Simulation of hyperelastic materials in real-time using deep learning

Andrea Mendizabal, Pablo Márquez-Neila, Stéphane Cotin

▶ **To cite this version:**

Andrea Mendizabal, Pablo Márquez-Neila, Stéphane Cotin. Simulation of hyperelastic materials in real-time using deep learning. 2019. hal-02097119v1

HAL Id: hal-02097119

https://inria.hal.science/hal-02097119v1

Preprint submitted on 11 Apr 2019 (v1), last revised 8 Nov 2019 (v3)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SIMULATION OF HYPERELASTIC MATERIALS IN REAL-TIME USING DEEP LEARNING

A PREPRINT

Andrea Mendizabal
Inria, Strasbourg, France
University of Strasbourg, ICube, Strasbourg, France
andrea.mendizabal@inria.fr

Pablo Márquez-Neila
ARTORG Center, University of Bern, Switzerland

Stéphane Cotin
Inria, Strasbourg, France

April 10, 2019

ABSTRACT

The finite element method (FEM) is among the most commonly used numerical methods for solving engineering problems. Due to its computational cost, various ideas have been introduced to reduce computation times, such as domain decomposition, parallel computing, adaptive meshing, and model order reduction. In this paper we present U-Mesh: a data-driven method based on a U-Net architecture that approximates the non-linear relation between a contact force and the displacement field computed by a FEM algorithm. We show that deep learning, one of the latest machine learning methods based on artificial neural networks, can enhance computational mechanics through its ability to encode highly non-linear models in a compact form. Our method is applied to two benchmark examples: a cantilever beam and an L-shape subject to moving punctual loads. A comparison between our method and proper orthogonal decomposition (POD) is done through the paper. The results show that U-Mesh can perform very fast simulations on various geometries, mesh resolutions and number of input forces with very small errors.

Keywords Real-time simulation, Deep neural network, Physics-based simulation, Finite element method, Hyperelasticity, Model reduction

1 Introduction

There are many applications in engineering where the deformation of non-linear structures needs to be simulated in real time, or would benefit from being computed interactively. Some important examples can be found in the field of medicine, in order to develop training systems for learning surgical skills [1] or in the field of surgical navigation, where augmented reality combined with interactive simulations can bring significant improvements to clinical practice [2]. Medical robotics, involving flexible robots or interactions with soft tissues, is another important area where real-time simulation of flexible structures is essential, in order here to have a better control of the robot.

While there are different numerical strategies for solving equations associated with elastic materials, we only consider the finite element method (FEM) in this article, for its accuracy and ability to simulate a large range of materials on potentially complex domains. However, obtaining real-time simulations with this method, in particular when considering non-linear materials, becomes a real challenge, in particular if this has to be done on consumers level hardware rather than a high-end parallel computer.

In order to speed up FEM simulations, several techniques have been proposed. We review here only some of the main ideas that were proposed. First, since solving the system of equations resulting from the FEM discretization is usually the bottleneck of the computation, many works have focused on linear solvers. Domain decomposition methods are based on the divide & conquer paradigm. Such methods consist in splitting the global problem domain into smaller independent sub-domains, making the approach suitable for parallel computing. This allows to build efficient preconditioners even though additional computation is required to synchronize the solution be-

tween neighboring sub-domains. Under the right conditions, in particular if the number of processors of the computer matches the number of sub-domains, a superlinear speedup can be obtained [3]. This is, however, impossible to achieve when considering problems (even if relatively small) which need to be solved in real-time on consumers level hardware. This is mainly due to the limited number of cores (only 10 cores on the latest Intel i9 processor) and communication costs which are significant compared to the expected computation times (about 50 ms per time step for an interactive simulation).

