characterizes the equilibrium gap induced by the motion itself but not the one induced by the discretization thanks to the projection step, does not have such pathological behavior, and allows the reduce the tracking error basically for all regularization levels. One can see that the nonlinear version (based on the finite strain framework and neoHookean & Ogden-Ciarlet-Geymonat potentials) performs slightly better than the linear version (based on the infinitesimal stain framework ant the Hooke law). Also, the surface traction regularization terms (fourth row in figure 5a) help reduce the tracking error mean and dispersion compared to the tracking with the bulk term only (third row in Figure 5a)).



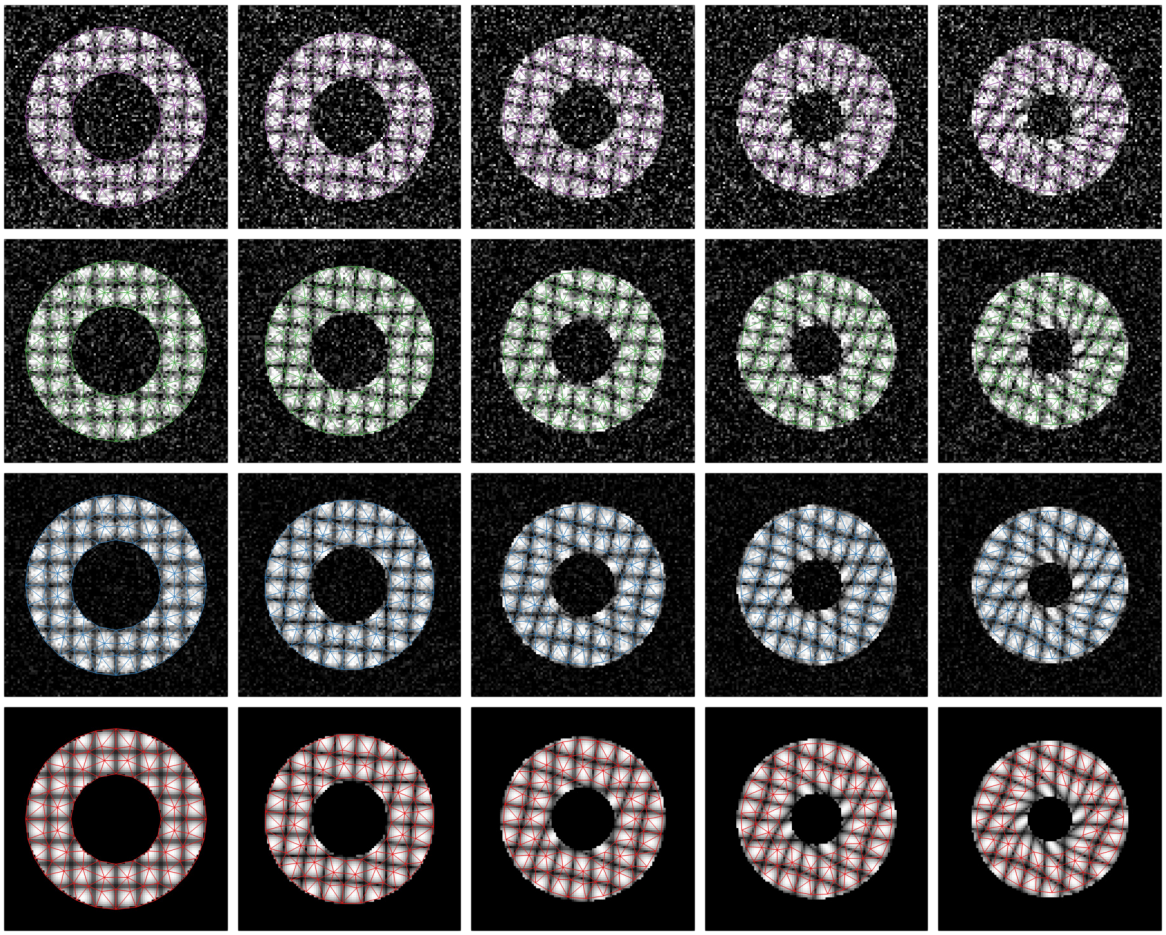
(a) Normalized displacement error (39) as a function of regularization strength (parameter β), for various levels of image noise, for various regularization terms (first row: elastic (7); second row: continuous version of the equilibrium gap (9); third row: discrete version of the equilibrium gap (12); fourth row: discrete version of the equilibrium gap including surface traction regularization (18) & (22)), and for various constitutive laws (left column: small strain, Hooke law; right column: large strain, neo-Hookean & Ogden-Ciarlet-Geymonat law). The elastic regularization terms interfere with the tracking because they basically prevent the mesh from deforming.

Figure 5: Cardiac-like case.



(b) Tracking solutions for the discrete version of the nonlinear equilibrium gap (12) including surface traction regularization terms (18) & (22), with regularization strength $\beta = 0.1$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). Tracking is visually satisfying for all noise levels.

Figure 5: Cardiac-like case.

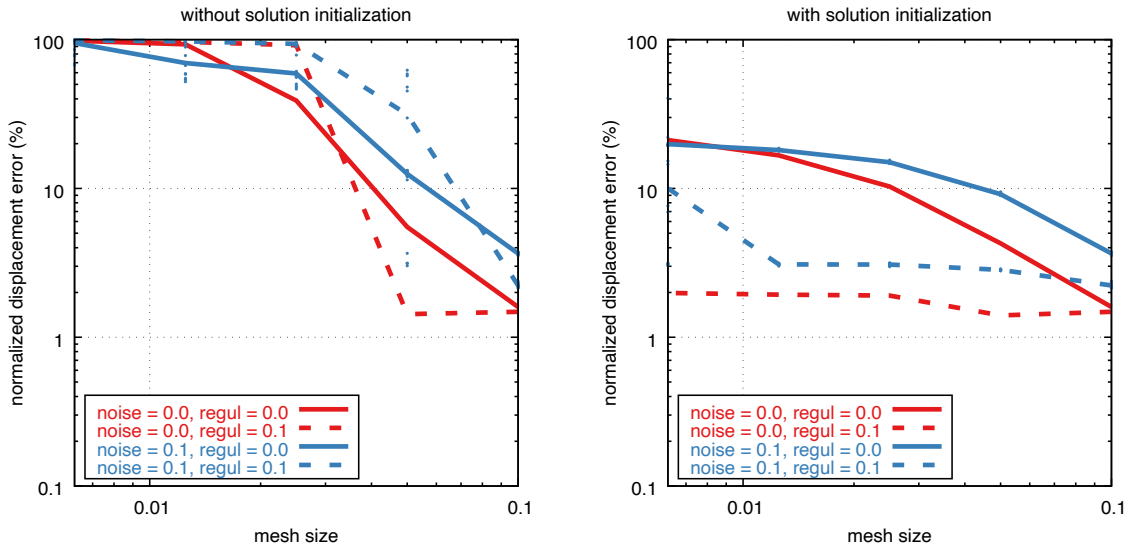


(c) Tracking solutions for the continuous version of the nonlinear equilibrium gap regularization term (9), with regularization strength $\beta = 0.8$, for various noise levels: 0 (red), 0.1 (blue), 0.2 (green), 0.3 (purple). The continuous version of the nonlinear equilibrium gap regularization term tends to prevent the mesh from deforming non homogeneously (which generates a discretization-induced equilibrium gap), which degrades the tracking.