Another option for speeding up simulation times is to lower the computational complexity of the problem through a reduction of the model's degrees of freedom. Depending on the problem, and acceptable loss of accuracy, it is possible to obtain speedups of several orders of magnitude. Proper Orthogonal Decomposition (POD) is one of the main model order reduction methods. POD techniques compute off-line the solution to several complete models and extract the modes that describe best the solution to the complete problem. Based on a priori knowledge of the solution, it encodes the high dimensional problem in a smaller subspace defined by a truncated basis of singular vectors. The dimension of such basis determines the ratio between accuracy of the method and computation times (the smaller the basis, the larger the error). In the context of real-time simulation of non-linear solids, several examples of POD have been proposed. Niroomandi *et al.* [4] proposed a POD method to simulate the palpation of the cornea. Haptic feedback rates were achieved, but with a relative error of about 20%. If more accurate solutions are needed, the number of modes used in the POD can be increased at the cost of higher computation times. Thus, to keep the method numerically efficient in the case of non linear materials, hyperreduction can be used in order to further reduce the computation times while reducing the error [5]. Goury *et al.* [6] applied the hyperreduced POD to control and simulate soft robots with very good accuracy in $25\ ms$ per time step. However the POD may be in some cases insufficient to capture correctly the high degrees of non linearity that can be found for example in biological soft tissues as it relies on a linear combination of few basis vectors [7]. To precisely account for non-linearities it might be necessary to recompute the entire stiffness matrix which is burdensome and not always possible[8]. Another model order reduction algorithm is the proper generalized decomposition (PGD), which, contrarily to POD, builds a reduced-order approximation without relying on the knowledge of the solution of the complete problem. PGD assumes that the solution of a multiparametric problem can be expressed as a sum of separable functions that are constructed by successive enrichment by invoking the weak form of the considered problem. An approach based on PGD is proposed in [9] for the simulation of hyperelastic soft tissue deformation at high frequency. However, when the solution is non-separable, PGD offers no particular advantage over classical FEM techniques.

A last class of worth mentioning solutions consists in using the Graphics Processing Unit (GPU) as a particular type of parallel machine. Although each core of the GPU is very limited in its computational performance, the very high number of cores available (several thousand) makes it possible to obtain significant speedups on computationally heavy problems. For instance, NiftySim [10] is a GPU-based non-linear finite element toolkit for the simulation of soft tissue biomechanics where speedups of 300x are obtained. SOFA [11] is an Open-source Framework focused on real-time simulation of complex interactions with deformable structures, which provides GPU-compatible FEM codes [12]. Based on a Total Lagrangian Explicit Dynamics algorithm from Miller *et al.*, [13], a speedup of more than 50x is reported.

Recently, machine learning started to revolutionize several fields (vision, language processing, image recognition, genomics) due to the continuously increasing amount of data available and the development of new algorithms and powerful GPUs. Deep learning, a class of machine learning methods based on learning data representations, as opposed to task-specific algorithms, has demonstrated strong abilities at extracting high-level representations of complex processes. With sufficient ground truth data, machine learning algorithms can map the input of a function to its output without any mathematical formulation of the problem, thus actuating like a black box. Since FEM can provide as much noise-free data as required, it seems interesting to train learning algorithms with such virtually generated data [14][15][16]. For example authors in [14] proposed a machine learning approach for modeling the mechanical behavior of the liver during breathing in real-time. They trained several regression models using the external displacement and the elasticity parameters as input and the FEM-based nodal displacements as output. Although they reached good accuracy their method is bounded to the small displacement regime. Authors in [16] proposed a preliminary work using a deep-autoencoder to approximate the deformations of a non-linear muscle actuated beam. They showed that their method produces lower reconstruction errors than the equivalently sized PCA model. However the computational gain of their method is not clear and it was limited to a very simple model and coarse mesh.

The objective of our work, which preliminary results are presented in this article, is to go beyond the state of the art on machine learning applied to computational mechanics. In particular we show that we can predict, in real-time, the shape of a non-linear elastic structure with a very good accuracy using a deep network inspired by model order reduction methods. Our solution relies on a U-Net architecture trained on FEM-generated data sets; both aspects are presented in section 2 while results and their comparison against a model order reduction algorithm are presented in section 3 and 4.

2 Method

As mentioned in the introduction, an important area of applications for real-time simulation of non-linear material is in the field of computer-aided surgery. In this context, accuracy is also very important but often left aside for the sake of rapidity. Achieving a better trade-off between accuracy and computation time is therefore mandatory to tackle more ambitious problems. This requirement for both accuracy and very fast computation can find applications in other areas of mechanical engineering, such as training of complex industrial processes [17] or virtual prototyping [18] just to name a few.

In this paper, we propose a method that does not need such a compromise. It allows for extremely fast and accurate simulations by using an artificial neural network that partially encodes the stress-strain relation in a low-dimensional space. Such a network can learn the desired biomechanical model, and predict deformations at haptic feedback rates with very good accuracy. This section is divided in three main segments. First the problem that we aim to solve is presented, with the corresponding modeling and discretization choices. Then, the selected network architecture is detailed, followed by our strategy to encode the stress-strain relationship and boundary conditions into the network and the data set generation used to train the network.