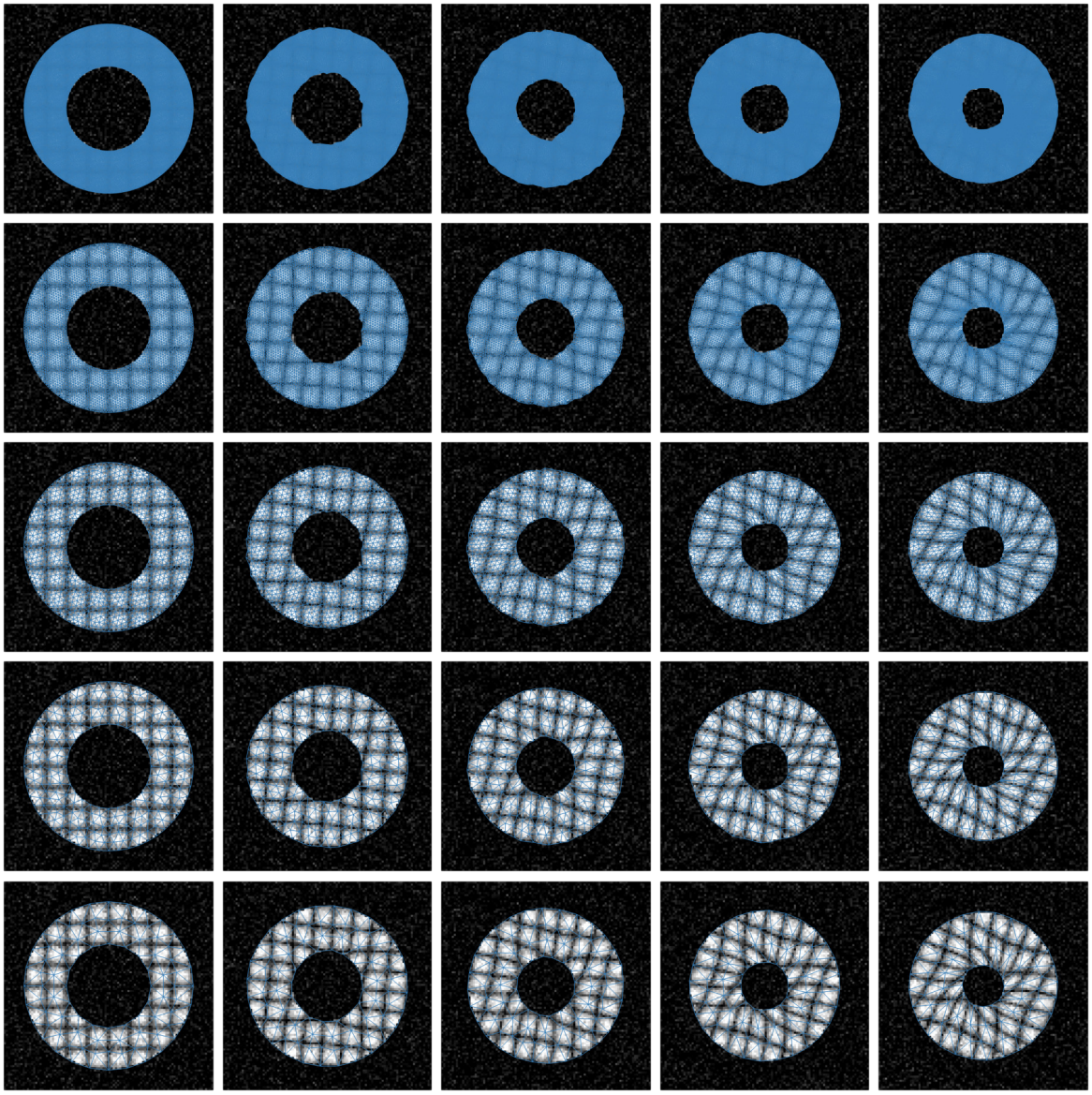
Figure 5: Cardiac-like case.

3.4 Impact of mesh size

We now investigate in details the impact of mesh size on tracking. Indeed, there is a fundamental trade-off between the richness of the motion model and the robustness of the tracking—the finite element discretization of the tracking already represents some kind of "geometrical' regularization, in the sense that the mesh controls the size of the finite dimensional functional space of the displacement field. Figure 6 shows tracking results for the cardiac-like problem, for various mesh sizes (from 0.1, which corresponds to the characteristic size of the image texture, like in the previous examples, to $0.1/2^4 = 0.00625$, which is smaller than the image pixel), for various levels of noise (0 & 0.1) and regularization strengths (O & 0.1), and for two different mesh refinement strategies: either we simply track the images with the different meshes independently (left in Figure 6a), or we track with successively refined meshes while initializing the tracking at a given mesh refinement level with the converged solution of the previous mesh refinement level (right in Figure 6a). First, one can see that directly tracking images with fine meshes is impossible—mechanical regularization helps a little in some cases, but not enough to obtain satisfying solutions. Thus, to obtain fine solutions, one needs to perform multi-resolution. However, one can also see that multi-resolution is not enough, and mechanical regularization is necessary to obtain fine solutions.



(a) Normalized displacement error (39) as a function of mesh size, for various levels of image noise, for the discrete version of the nonlinear equilibrium gap (12) including surface traction regularization terms (18) & (22), with various regularization strength.



(b) Tracking solutions for the discrete version of the nonlinear equilibrium gap (12) including surface traction regularization terms (18) & (22), with regularization strength $\beta = 0.1$.

Figure 6: Cardiac-like case, impact of mesh size.

4 Conclusion

In this paper, we introduced a consistent formulation, in the nonlinear large deformation setting, of the discrete equilibrium gap principle, and used it as a regularization for the large motion tracking problem. This principle enforces that the tracked motion corresponds to the motion of a body in equilibrium with some tractions applied on its boundary. We also introduced a novel regularization of the boundary tractions involved in the equilibrium gap principle, which naturally applies to both the finite and infinitesimal strain settings. All regularization terms have been implemented in an open-source finite element motion tracking library [Genet 2023a] (currently at `https://gitlab.inria.fr/mgenet/dolfin_warp`); moreover, the specific code to reproduce the results of this paper is also freely available [Genet 2023b] (in the appendix and currently at `https://mgenet.gitlabpages.inria.fr/N-DEG-paper-demos/index.html`).

In summary, we validated our approach by generating images representing various motions and with different signal-to-noise ratios, and then running our motion tracking algorithm with various regularization terms and regularization strengths. Basically we concluded that the finite strain framework is necessary when large motion —especially large rotation— is involved. We showed that elastic regularization interferes pathologically with nonrigid motion tracking, and should essentially not be used in such cases. In the case of heterogenous motion, which is the most representative of actual applications of motion tracking [Genet, Lee, et al. 2014; Finsberg et al. 2019], we also concluded that our previous "continuous" formulation of the equilibrium gap principle interferes with the tracking, while our new "discrete" formulation performs well. We also showed that, combining multi-level resolution and our mechanical regularization, we were able to track rather large motion with a displacement discretization as fine as the image discretization itself, so that all features of the images motion can be tracked.

However, there are multiple limitations to our current approach, at various levels. The formulation of the equilibrium gap principle introduced here only considers a single homogeneous domain, which might be limiting when tracking motion of highly heterogeneous structures such as diseases organs [Patte et al. 2022; Laville et al. 2023]. In such case, one might need to extend the formulation to multiple zones. Another option would be to estimate the material parameters of the mechanical model used for the regularization term at the same time as the tracking [Mathieu et al. 2015]. Actually, the equilibrium gap principle could also be used as a cost function for direct (*i.e.*, without the need for a resolution of the direct mechanics problem) material parameter estimation from displacement measured by any method—the small stain formulation has already been used successfully [Claire et al. 2004], but in large strain only the virtual fields method has been used, which represents an approximation of the equilibrium gap method as it only satisfies the equilibrium in a weak sense for a selected member of test functions [Avril et al. 2008].

Another limitation, especially in terms of applications, is that the performance of the method comes at a significant computational cost, limiting its use notably in the clinics. Thus, our method could be used to generate ground truth and/or validation data for machine learning algorithms, which could then perform tracking almost instantaneously. This is an active field of research today [Leiner et al. 2019; Friedrich et al. 2021; Koehler et al. 2022; López et al. 2022].

An important perspective of our motion tracking algorithm is toward low resolution images, which have been proven to drastically impact the tracking quality [Berberoğlu, Stoeck, Moireau, et al. 2021]. One option is to combine images from multiple modalities and combine the strength of both images [Berberoğlu, Stoeck, Kozerke, et al. 2022]. Another option to consider is to introduce a model of the imaging modality (e.g., MRI [Škardová et al. 2019]), in order to control the bias induced by the image discretization. Thus, by combining models of the imaging process and the mechanical deformation, we will be able to perform high quality tracking even with low quality images.

Acknowledgements

MG would like to thank Philippe Moireau, Patrick Le Tallec & Pierre Kerfriden for fruitful discussions.

References

Allaire, G. (2007). *Numerical Analysis and Optimization: An Introduction to Mathematical Modelling and Numerical Simulation*. Numerical Mathematics and Scientific Computation. Oxford ; New York: Oxford University Press. 455 pp.