2.1 Mechanical formulation of the problem and offline numerical resolution

Without lack of generality, we consider the boundary value problem of computing the deformation of a hyperelastic material under both Dirichlet and Neumann boundary conditions. The solid occupies a volume Ω whose boundary is Γ. We assume the Dirichlet conditions on Γ_D known a priori, while Neumann boundary conditions on Γ_N can change at any time step. We consider a Lagrangian description of the deformation whose material coordinates are given by the vector $\mathbf{X}$. The deformed state of each point of the solid is given by

$$\mathbf{x} = \mathbf{X} + \mathbf{u} \tag{1}$$

where $\mathbf{u}$ is the displacement field. We propose to describe the linear relation between the stress and the strain using a Saint-Venant-Kirchhoff constitutive model. The Green-Lagrange strain tensor $\mathbf{E} \in \mathbf{R}^{3\times3}$ is computed as a non-linear (quadratic) function of the deformation gradient $\mathbf{F} = \mathbf{I} + \nabla_{\mathbf{X}}\mathbf{u}$:

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{I}) \tag{2}$$

where $\mathbf{I} \in \mathbf{R}^{3\times3}$ is the identity matrix. The strain-energy density function $\mathbf{W}$ is obtained according to the following equation:

$$\mathbf{W}(\mathbf{E}) = \frac{\lambda}{2}[tr(\mathbf{E})]^2 + \mu tr(\mathbf{E}^2) \tag{3}$$

where λ and μ are the Lame's constants derived from the Young's modulus Y and the Poisson's ratio ν.

Then the second Piola-Kirchhoff stress $\mathbf{S}$ reads as follows:

$$\mathbf{S} = \frac{\partial \mathbf{W(E)}}{\partial \mathbf{E}} = \mathbf{C} : \mathbf{E} \tag{4}$$

where $\mathbf{C}$ is the fourth-order constitutive tensor. $\mathbf{S}$ is related to the first Piolat tensor $\mathbf{P}$ by $\mathbf{P} = \mathbf{FS}$.

Ignoring time-dependant terms, the boundary value problem is then given in material coordinates by :

$$\begin{cases} \nabla(\mathbf{FS}) = \mathbf{b}\ in\ \Omega \\ \mathbf{u}(\mathbf{X}) = 0\ on\ \Gamma_D \\ (\mathbf{FS})\mathbf{n} = \mathbf{t}\ on\ \Gamma_N \end{cases} \tag{5}$$

where $\mathbf{b}$ is the external body force, $\mathbf{n}$ is the unit normal to Γ_N and $\mathbf{t}$ is the traction applied to the boundary. The weak form of (5), obtained from the principle of virtual work, brings forward the boundary term and reads as:

$$\int_\Omega (\mathbf{FS}) : \delta\mathbf{E}\ d\Omega = \int_\Omega \mathbf{b}\eta\ d\Omega + \int_{\Gamma_N} \mathbf{t}\eta\ d\Gamma \tag{6}$$

where $\delta\mathbf{E} = \frac{1}{2}(\mathbf{F}^T\nabla\eta + \nabla^T\eta\mathbf{F})$ is the variation of the strain, and $\eta = \{\eta \in H^1(\Omega) \mid \eta = 0 \text{ on } \Gamma_D\}$ is any vector-valued test function. The left side of equation (6) denotes the internal virtual work, and the right side, the virtual work from the applied external load.

Finite element simulation Equation (6) is solved using a finite element method. The domain was discretized in hexahedral (H8) elements, providing a set of unknown displacements at the element nodes. This choice is not only motivated by the good convergence and stability of such elements: hexahedral elements are also required for our convolutional neural network (see section 2.2).

Due to the non-linearity of equation (2), we need to solve a non-linear system of equations to approximate the unknown displacement. Using an iterative Newton-Raphson method, from an initial displacement $\mathbf{u}^0$, we try to find a correction δ_u^n after n iterations that balances the linearized set of equations:

$$\dot{\mathbf{K}}^{n-1}\delta_u^n = \mathbf{r}(\mathbf{u}^0 + \delta_u^{n-1}) + \mathbf{b} \tag{7}$$

where $\dot{\mathbf{K}}$ is the tangent stiffness matrix and $\mathbf{r}$ is the internal elastic force vector. Here, at each iteration, both the matrix $\dot{\mathbf{K}}$ and the vector $\mathbf{r}$ need to be computed, and the linear system needs to be solved. Since the convergence of the Newton-Raphson method is only valid for a displacement $\mathbf{u}^0$ near the solution, large external loads must be applied by small increments and can require a large number of iterations to converge. This is an important characteristic in order to evaluate the performance of the proposed neural network on a non-linear model. Depending on the mesh resolution, solving this problem can be extremely long even for very sophisticated and optimized codes. Dimensionality reduction techniques have shown real benefits in speeding-up FEM simulations. Among

them, POD is a very popular one since it leads to very realistic real-time simulations. Although authors in [4] have shown good results using this method for large deformation, POD is better suited for linear or weakly non-linear processes. We propose a non-linear dimensionality reduction technique using a neural network to learn the correspondence from contact forces to volumetric deformations of a given mesh.

2.2 Deep neural network for online prediction of the displacement field

Formally, our network h is a parameterized function that accepts a $3 \times n_x \times n_y \times n_z$ tensor $\mathbf{f}$ as input and produces a tensor $\mathbf{u}$ of the same size as output. The domain Ω is sampled by a 3-dimensional grid of resolution $n_x \times n_y \times n_z$. Practically speaking, the nodes of this grid match the nodes of the FEM mesh, although this is not required and an interpolation could be used instead. The tensor $\mathbf{f}$ represents the contact forces applied to the mesh, and the tensor $\mathbf{u}$ contains the corresponding displacement of the mesh. In particular, each vector $\mathbf{f}[:, i, j, k]$ represents the force vector (f_x, f_y, f_z) applied over the node (i, j, k) of the grid. Similarly, the vector $\mathbf{u}[:, i, j, k]$ represents the displacement (u_x, u_y, u_z) of the node (i, j, k).

Our problem consists of finding the function f that produces the best estimations of the displacement field given some contact forces (traction). This is performed by minimizing the expected error over the distribution $\mathcal{D}$ of pairs $(\mathbf{f}, \mathbf{u})$,

$$\min_{\theta} \mathbb{E}_{(\mathbf{f},\mathbf{u})\sim\mathcal{D}} \left[\|h(\mathbf{f}) - \mathbf{u}\|_2^2 \right], \tag{8}$$

where θ is the set of parameters of the network h. In practice, the expectation of Eq. (8) is approximated by Monte-Carlo sampling with a training set $\{(\mathbf{f}_n, \mathbf{u}_n)\}_{n=1}^N$ of N samples:

$$\min_{\theta} \frac{1}{N} \sum_{n=1}^{N} \|h(\mathbf{f}_n) - \mathbf{u}_n\|_2^2. \tag{9}$$

We build our training set by randomly applying forces on the mesh and running FEM simulations to produce corresponding displacements.

Let us characterize the architecture chosen for our network h. We propose to use the U-Net [19], a modified fully convolutional network initially built for precise medical image segmentation. As depicted in Fig. 1, the network is similar to an auto-encoder, with an encoding path to transform the input space into a low-dimensional representation, and a decoding path to expand it back to the original size. Additional skip connections transfer detailed information along matching levels from the encoding path to the decoding path.

The encoding path consists of k sequences of two padded $3 \times 3 \times 3$ convolutions ($k = 4$ in [19]) and a $2 \times 2 \times 2$ max pooling operation (see Fig. 1). Intuitively, each convolution kernel learns to isolate the different characteristics

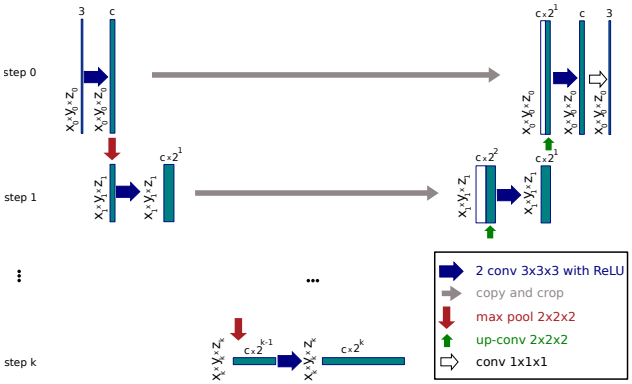

Figure 1: General network architecture for an object with a resolution of $x \times y \times z$ nodes, c channels in the first layer and k steps. Note that $x_0 \times y_0 \times z_0$ is the padded grid.