Alnæs, M., J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. N. Wells (2015). "The FEniCS Project Version 1.5". In: *Archive of Numerical Software* Vol 3. In collab. with A. O. N. Software. DOI: 10.11588/ans.2015.100.20553.

Avril, S., M. Bonnet, A.-S. Bretelle, M. Grédiac, F. Hild, P. Ienny, F. Latourte, D. Lemosse, S. Pagano, E. Pagnacco, and F. Pierron (2008). "Overview of Identification Methods of Mechanical Parameters Based on Full-field Measurements". In: *Experimental Mechanics* 48.4, pp. 381–402. DOI: 10.1007/s11340-008-9148-y.

Berberoğlu, E., C. T. Stoeck, S. Kozerke, and M. Genet (2022). "Quantification of Left Ventricular Strain and Torsion by Joint Analysis of 3D Tagging and Cine MR Images". In: *Medical Image Analysis* 82, p. 102598. DOI: 10.1016/j.media.2022.102598.

Berberoğlu, E., C. T. Stoeck, P. Moireau, S. Kozerke, and M. Genet (2019). "Validation of Finite Element Image Registration-based Cardiac Strain Estimation from Magnetic Resonance Images". In: *PAMM* 19.1. DOI: 10.1002/pamm.201900418.

— (2021). "In-Silico Study of Accuracy and Precision of Left-Ventricular Strain Quantification from 3D Tagged MRI". In: *PLOS ONE* 16.11, e0258965. DOI: 10.1371/journal.pone.0258965.

Bornert, M., F. Brémand, P. Doumalin, J.-C. Dupré, M. Fazzini, M. Grédiac, F. Hild, S. Mistou, J. Molimard, J.-J. Orteu, L. Robert, Y. Surrel, P. Vacher, and B. Wattrisse (2009). "Assessment of Digital Image Correlation Measurement Errors: Methodology and Results". In: *Experimental Mechanics* 49.3, pp. 353–370. DOI: 10.1007/s11340-008-9204-7.

Brandner, P., T. Jankuhn, S. Praetorius, A. Reuksen, and A. Voigt (2021). "Finite Element Discretization Methods for Velocity-Pressure and Stream Function Formulations of Surface Stokes Equations".

Castellanos, D. A., K. Škardová, A. Bhattaru, E. Berberoğlu, G. Greil, A. Tandon, J. Dillenbeck, B. Burkhardt, T. Hussain, M. Genet, and R. Chabiniok (2021). "Left Ventricular Torsion Obtained Using Equilibrated Warping in Patients with Repaired Tetralogy of Fallot". In: *Pediatric Cardiology*. DOI: 10.1007/s00246-021-02608-y.

Christensen, G. E., R. D. Rabbitt, and M. I. Miller (1996). "Deformable Templates Using Large Deformation Kinematics". In: *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society* 5.10, pp. 1435–47. DOI: 10.1109/83.536892.

Chu, T. C., W. F. Ranson, and M. A. Sutton (1985). "Applications of Digital-Image-Correlation Techniques to Experimental Mechanics". In: *Experimental Mechanics* 25.3, pp. 232–244. DOI: 10.1007/BF02325092.

Ciarlet, P. G. and G. Geymonat (1982). "Sur Les Lois de Comportement En Élasticité Non-Linéaire Compressible". In: *Comptes Rendus de l'Académie des Sciences Série II* 295, pp. 423–426.

Claire, D., F. Hild, and S. Roux (2004). "A Finite Element Formulation to Identify Damage Fields: The Equilibrium Gap Method". In: *International Journal for Numerical Methods in Engineering* 61.2, pp. 189–208. DOI: 10.1002/nme.1057.

Finsberg, H., C. Xi, X. Zhao, J. L. Tan, M. Genet, J. Sundnes, L. C. Lee, L. Zhong, and S. T. Wall (2019). "Computational Quantification of Patient-Specific Changes in Ventricular Dynamics Associated with Pulmonary Hypertension". In: *American Journal of Physiology-Heart and Circulatory Physiology* 317.6, H1363–H1375. DOI: 10.1152/ajpheart.00094.2019.

Friedrich, S., S. Groß, I. R. König, S. Engelhardt, M. Bahls, J. Heinz, C. Huber, L. Kaderali, M. Kelm, A. Leha, J. Rühl, J. Schaller, C. Scherer, M. Vollmer, T. Seidler, and T. Friede (2021). "Applications of Artificial Intelligence/Machine Learning Approaches in Cardiovascular Medicine: A Systematic Review with Recommendations". In: *European Heart Journal - Digital Health* 2.3, pp. 424–436. DOI: 10.1093/ehjdh/ztab054.

Garot, J., D. A. Bluemke, N. F. Osman, C. E. Rochitte, E. R. McVeigh, E. A. Zerhouni, J. L. Prince, and J. A. C. Lima (2000). "Fast Determination of Regional Myocardial Strain Fields From Tagged Cardiac Images Using Harmonic Phase MRI". In: *Circulation* 101.9, pp. 981–988. DOI: 10.1161/01.CIR.101.9.981.

Genet, M. (2019). "A Relaxed Growth Modeling Framework for Controlling Growth-Induced Residual Stresses". In: *Clinical Biomechanics* 70, pp. 270–277. DOI: 10.1016/j.clinbiomech.2019.08.015.

— (2023a). *Dolfin_warp*. URL: https://doi.org/10.5281/zenodo.8010275.

— (2023b). *N-DEG-paper-demos*. URL: https://doi.org/10.5281/zenodo.8010517.

Genet, M., L. C. Lee, R. Nguyen, H. Haraldsson, G. Acevedo-Bolton, Z. Zhang, L. Ge, K. Ordovas, S. Kozerke, and J. M. Guccione (2014). "Distribution of Normal Human Left Ventricular Myofiber Stress at End Diastole and End Systole: A Target for in Silico Design of Heart Failure Treatments". In: *Journal of Applied Physiology* 117, pp. 142–152. DOI: 10.1152/japplphysiol.00255.2014.

Genet, M., C. T. Stoeck, C. von Deuster, L. C. Lee, and S. Kozerke (2018). "Equilibrated Warping: Finite Element Image Registration with Finite Strain Equilibrium Gap Regularization". In: *Medical Image Analysis* 50, pp. 1–22. DOI: 10.1016/j.media.2018.07.007.

Hild, F. and S. Roux (2012). "Comparison of Local and Global Approaches to Digital Image Correlation". In: *Experimental Mechanics* 52.9, pp. 1503–1519. DOI: 10.1007/s11340-012-9603-7.

Hild, F. and S. Roux (2006). "Digital Image Correlation: From Displacement Measurement to Identification of Elastic Properties - a Review". In: *Strain* 42.2, pp. 69–80. DOI: 10.1111/j.1475-1305.2006.00258.x.

Koehler, S., T. Hussain, H. Hussain, D. Young, S. Sarikouch, T. Pickardt, G. Greil, and S. Engelhardt (2022). "Self-Supervised Motion Descriptor for Cardiac Phase Detection in 4D CMR Based on Discrete Vector Field Estimations". In: *Statistical Atlases and Computational Models of the Heart. Regular and CMRxMotion Challenge Papers*. Ed. by O. Camara, E. Puyol-Antón, C. Qin, M. Sermesant, A. Suinesiaputra, S. Wang, and A. Young. Vol. 13593. Cham: Springer Nature Switzerland, pp. 65–78. DOI: 10.1007/978-3-031-23443-9_7.

Ladevèze, P. and J. P. Pelle (2005). *Mastering Calculations in Linear and Nonlinear Mechanics*. Mechanical Engineering Series. New York: Springer Science. 413 pp.

Laville, C., C. Fetita, T. Gille, P.-Y. Brillet, H. Nunes, J.-F. Bernaudin, and M. Genet (2023). "Comparison of Optimization Parametrizations for Regional Lung Compliance Estimation Using Personalized Pulmonary Poromechanical Modeling". In: *Biomechanics and Modeling in Mechanobiology*. DOI: 10.1007/s10237-023-01691-9.

Le Tallec, P. (1994). "Numerical Methods for Nonlinear Elasticity". In: *Handbook of Numerical Analysis*. Vol. 3, pp. 465–622. DOI: 10.1016/S1570-8659(05)80018-3.

Leclerc, H., J.-N. Périé, S. Roux, and F. Hild (2010). "Voxel-Scale Digital Volume Correlation". In: *Experimental Mechanics* 51.4, pp. 479–490. DOI: 10.1007/s11340-010-9407-6.

Lee, L. C. and M. Genet (2019). "Validation of Equilibrated Warping—Image Registration with Mechanical Regularization—On 3D Ultrasound Images". In: *Functional Imaging and Modeling of the Heart (FIMH)*. Bordeaux, France. DOI: 10.1007/978-3-030-21949-9_36.

Leiner, T., D. Rueckert, A. Suinesiaputra, B. Baeßler, R. Nezafat, I. Išgum, and A. A. Young (2019). "Machine Learning in Cardiovascular Magnetic Resonance: Basic Concepts and Applications". In: *Journal of Cardiovascular Magnetic Resonance* 21.1, pp. 1–14. DOI: 10.1186/s12968-019-0575-y.

Lenoir, N., M. Bornert, J. Desrues, P. Bésuelle, and G. Viggiani (2007). "Volumetric Digital Image Correlation Applied to X-ray Microtomography Images from Triaxial Compression Tests on Argillaceous Rock". In: *Strain* 43.3, pp. 193–205. DOI: 10.1111/j.1475-1305.2007.00348.x.

Logg, A., K.-A. Mardal, and G. Wells, eds. (2012). *Automated Solution of Differential Equations by the Finite Element Method: The FEniCS Book*. Lecture Notes in Computational Science and Engineering 84. Heidelberg: Springer. 723 pp.

López, P. A., H. Mella, S. Uribe, D. E. Hurtado, and F. S. Costabal (2022). *WarpPINN: Cine-MR Image Registration with Physics-Informed Neural Networks*. URL: http://arxiv.org/abs/2211.12549. preprint.

Mansi, T., X. Pennec, M. Sermesant, H. Delingette, and N. Ayache (2011). "iLogDemons: A Demons-Based Registration Algorithm for Tracking Incompressible Elastic Biological Tissues". In: *International Journal of Computer Vision* 92.1, pp. 92–111. DOI: 10.1007/s11263-010-0405-z.

Mathieu, F., H. Leclerc, F. Hild, and S. Roux (2015). "Estimation of Elastoplastic Parameters via Weighted FEMU and Integrated-DIC". In: *Experimental Mechanics* 55.1, pp. 105–119. DOI: 10.1007/s11340-014-9888-9.

Miller, M. I., G. E. Christensen, Y. Amit, and U. Grenander (1993). "Mathematical Textbook of Deformable Neuroanatomies." In: *Proceedings of the National Academy of Sciences* 90.24, pp. 11944–11948. DOI: 10.1073/pnas.90.24.11944.

Ogden, R. W. (1972). "Large Deformation Isotropic Elasticity: On the Correlation of Theory and Experiment for Compressible Rubberlike Solids". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 328.1575, pp. 567–583. DOI: 10.1098/rspa.1972.0096.

Passieux, J.-C. and J.-N. Périé (2012). "High Resolution Digital Image Correlation Using Proper Generalized Decomposition: PGD-DIC". In: *International Journal for Numerical Methods in Engineering* 92.6, pp. 531–550. DOI: 10.1002/nme.4349.

Patte, C., P.-Y. Brillet, C. Fetita, T. Gille, J.-F. Bernaudin, H. Nunes, D. Chapelle, and M. Genet (2022). "Estimation of Regional Pulmonary Compliance in Idiopathic Pulmonary Fibrosis Based on Personalized Lung Poromechanical Modeling". In: *Journal of Biomechanical Engineering*. DOI: 10.1115/1.4054106.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (2007). *Numerical Recipes: The Art of Scientific Computing*. 3rd ed. Cambridge, UK ; New York: Cambridge University Press. 1235 pp.

Réthoré, J., S. Roux, and F. Hild (2009). "An extended and integrated digital image correlation technique applied to the analysis of fractured samples: The equilibrium gap method as a mechanical filter". In: *European Journal of Computational Mechanics* 18.3-4, pp. 285–306. DOI: 10.3166/ejcm.18.285-306.

Rutz, A. K., S. Ryf, S. Plein, P. Boesiger, and S. Kozerke (2008). "Accelerated Whole-Heart 3D CSPAMM for Myocardial Motion Quantification." In: *Magnetic resonance in medicine* 59.4, pp. 755–63. DOI: 10.1002/mrm.21363.

Schroeder, W., K. Martin, and B. Lorensen (2006). *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*. 4. ed. Clifton Park, NY: Kitware, Inc. 512 pp.

Škardová, K., M. Rambausek, R. Chabiniok, and M. Genet (2019). "Mechanical and Imaging Models-Based Image Registration". In: *VipIMAGE 2019*. Ed. by J. M. R. S. Tavares and R. M. Natal Jorge. Vol. 34. Cham: Springer International Publishing, pp. 77–85. DOI: 10.1007/978-3-030-32040-9_9.

Smith, N. P., A. de Vecchi, M. McCormick, D. A. Nordsletten, O. Camara, A. F. Frangi, H. Delingette, M. Sermesant, J. Relan, N. Ayache, M. W. Krueger, W. H. W. Schulze, R. Hose, I. Valverde, P. Beerbaum, C. Staicu, M. Siebes, J. Spaan, P. J. Hunter, J. Weese, H. Lehmann, D. Chapelle, and R. Rezavi (2011). "euHeart: Personalized and Integrated Cardiac Care Using Patient-Specific Cardiovascular Modelling". In: *Interface focus* 1.3, pp. 349–64. DOI: 10.1098/rsfs.2010.0048.

Sotiras, A., C. Davatzikos, and N. Paragios (2013). "Deformable Medical Image Registration: A Survey". In: *IEEE Transactions on Medical Imaging* 32.7, pp. 1153–1190. DOI: 10.1109/TMI.2013.2265603.

Tobon-Gomez, C., M. De Craene, K. McLeod, L. Tautz, W. Shi, A. Hennemuth, A. Prakosa, H. Wang, G. S. Carr-White, S. Kapetanakis, A. Lutz, V. Rasche, T. Schaeffter, C. Butakoff, O. Friman, T. Mansi, M. Sermesant, X. Zhuang, S. Ourselin, H.-O. Peitgen, X. Pennec, R. Razavi, D. Rueckert, A. F. Frangi, and K. S. Rhode (2013). "Benchmarking Framework for Myocardial Tracking and Deformation Algorithms: An Open Access Database". In: *Medical Image Analysis* 17.6, pp. 632–648. DOI: 10.1016/j.media.2013.03.008.

Tueni, N., J. Vizet, M. Genet, A. Pierangelo, and J.-M. Allain (2020). "Microstructural Deformation Observed by Mueller Polarimetry during Traction Assay on Myocardium Samples". In: *Scientific Reports* 10.1, p. 20531. DOI: 10.1038/s41598-020-76820-w.

Veress, A. I., G. T. Gullberg, and J. A. Weiss (2005). "Measurement of Strain in the Left Ventricle during Diastole with Cine-MRI and Deformable Image Registration". In: *Journal of Biomechanical Engineering* 127.7, p. 1195. DOI: 10.1115/1.2073677.

Xi, C., C. Latnie, X. Zhao, J. L. Tan, S. T. Wall, M. Genet, L. Zhong, and L. C. Lee (2016). "Patient-Specific Computational Analysis of Ventricular Mechanics in Pulmonary Arterial Hypertension". In: *Journal of Biomechanical Engineering* 138.11, p. 111001. DOI: 10.1115/1.4034559.

Zienkiewicz, O., R. Taylor, and J. Zhu (2013). "Ch. 15 — Errors, Recovery Processes, and Error Estimates". In: *The Finite Element Method: Its Basis and Fundamentals*. Elsevier, pp. 493–543. DOI: 10.1016/B978-1-85617-633-0.00015-0.

Zou, H., S. Leng, C. Xi, X. Zhao, A. S. Koh, F. Gao, J. L. Tan, R.-S. Tan, J. C. Allen, L. C. Lee, M. Genet, and L. Zhong (2020). "Three-Dimensional Biventricular Strains in Pulmonary Arterial Hypertension Patients Using Hyperelastic Warping". In: *Computer Methods and Programs in Biomedicine* 189, p. 105345. DOI: 10.1016/j.cmpb.2020.105345.