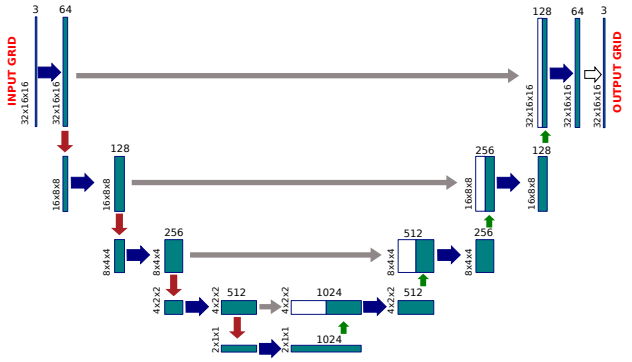

Figure 2: Network architecture for a beam with $28 \times 12 \times 12$ nodes, padded to $32 \times 16 \times 16$, 64 channels in the first layer and 4 steps (see Fig. 1 for notations).

of the displacement field (orientation, direction, amplitude). At each step, each feature map doubles the number of channels and halves the spatial dimensions. We assume that the number of channels is directly related to the amount of detectable variations in the displacement field. In the bottom part there are two extra $3 \times 3 \times 3$ convolutional layers leading to a $(c \times 2^k)$-dimensional array. This feature space is similar to the reduced space in POD where the order p of the singular vector truncation is the counterpart for the number of units in the code layer. In a symmetric manner, the decoding path consists of k sequences of an upsampling $2 \times 2 \times 2$ transposed convolutions followed by two padded $3 \times 3 \times 3$ convolutions. The features from the encoding path at the same stage are cropped and concatenated to the upsampled feature maps. At each step of the decoding path, each feature map halves the number of channels and doubles the spatial dimensions. There is a final $1 \times 1 \times 1$ convolutional layer to transform the last feature map to the desired number of channels of the output (3 channels in our case).

The number of steps k and the number of channels c control the accuracy of the prediction just like the number of

singular vectors in POD. Higher values of k and c lead to a more complex network suitable for difficult meshes at the expenses of longer computational times for both training and prediction, and higher requirements of training data. We tested several values for k and c in order to select the most appropriate values for our experiments depending on the desired accuracy and the eventual time restrictions.

2.3 Data set generation for U-Net training

In order to train such a network, we build a data set of pairs $(\mathbf{f}, \mathbf{u})$ obtained with the previously explained FEM. Once we are given a 3D mesh with its corresponding constitutive law, material properties and boundary conditions, we perform multiple simulations by applying random forces to nodes of the object. After each simulation, the pair of applied forces and obtained deformation is stored as an element of the data set. To speed up the generation of the training and testing data sets, the linearized system of equation (7) is solved using an iterative preconditioned conjugate gradient method [20].

The variability of the data relies on the force magnitude, its direction and its application point. In this work, a force is applied on a local surface area Γ_N which location varies such that the boundary of the computation domain is completely covered. The force direction is uniformly sampled on the unit sphere and the force magnitude is a uniform random value between 0 and 1. At each sample of the data set, one force is applied on a small region Γ_N. There are Λ samples for each Γ_N, meaning that Λ different forces are applied per region. We can consider one force per sample or several forces applied simultaneously at different locations.

Training The network is trained minimizing Eq. (9) with training data generated as explained above. The minimization is performed using the Adam optimizer [21], a stochastic gradient descent procedure with parameter-wise adjusted learning rates.

3 Results

In this section, we perform a model selection over the space of hyperparameters k and c in order to find the combination that leads to the best results in a cantilever beam. Then we apply our method, with the selected hyperparameters, to two benchmark examples: a cantilever beam and an L-shaped object under point loads. All our experiments are performed in a GeForce 1080 Ti using a batch size of 4 and 100000 iterations for training. We use a PyTorch implementation of the U-Net. We recall that the batch size is the number of samples that are given to the network at each iteration of the minimization process.