Zou, H., C. Xi, X. Zhao, A. S. Koh, F. Gao, Y. Su, R.-S. Tan, J. Allen, L. C. Lee, M. Genet, and L. Zhong (2018). "Quantification of Biventricular Strains in Heart Failure With Preserved Ejection Fraction Patient Using Hyperelastic Warping Method". In: *Frontiers in Physiology* 9. DOI: 10.3389/fphys.2018.01295.

Appendix

A Code for Figures 1, 2, 3 & 4

Imports

[ ]:
```
import dolfin  # https://fenicsproject.org
import IPython # https://ipython.org
import vtk     # https://vtk.org

import dolfin_warp as dwarp # https://gitlab.inria.fr/mgenet/dolfin_warp

from generate_images_and_meshes_from_Struct import generate_images_and_meshes_from_Struct
from plot_disp_error_vs_regul_strength      import plot_disp_error_vs_regul_strength
from lib_viewer                             import Viewer
```

Parameters

[ ]:
```
n_dim = 2

images_folder = "generate_images"

n_voxels = 100

structure_deformation_type_lst  = [                          ]
structure_deformation_type_lst += [["square", "translation"]]
structure_deformation_type_lst += [["square", "rotation"   ]]
structure_deformation_type_lst += [["square", "compression"]]
structure_deformation_type_lst += [["square", "shear"      ]]

texture_type_lst  = [         ]
texture_type_lst += ["tagging"]

noise_level_lst  = [   ]
noise_level_lst += [0.0]
noise_level_lst += [0.1]
noise_level_lst += [0.2]
noise_level_lst += [0.3]

n_runs_for_noisy_images = 10

working_folder  = "run_warp"

mesh_size_lst  = [   ]
mesh_size_lst += [0.1]

regul_type_lst  = [                                                  ]
regul_type_lst += ["continuous-linear-elastic"                       ]
regul_type_lst += ["continuous-linear-equilibrated"                  ]
regul_type_lst += ["continuous-elastic"                              ]
regul_type_lst += ["continuous-equilibrated"                         ]
regul_type_lst += ["discrete-simple-elastic"                         ]
regul_type_lst += ["discrete-simple-equilibrated"                    ]
regul_type_lst += ["discrete-linear-equilibrated"                    ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal"   ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-tangential"]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal-tangential"]
regul_type_lst += ["discrete-equilibrated"                           ]
regul_type_lst += ["discrete-equilibrated-tractions-normal"          ]
regul_type_lst += ["discrete-equilibrated-tractions-tangential"      ]
regul_type_lst += ["discrete-equilibrated-tractions-normal-tangential"]

regul_level_lst  = [        ]
regul_level_lst += [0.99    ]
regul_level_lst += [0.1*2**3]
```

```
regul_level_lst += [0.1*2**2]
regul_level_lst += [0.1*2**1]
regul_level_lst += [0.1     ]
regul_level_lst += [0.1/2**1]
regul_level_lst += [0.1/2**2]
regul_level_lst += [0.1/2**3]
regul_level_lst += [0.0     ]

do_generate_images                  = 1
do_generate_meshes                  = 1
do_run_warp                         = 1
do_plot_disp_error_vs_regul_strength = 1
```

Synthetic images

[ ]:
```
if (do_generate_images):
 for structure_type, deformation_type in structure_deformation_type_lst:
  for texture_type                     in texture_type_lst              :
   for noise_level                      in noise_level_lst               :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run in range(1, n_runs+1):

        print("*** generate_images ***"           )
        print("structure_type:"  , structure_type  )
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )

        generate_images_and_meshes_from_Struct(
            n_dim            = n_dim                          ,
            n_voxels         = n_voxels                       ,
            structure_type   = structure_type                 ,
            deformation_type = deformation_type               ,
            texture_type     = texture_type                   ,
            noise_level      = noise_level                    ,
            k_run            = k_run if (n_runs > 1) else None,
            generate_images  = 1                              ,
            compute_meshes   = 0                              )
```

Ground truth motion

[ ]:
```
if (do_generate_meshes):
 for structure_type, deformation_type in structure_deformation_type_lst:
  for mesh_size                        in mesh_size_lst                :

    print("*** generate_meshes ***"           )
    print("structure_type:"  , structure_type  )
    print("deformation_type:", deformation_type)
    print("mesh_size:"       , mesh_size       )

    generate_images_and_meshes_from_Struct(
        n_dim            = n_dim          ,
        n_voxels         = n_voxels       ,
        structure_type   = structure_type ,
        deformation_type = deformation_type,
        texture_type     = "no"           ,
        noise_level      = 0              ,
        mesh_size        = mesh_size      ,
        generate_images  = 0              ,
        compute_meshes   = 1              )
```

Tracking

```
[ ]: if (do_run_warp):
      for structure_type, deformation_type in structure_deformation_type_lst:
       for texture_type                    in texture_type_lst              :
        for noise_level                     in noise_level_lst              :

         n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

         for k_run       in range(1, n_runs+1):
          for mesh_size   in mesh_size_lst     :
           for regul_type  in regul_type_lst    :
            for regul_level in regul_level_lst   :

             if any([_ in regul_type for _ in ["linear", "simple"]]):
                 regul_model = "hooke"
             else:
                 regul_model = "ciarletgeymonatneohookean"

             regul_poisson = 0.0

             print("*** run_warp ***"                  )
             print("structure_type:"  , structure_type  )
             print("deformation_type:", deformation_type)
             print("texture_type:"    , texture_type    )
             print("noise_level:"     , noise_level     )
             print("k_run:"           , k_run           )
             print("mesh_size:"       , mesh_size       )
             print("regul_type:"      , regul_type      )
             print("regul_model:"     , regul_model     )
             print("regul_level:"     , regul_level     )
             print("regul_poisson:"   , regul_poisson   )

             images_basename  = structure_type
             images_basename += "-"+deformation_type
             images_basename += "-"+texture_type
             images_basename += "-noise="+str(noise_level)
             if (n_runs > 1):
                 images_basename += "-run="+str(k_run).zfill(2)

             mesh_folder = images_folder

             mesh_basename  = structure_type
             mesh_basename += "-"+deformation_type
             mesh_basename += "-h="+str(mesh_size)
             if (structure_type == "heart"):
                 mesh_basename += "-mesh"

             working_basename  = images_basename
             working_basename += "-h="+str(mesh_size)
             working_basename += "-"+regul_type
             working_basename += "-regul="+str(regul_level)

             dwarp.warp(
                 working_folder                          = working_folder  ,
                 working_basename                        = working_basename,
                 images_folder                           = images_folder   ,
                 images_basename                         = images_basename ,
                 mesh_folder                             = mesh_folder     ,
                 mesh_basename                           = mesh_basename   ,
                 regul_type                              = regul_type      ,
                 regul_model                             = regul_model     ,
                 regul_level                             = regul_level     ,
                 regul_poisson                           = regul_poisson   ,
                 relax_type                              = "backtracking"  ,
                 normalize_energies                      = 1               ,
```

```
            tol_dU                                        = 1e-2          ,
            n_iter_max                                    = 100           ,
            continue_after_fail                           = 1             ,
            write_VTU_files                               = 1             ,
            write_VTU_files_with_preserved_connectivity = 1               )
```