3.1 Validation metrics

To assess the efficiency of our method, we perform a statistical analysis of the *mean norm error* e over a testing data set $\{(\mathbf{f}_m, \mathbf{u}_m)\}_{m=1}^{M}$, which is built similarly as the training data set. Note that the training data set and the testing data set are disjoint. Let $\mathbf{u}_m$ be the ground truth displacement tensor for sample m generated using the FEM described in section 2.1 and $h(\mathbf{f}_m)$ the U-Mesh prediction. The mean norm error between $\mathbf{u}_m$ and $h(\mathbf{f}_m)$ for sample m reads as:

$$e(\mathbf{u}_m, h(\mathbf{f}_m)) = \frac{1}{n}\sum_{i=1}^{n} |\mathbf{u}_m^i - h(\mathbf{f}_m)^i|. \qquad (10)$$

where n is the number of degrees of freedom of the mesh. We compute the average $\overline{e}$ and standard deviation $\sigma(e)$ of such norm over the testing data set:

$$\overline{e} = \frac{1}{M}\sum_{m=1}^{M} e(h(\mathbf{f}_m), \mathbf{u}_m), \qquad (11)$$

and

$$\sigma(e) = \sqrt{\frac{1}{M-1}\sum_{m=1}^{M} [e(h(\mathbf{f}_m), \mathbf{u}_m) - \overline{e}\,]^2}. \qquad (12)$$

3.2 U-Mesh applied to a cantilever beam

We consider a deformable beam of size $4 \times 1 \times 1\, m^3$ subjected to fixed boundary on one end. The beam follows the Saint-Venant-Kirchhoff behavior described in section 2 with a Young's modulus Y of 500 Pa and a Poisson's ratio of 0.4, and is discretized with 135 H8 elements. To generate the data set, 100 forces are applied on each of the 64 nodes of the upper face (that is $\Lambda = 100$), up to a total of 6400 samples. 80% of the samples are used for training ($N = 5120$) and the remaining 20% are kept for testing ($M = 1280$). Using this data set, we select the best machine learning model by training several U-Net architectures with different combinations of hyperparemeters k and c. In Table 1 are reported the training and prediction times as well as $\overline{e}$ and $\sigma(e)$. As seen in this table, the higher the feature space size (FSS), the lower the errors. Prediction time is proportional to the depth of the network. Choosing the best model is a tradeoff between network performance and speed, and we will show that the selected hyperparameters lead to good results also on different problems. Choosing $k = 3$ and $c = 128$ seems to be a good compromise for our needs. The selected parameters are framed within a red box in Table 1. For the selected set of parameters, e over the data set is equal to $0.0007 \pm 0.0006\, m$ for a maximal deformation over the testing data set of $0.724\, m$. In Fig. 4, we show the sample with maximal error. We perform a sensitivity analysis of the method to the amplitude of deformation. The results are plotted in Fig. 3. We perform a least squares line fit to find the relation between the maximal deformation and the mean norm error e. We can observe a very small sensitivity of the error e with respect to the deformation amplitude, and a very small error in the estimation of the displacement field in general. This shows an important

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
64	2	256	0.0028	0.0008	2	24
16	4	256	0.0012	0.0009	3.2	40
32	4	512	0.0009	0.0008	3.2	40
64	3	512	0.0007	0.0007	2.5	80
64	4	1024	0.0007	0.0007	3.3	95
128	3	1024	0.0007	0.0006	2.5	35
64	5	2048	0.0006	0.0005	4	426
128	4	2048	0.0006	0.0005	3.45	320

Table 1: Error measures for a beam having 135 H8 elements. Rows are sorted in decreasing $\overline{e}$. The selected architecture is framed within a red box.

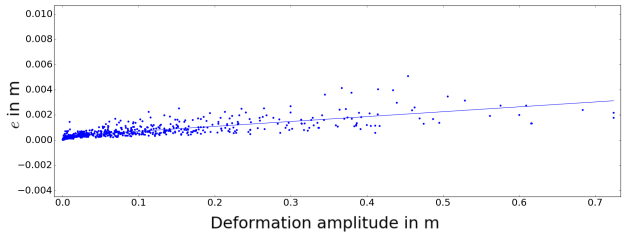

Figure 3: The point cloud represents the e for some randomly selected samples of the testing data set. The regression line of equation $y = 0.00352 \times x$ shows the low sensitivity of the U-Mesh to the deformation amplitude.