Visualization

[ ]:
```
structure_type = "square"

deformation_type = "translation"
# deformation_type = "rotation"
# deformation_type = "compression"
# deformation_type = "shear"

texture_type = "tagging"

noise_level = 0.
# noise_level = 0.1
# noise_level = 0.2
# noise_level = 0.3

k_run = 0

mesh_size = 0.1

# regul_type = "continuous-linear-elastic"
# regul_type = "continuous-linear-equilibrated"
# regul_type = "continuous-elastic"
# regul_type = "continuous-equilibrated"
# regul_type = "discrete-simple-elastic"
# regul_type = "discrete-simple-equilibrated"
# regul_type = "discrete-linear-equilibrated"
# regul_type = "discrete-linear-equilibrated-tractions-normal"
# regul_type = "discrete-linear-equilibrated-tractions-tangential"
# regul_type = "discrete-linear-equilibrated-tractions-normal-tangential"
# regul_type = "discrete-equilibrated"
# regul_type = "discrete-equilibrated-tractions-normal"
# regul_type = "discrete-equilibrated-tractions-tangential"
regul_type = "discrete-equilibrated-tractions-normal-tangential"

# regul_level = 0.99
# regul_level = 0.1*2**3
# regul_level = 0.1*2**2
# regul_level = 0.1*2**1
regul_level = 0.1
# regul_level = 0.1/2**1
# regul_level = 0.1/2**2
# regul_level = 0.1/2**3
# regul_level = 0.0

images_basename  = structure_type
images_basename += "-"+deformation_type
images_basename += "-"+texture_type
images_basename += "-noise="+str(noise_level)
if (k_run > 0):
    images_basename += "-run="+str(k_run).zfill(2)

working_basename  = images_basename
working_basename += "-h="+str(mesh_size)
working_basename += "-"+regul_type
working_basename += "-regul="+str(regul_level)

viewer = Viewer(
    images=images_folder+"/"+images_basename+"_*.vti",
```

```
    meshes=working_folder+"/"+working_basename+"_*.vtu")
viewer.view()
```

Plot

```
if (do_plot_disp_error_vs_regul_strength):
 for structure_type, deformation_type in structure_deformation_type_lst:
  for texture_type                    in texture_type_lst              :
   for regul_type                      in regul_type_lst               :

    print("*** plot_disp_error_vs_regul_strength ***")
    print("structure_type:"  , structure_type  )
    print("deformation_type:", deformation_type)
    print("texture_type:"    , texture_type    )
    print("regul_type:"      , regul_type      )

    plot_disp_error_vs_regul_strength(
        images_folder           = images_folder          ,
        sol_folder              = working_folder         ,
        structure_type          = structure_type         ,
        deformation_type        = deformation_type       ,
        texture_type            = texture_type           ,
        regul_type              = regul_type             ,
        noise_level_lst         = noise_level_lst        ,
        n_runs_for_noisy_images = n_runs_for_noisy_images,
        regul_level_lst         = regul_level_lst        ,
        regul_level_for_zero    = 1e-3                   ,
        generate_datafile       = 1                      ,
        generate_plotfile       = 1                      ,
        generate_plot           = 1                      )

    plotfile_basename  = "plot_disp_error_vs_regul_strength"
    plotfile_basename += "/"+structure_type
    plotfile_basename += "-"+deformation_type
    plotfile_basename += "-"+texture_type
    plotfile_basename += "-"+regul_type
    IPython.display.display(IPython.display.Image(filename=plotfile_basename+'.png'))
```

B Code for Figure 5

Imports

```
import dolfin  # https://fenicsproject.org
import IPython # https://ipython.org
import vtk     # https://vtk.org

import dolfin_warp as dwarp # https://gitlab.inria.fr/mgenet/dolfin_warp

from generate_images_and_meshes_from_HeartSlice import generate_images_and_meshes_from_HeartSlice
from plot_disp_error_vs_regul_strength          import plot_disp_error_vs_regul_strength
from lib_viewer                                 import Viewer
```

Parameters

```
images_folder = "generate_images"

n_voxels = 100

deformation_type_lst  = [                  ]
deformation_type_lst += ["contractandtwist"]

texture_type_lst  = [         ]
texture_type_lst += ["tagging"]
```

```
noise_level_lst  = [   ]
noise_level_lst += [0.0]
noise_level_lst += [0.1]
noise_level_lst += [0.2]
noise_level_lst += [0.3]

n_runs_for_noisy_images = 10

working_folder  = "run_warp"

mesh_size_lst  = [   ]
mesh_size_lst += [0.1]

regul_type_lst  = [                                                  ]
regul_type_lst += ["continuous-linear-elastic"                       ]
regul_type_lst += ["continuous-linear-equilibrated"                  ]
regul_type_lst += ["continuous-elastic"                              ]
regul_type_lst += ["continuous-equilibrated"                         ]
regul_type_lst += ["discrete-simple-elastic"                         ]
regul_type_lst += ["discrete-simple-equilibrated"                    ]
regul_type_lst += ["discrete-linear-equilibrated"                    ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal"   ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-tangential"]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal-tangential"]
regul_type_lst += ["discrete-equilibrated"                           ]
regul_type_lst += ["discrete-equilibrated-tractions-normal"          ]
regul_type_lst += ["discrete-equilibrated-tractions-tangential"      ]
regul_type_lst += ["discrete-equilibrated-tractions-normal-tangential"]

regul_level_lst  = [        ]
regul_level_lst += [0.99    ]
regul_level_lst += [0.1*2**3]
regul_level_lst += [0.1*2**2]
regul_level_lst += [0.1*2**1]
regul_level_lst += [0.1     ]
regul_level_lst += [0.1/2**1]
regul_level_lst += [0.1/2**2]
regul_level_lst += [0.1/2**3]
regul_level_lst += [0.0     ]

do_generate_images                  = 1
do_generate_meshes                  = 1
do_run_warp                         = 1
do_plot_disp_error_vs_regul_strength = 1
```

Synthetic images

[ ]:
```
if (do_generate_images):
 for deformation_type in deformation_type_lst:

    print("*** running model ***"              )
    print("deformation_type:", deformation_type)

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model        = 1               ,
        generate_images  = 0               )

    for texture_type in texture_type_lst:
     for noise_level  in noise_level_lst :
```

```
        n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

        for k_run in range(1, n_runs+1):

            print("*** generate_images ***"            )
            print("deformation_type:", deformation_type)
            print("texture_type:"    , texture_type    )
            print("noise_level:"     , noise_level     )
            print("k_run:"           , k_run           )

            generate_images_and_meshes_from_HeartSlice(
                n_voxels         = n_voxels                        ,
                deformation_type = deformation_type                ,
                texture_type     = texture_type                    ,
                noise_level      = noise_level                     ,
                k_run            = k_run if (n_runs > 1) else None,
                run_model        = 0                               ,
                generate_images  = 1                               )
```

Ground truth motion

[ ]:
```
if (do_generate_meshes):
 for deformation_type in deformation_type_lst:
  for mesh_size        in mesh_size_lst       :

    print("*** generate_meshes ***"           )
    print("deformation_type:", deformation_type)
    print("mesh_size:"       , mesh_size       )

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model        = 1               ,
        generate_images  = 0               ,
        mesh_size        = mesh_size       )
```

Tracking

[ ]:
```
if (do_run_warp):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for noise_level      in noise_level_lst     :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run       in range(1, n_runs+1):
     for mesh_size   in mesh_size_lst     :
      for regul_type  in regul_type_lst    :
       for regul_level in regul_level_lst   :

        if any([_ in regul_type for _ in ["linear", "simple"]]):
            regul_model = "hooke"
        else:
            regul_model = "ciarletgeymonatneohookean"

        regul_poisson = 0.3

        print("*** run_warp ***")
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )
```

```
        print("mesh_size:"       , mesh_size       )
        print("regul_type:"      , regul_type      )
        print("regul_model:"     , regul_model     )
        print("regul_level:"     , regul_level     )
        print("regul_poisson:"   , regul_poisson   )

        images_basename  = "heart"
        images_basename += "-"+deformation_type
        images_basename += "-"+texture_type
        images_basename += "-noise="+str(noise_level)
        if (n_runs > 1):
            images_basename += "-run="+str(k_run).zfill(2)

        mesh_folder = images_folder

        mesh_basename  = "heart"
        mesh_basename += "-"+deformation_type
        mesh_basename += "-h="+str(mesh_size)
        mesh_basename += "-mesh"

        working_basename = images_basename
        working_basename += "-h="+str(mesh_size)
        working_basename += "-"+regul_type
        working_basename += "-regul="+str(regul_level)

        dwarp.warp(
            working_folder                                = working_folder  ,
            working_basename                              = working_basename,
            images_folder                                 = images_folder   ,
            images_basename                               = images_basename ,
            mesh_folder                                   = mesh_folder     ,
            mesh_basename                                 = mesh_basename   ,
            regul_type                                    = regul_type      ,
            regul_model                                   = regul_model     ,
            regul_level                                   = regul_level     ,
            regul_poisson                                 = regul_poisson   ,
            relax_type                                    = "backtracking"  ,
            normalize_energies                            = 1               ,
            tol_dU                                        = 1e-2            ,
            n_iter_max                                    = 100             ,
            continue_after_fail                           = 1               ,
            write_VTU_files                               = 1               ,
            write_VTU_files_with_preserved_connectivity = 1               )
```

Visualization

[ ]:
```
deformation_type = "contractandtwist"

texture_type = "tagging"

noise_level = 0.
# noise_level = 0.1
# noise_level = 0.2
# noise_level = 0.3

k_run = 0

mesh_size = 0.1

# regul_type = "continuous-linear-elastic"
# regul_type = "continuous-linear-equilibrated"
# regul_type = "continuous-elastic"
# regul_type = "continuous-equilibrated"
# regul_type = "discrete-simple-elastic"
# regul_type = "discrete-simple-equilibrated"
```

```
# regul_type = "discrete-linear-equilibrated"
# regul_type = "discrete-linear-equilibrated-tractions-normal"
# regul_type = "discrete-linear-equilibrated-tractions-tangential"
# regul_type = "discrete-linear-equilibrated-tractions-normal-tangential"
# regul_type = "discrete-equilibrated"
# regul_type = "discrete-equilibrated-tractions-normal"
# regul_type = "discrete-equilibrated-tractions-tangential"
regul_type = "discrete-equilibrated-tractions-normal-tangential"

# regul_level = 0.99
# regul_level = 0.1*2**3
# regul_level = 0.1*2**2
# regul_level = 0.1*2**1
regul_level = 0.1
# regul_level = 0.1/2**1
# regul_level = 0.1/2**2
# regul_level = 0.1/2**3
# regul_level = 0.0

images_basename  = "heart"
images_basename += "-"+deformation_type
images_basename += "-"+texture_type
images_basename += "-noise="+str(noise_level)
if (k_run > 0):
    images_basename += "-run="+str(k_run).zfill(2)

working_basename  = images_basename
working_basename += "-h="+str(mesh_size)
working_basename += "-"+regul_type
working_basename += "-regul="+str(regul_level)

viewer = Viewer(
    images=images_folder+"/"+images_basename+"_*.vti",
    meshes=working_folder+"/"+working_basename+"_*.vtu")
viewer.view()
```

Plot

[ ]:
```
if (do_plot_disp_error_vs_regul_strength):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for regul_type       in regul_type_lst      :

    print("*** plot_disp_error_vs_regul_strength ***")
    print("deformation_type:", deformation_type)
    print("texture_type:"    , texture_type    )
    print("regul_type:"      , regul_type      )

    plot_disp_error_vs_regul_strength(
        images_folder           = images_folder          ,
        sol_folder              = working_folder         ,
        structure_type          = "heart"                ,
        deformation_type        = deformation_type       ,
        texture_type            = texture_type           ,
        regul_type              = regul_type             ,
        noise_level_lst         = noise_level_lst        ,
        n_runs_for_noisy_images = n_runs_for_noisy_images,
        regul_level_lst         = regul_level_lst        ,
        regul_level_for_zero    = 1e-3                   ,
        generate_datafile       = 1                      ,
        generate_plotfile       = 1                      ,
        generate_plot           = 1                      )

    plotfile_basename  = "plot_disp_error_vs_regul_strength"
    plotfile_basename += "/"+"heart"
```

```
    plotfile_basename += "-"+deformation_type
    plotfile_basename += "-"+texture_type
    plotfile_basename += "-"+regul_type
    IPython.display.display(IPython.display.Image(filename=plotfile_basename+'.png'))
```

C Code for Figure 6

Imports

[ ]:
```
import dolfin  # https://fenicsproject.org
import IPython # https://ipython.org
import vtk     # https://vtk.org

import dolfin_warp as dwarp # https://gitlab.inria.fr/mgenet/dolfin_warp

from generate_images_and_meshes_from_HeartSlice import generate_images_and_meshes_from_HeartSlice
from plot_disp_error_vs_mesh_size               import plot_disp_error_vs_mesh_size
from lib_viewer                                 import Viewer
```

Parameters

[ ]:
```
images_folder = "generate_images"

n_voxels = 100

deformation_type_lst  = [                   ]
deformation_type_lst += ["contractandtwist"]

texture_type_lst = []
texture_type_lst += ["tagging"]

noise_level_lst  = []
noise_level_lst += [0.0]
noise_level_lst += [0.1]
noise_level_lst += [0.2]
noise_level_lst += [0.3]

n_runs_for_noisy_images = 10

working_folder  = "run_warp"

mesh_size_lst  = [        ]
mesh_size_lst += [0.1     ]
mesh_size_lst += [0.1/2**1]
mesh_size_lst += [0.1/2**2]
mesh_size_lst += [0.1/2**3]
mesh_size_lst += [0.1/2**4]

regul_type_lst  = [                                                    ]
regul_type_lst += ["continuous-linear-elastic"                         ]
regul_type_lst += ["continuous-linear-equilibrated"                    ]
regul_type_lst += ["continuous-elastic"                                ]
regul_type_lst += ["continuous-equilibrated"                           ]
regul_type_lst += ["discrete-simple-elastic"                           ]
regul_type_lst += ["discrete-simple-equilibrated"                      ]
regul_type_lst += ["discrete-linear-equilibrated"                      ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal"     ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-tangential" ]
regul_type_lst += ["discrete-linear-equilibrated-tractions-normal-tangential"]
regul_type_lst += ["discrete-equilibrated"                             ]
regul_type_lst += ["discrete-equilibrated-tractions-normal"            ]
regul_type_lst += ["discrete-equilibrated-tractions-tangential"        ]
regul_type_lst += ["discrete-equilibrated-tractions-normal-tangential" ]
```

```
regul_level_lst  = [        ]
regul_level_lst += [0.0     ]
regul_level_lst += [0.1/2**3]
regul_level_lst += [0.1/2**2]
regul_level_lst += [0.1/2**1]
regul_level_lst += [0.1     ]
regul_level_lst += [0.1*2**1]
regul_level_lst += [0.1*2**2]
regul_level_lst += [0.1*2**3]
regul_level_lst += [0.99    ]

do_generate_images                          = 1
do_generate_meshes                          = 1
do_run_warp                                 = 1
do_run_warp_and_refine                      = 1
do_plot_disp_error_vs_mesh_size             = 1
do_plot_disp_error_vs_mesh_size_with_refine = 1
```

Synthetic images

[ ]:
```
if (do_generate_images):
 for deformation_type in deformation_type_lst:

    print("*** running model ***"              )
    print("deformation_type:", deformation_type)

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model        = 1               ,
        generate_images  = 0               )

    for texture_type in texture_type_lst:
     for noise_level  in noise_level_lst :

        n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

        for k_run in range(1, n_runs+1):

            print("*** generate_images ***"           )
            print("deformation_type:", deformation_type)
            print("texture_type:"    , texture_type    )
            print("noise_level:"     , noise_level     )
            print("k_run:"           , k_run           )

            generate_images_and_meshes_from_HeartSlice(
                n_voxels         = n_voxels                        ,
                deformation_type = deformation_type                ,
                texture_type     = texture_type                    ,
                noise_level      = noise_level                     ,
                k_run            = k_run if (n_runs > 1) else None,
                run_model        = 1                               ,
                generate_images  = 0                               )
```

Ground truth motion

[ ]:
```
if (do_generate_meshes):
 for deformation_type in deformation_type_lst:
  for mesh_size        in mesh_size_lst       :

    print("*** generate_meshes ***"           )
    print("deformation_type:", deformation_type)
```

```
    print("mesh_size:"       , mesh_size       )

    generate_images_and_meshes_from_HeartSlice(
        n_voxels         = n_voxels        ,
        deformation_type = deformation_type,
        texture_type     = "no"            ,
        noise_level      = 0               ,
        run_model        = 1               ,
        generate_images  = 0               ,
        mesh_size        = mesh_size       )
```

Tracking (single-level)