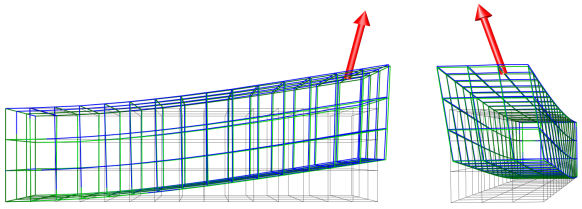

Figure 4: Front and side views of the maximal error sample obtained with a network having 3 steps and 128 channels. U-Mesh output is in blue and the reference is in green. The red arrow represents the force applied. Note that the difference between the two meshes is very small. The relative $l2$ norm at the tip of the beam is of 5.1% and the deformation amplitude is $0.45\,m$

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
64	4	1024	0.0007	0.0003	3.5	100

Table 2: Error measures for a beam having 135 H8 elements and three simultaneous force application. Rows are sorted in decreasing $\overline{e}$.

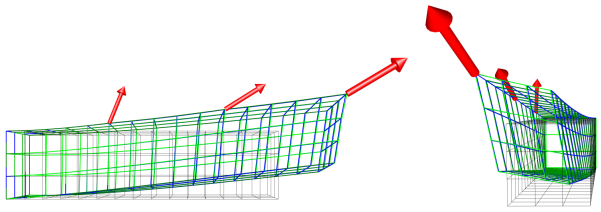

Figure 5: Largest deformation for 3 simultaneous force applications (side and front view for the same sample). U-Mesh output is in blue, the reference is in green and the rest shape is in grey.

characteristic of our method, in addition to its very limited computational cost.

In Table 2 are shown the prediction errors for 3 simultaneous input forces and their corresponding training and prediction times. In this scenario, the Young's modulus is set to $400\ Pa$ and three forces are applied simultaneously on three different regions of the upper face of the beam. In order to avoid mechanical coupling, such regions must be far enough from each other. There are 12360 possible combinations of nodes fulfilling the above stated condition. Only one force ($\Lambda = 1$) is applied per viable combination of nodes. Hence the data set has 12360 samples ($N = 9888$ samples for training and $M = 2472$ samples for testing). We can see that the errors and the time needed for the prediction are comparable to that needed for only one force application. It is important to note that there are 2x more samples in this data set (than in previous ones) due to the large number of possible combinations. This explains the particularly low error in this case. The sample with maximal deformation ($1.0035\ m$) is shown in Fig. 5. At the tip of the beam, the relative $l2$ norm is equal to 1.5%.

So far we have seen that U-Mesh is able to predict the displacement field due to one or several forces with high accuracy and in an extremely short amount of time. Nevertheless, FEM codes are also able to compute the solutions on such meshes in limited computation times, even for hyperelastic models. Hence, in order to put ahead the real contribution of our work, we test our method on a computationally expensive problem.

We consider the same beam as previously but this time discretized in 3267 H8 elements (see Fig. 7). There are 336 nodes on the upper face and Λ is set to 100. Overall the data set has 33600 samples ($N = 26880$ samples for training and $M = 6720$ samples for testing). The

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
128	3	1024	0.0019	0.0018	3	210

Table 3: Error measures for a beam having 3267 H8 elements and one simultaneous force application.

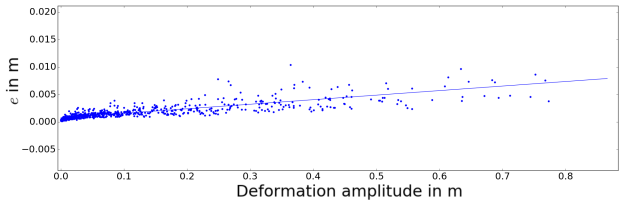

Figure 6: Sensitivity of e to the deformation amplitude for a 3267 H8-elements beam. The point cloud represents the values of e for randomly selected samples of the testing data set. The regression line of equation $y = 0.0083 \times x$ shows the low sensitivity of the U-Mesh to the deformation range. The average computation time is $3\,ms$.

U-Net is trained with the previously selected parameters, that is, 128 channels for the first layer and 3 steps. The metrics computed over the testing data set are shown in Table 3. The most interesting result is the low prediction time (only $3\,ms$). A very optimized version of the Saint-Venant-Kirchhoff FEM using a Pardiso solver [22] that is among the most efficient solvers available, takes more than $300\,ms$ to solve such simulation. The speedup obtained with U-Mesh is of 100x. All the samples of the testing data set have an error bellow $0.0265\,m$ and an average error of $0.0019\,m$ for a maximal deformation of $1.011\,m$.