[ ]:
```
if (do_run_warp):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for noise_level      in noise_level_lst     :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run       in range(1, n_runs+1):
     for mesh_size   in mesh_size_lst     :
      for regul_type  in regul_type_lst    :
       for regul_level in regul_level_lst   :

        if any([_ in regul_type for _ in ["linear", "simple"]]):
            regul_model = "hooke"
        else:
            regul_model = "ciarletgeymonatneohookean"

        regul_poisson = 0.3

        print("*** run_warp ***"                  )
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )
        print("mesh_size:"       , mesh_size       )
        print("regul_type:"      , regul_type      )
        print("regul_model:"     , regul_model     )
        print("regul_level:"     , regul_level     )
        print("regul_poisson:"   , regul_poisson   )

        images_basename  = "heart"
        images_basename += "-"+deformation_type
        images_basename += "-"+texture_type
        images_basename += "-noise="+str(noise_level)
        if (n_runs > 1):
            images_basename += "-run="+str(k_run).zfill(2)

        mesh_folder = images_folder

        mesh_basename  = "heart"
        mesh_basename += "-"+deformation_type
        mesh_basename += "-h="+str(mesh_size)
        mesh_basename += "-mesh"

        working_basename = images_basename
        working_basename += "-h="+str(mesh_size)
        working_basename += "-"+regul_type
        working_basename += "-regul="+str(regul_level)

        dwarp.warp(
            working_folder                              = working_folder  ,
            working_basename                            = working_basename,
```

```
            images_folder                                = images_folder   ,
            images_basename                              = images_basename ,
            mesh_folder                                  = mesh_folder     ,
            mesh_basename                                = mesh_basename   ,
            regul_type                                   = regul_type      ,
            regul_model                                  = regul_model     ,
            regul_level                                  = regul_level     ,
            regul_poisson                                = regul_poisson   ,
            relax_type                                   = "backtracking"  ,
            normalize_energies                           = 1               ,
            tol_dU                                       = 1e-2            ,
            n_iter_max                                   = 100             ,
            continue_after_fail                          = 1               ,
            write_VTU_files                              = 1               ,
            write_VTU_files_with_preserved_connectivity = 1               ,
            print_iterations                             = 0               )
```

Tracking (multi-level)

[ ]:
```
if (do_run_warp_and_refine):
 for deformation_type in deformation_type_lst:
  for texture_type     in texture_type_lst    :
   for noise_level      in noise_level_lst     :

    n_runs = n_runs_for_noisy_images if (noise_level > 0) else 1

    for k_run       in range(1, n_runs+1):
     for regul_type  in regul_type_lst    :
      for regul_level in regul_level_lst   :

        if any([_ in regul_type for _ in ["linear", "simple"]]):
            regul_model = "hooke"
        else:
            regul_model = "ciarletgeymonatneohookean"

        regul_poisson = 0.3

        print("*** run_warp_and_refine ***"        )
        print("deformation_type:", deformation_type)
        print("texture_type:"    , texture_type    )
        print("noise_level:"     , noise_level     )
        print("k_run:"           , k_run           )
        print("regul_type:"      , regul_type      )
        print("regul_model:"     , regul_model     )
        print("regul_level:"     , regul_level     )
        print("regul_poisson:"   , regul_poisson   )

        images_basename  = "heart"
        images_basename += "-"+deformation_type
        images_basename += "-"+texture_type
        images_basename += "-noise="+str(noise_level)
        if (n_runs > 1):
            images_basename += "-run="+str(k_run).zfill(2)

        mesh_folder = "generate_images"

        mesh_basenames = []
        for mesh_size in mesh_size_lst:
            mesh_basename  = "heart"
            mesh_basename += "-"+deformation_type
            mesh_basename += "-h="+str(mesh_size)
            mesh_basename += "-mesh"

            mesh_basenames += [mesh_basename]
```

```
        working_basename = images_basename
        working_basename += "-"+regul_type
        working_basename += "-regul="+str(regul_level)

        dwarp.warp_and_refine(
            working_folder      = working_folder     ,
            working_basename    = working_basename   ,
            images_folder       = images_folder      ,
            images_basename     = images_basename    ,
            mesh_folder         = mesh_folder        ,
            mesh_basenames      = mesh_basenames     ,
            regul_type          = regul_type         ,
            regul_model         = regul_model        ,
            regul_level         = regul_level        ,
            regul_poisson       = regul_poisson      ,
            relax_type          = "backtracking"     ,
            normalize_energies  = 1                  ,
            tol_dU              = 1e-2               ,
            n_iter_max          = 100                ,
            continue_after_fail = 1                  )
```

Visualization

[ ]:
```
deformation_type = "contractandtwist"

texture_type = "tagging"

noise_level = 0.
# noise_level = 0.1
# noise_level = 0.2
# noise_level = 0.3

k_run = 0

mesh_size = 0.1     ; k_mesh_size = 0
# mesh_size = 0.1/2**1; k_mesh_size = 1
# mesh_size = 0.1/2**2; k_mesh_size = 2
# mesh_size = 0.1/2**3; k_mesh_size = 3
# mesh_size = 0.1/2**4; k_mesh_size = 4

with_refine = 1

# regul_type = "continuous-linear-elastic"
# regul_type = "continuous-linear-equilibrated"
# regul_type = "continuous-elastic"
# regul_type = "continuous-equilibrated"
# regul_type = "discrete-simple-elastic"
# regul_type = "discrete-simple-equilibrated"
# regul_type = "discrete-linear-equilibrated"
# regul_type = "discrete-linear-equilibrated-tractions-normal"
# regul_type = "discrete-linear-equilibrated-tractions-tangential"
# regul_type = "discrete-linear-equilibrated-tractions-normal-tangential"
# regul_type = "discrete-equilibrated"
# regul_type = "discrete-equilibrated-tractions-normal"
# regul_type = "discrete-equilibrated-tractions-tangential"
regul_type = "discrete-equilibrated-tractions-normal-tangential"

# regul_level = 0.99
# regul_level = 0.1*2**3
# regul_level = 0.1*2**2
# regul_level = 0.1*2**1
regul_level = 0.1
# regul_level = 0.1/2**1
# regul_level = 0.1/2**2
# regul_level = 0.1/2**3
```

```
# regul_level = 0.0

images_basename  = "heart"
images_basename += "-"+deformation_type
images_basename += "-"+texture_type
images_basename += "-noise="+str(noise_level)
if (k_run > 0):
    images_basename += "-run="+str(k_run).zfill(2)

working_basename  = images_basename
if not (with_refine):
    working_basename += "-h="+str(mesh_size)
working_basename += "-"+regul_type
working_basename += "-regul="+str(regul_level)
if (with_refine):
    working_basename += "-refine="+str(k_mesh_size)

viewer = Viewer(
    images=images_folder+"/"+images_basename+"_*.vti",
    meshes=working_folder+"/"+working_basename+"_*.vtu")
viewer.view()
```

Plot

[ ]:
```
if (do_plot_disp_error_vs_mesh_size) or (do_plot_disp_error_vs_mesh_size_with_refine):

    with_refine_lst = []
    if (do_plot_disp_error_vs_mesh_size            ): with_refine_lst += [False]
    if (do_plot_disp_error_vs_mesh_size_with_refine): with_refine_lst += [True ]

    for with_refine      in with_refine_lst     :
     for deformation_type in deformation_type_lst:
      for texture_type     in texture_type_lst    :
       for regul_type       in regul_type_lst      :

        plot_disp_error_vs_mesh_size(
            images_folder           = images_folder          ,
            sol_folder              = working_folder         ,
            structure_type          = "heart"          ,
            deformation_type        = deformation_type       ,
            texture_type            = texture_type           ,
            regul_type              = regul_type             ,
            noise_level_lst         = noise_level_lst        ,
            n_runs_for_noisy_images = n_runs_for_noisy_images,
            regul_level_lst         = regul_level_lst        ,
            mesh_size_lst           = mesh_size_lst          ,
            error_for_nan           = 10                     ,
            with_refine             = with_refine            ,
            generate_datafile       = 1                      ,
            generate_plotfile       = 1                      ,
            generate_plot           = 1                      )

        plotfile_basename  = "plot_disp_error_vs_mesh_size"
        if (with_refine):
            plotfile_basename += "-with_refine"
        plotfile_basename += "/"+"heart"
        plotfile_basename += "-"+deformation_type
        plotfile_basename += "-"+texture_type
        plotfile_basename += "-"+regul_type
        IPython.display.display(IPython.display.Image(filename=plotfile_basename+'.png'))
```