3.3 U-Mesh applied to an L-shaped object

In this section we will show that U-Mesh generalizes well on other geometries. We apply U-Mesh on an L-shape of size $28.424 \times 10 \times 40\ m^3$ discretized with 335 H8 elements. Since U-Net requires a regular grid as input, we added zeros to fill the L-shaped mesh so that it is included in a regular grid. The Young's modulus is equal to $100\ Pa$

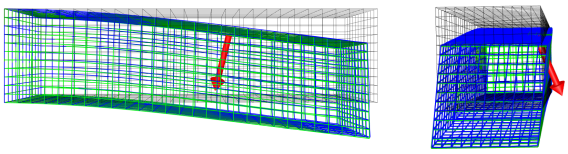

Figure 7: Beam mesh with 3267 H8 elements deformed with U-Mesh, front and side views. U-Mesh output is in blue, the reference is in green and the rest shape is in grey. The computation time is equal to $2.9\,ms$ for this sample. The relative $l2$ norm at the tip of the beam is 1.6% and the deformation amplitude is $0.4\,m$

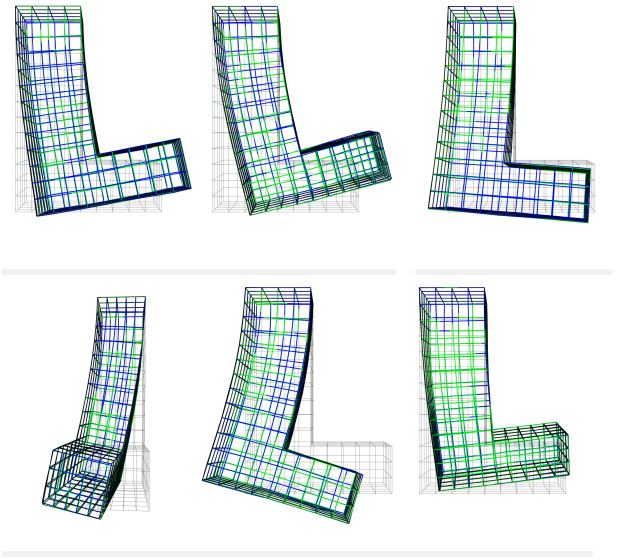

Figure 8: Samples of deformed L-shapes. In green is shown the reference solution, in blue the output of U-Mesh and in grey the rest shape.

c	k	FSS	$\overline{e}$ in m	$\sigma(e)$ in m	pred_t in ms	train_t in min
64	4	1024	0.0433	0.0281	3.2	95

Table 4: Error measures for the L-shape.

and the Poisson's ratio is 0.4. To build the data set, external forces ranging from 0 to $40\,N$ are applied on the bottom face of the L. We set Λ to 100. Overall the data set contains 6,000 samples ($N = 4,800$ and $M = 1,200$). We train a U-Net with 64 channels in the first layer and 4 steps. In Fig. 8 are shown some samples of deformed L-shapes. The U-Mesh output is in blue and the reference solution is in green. In order to perceive the amount of deformation, the rest position is also shown (thin grey lines). The average and maximal errors are given in Table 4, and the prediction times are in the same range as for the beam scenario. The average error is equal to $0.0433\,m$ where the maximal deformation is $15.15\,m$. The slope of the regression in Fig. 9 shows that the increase of the error with the deformation amplitude is controlled.

Overall, we have seen that U-Mesh predicts deformations for different geometries and mesh resolutions, with small controlled errors, and in a time always smaller than $4\,ms$.

4 Discussion

4.1 Comparison of U-Mesh and POD

In this section we compare the predictions made by U-Mesh to the simulations computed on a reduced model using POD. We used the POD code available at `https://`

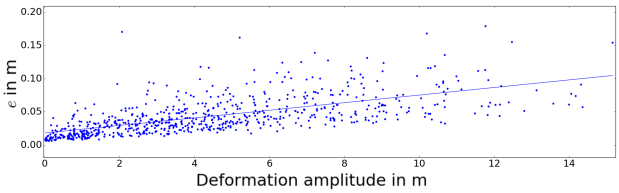

Figure 9: Sensitivity of e to the deformation amplitude for the L-shape. The point cloud represents the e all the samples of the testing data set. The regression line of equation $y = 0.005716 \times x$ shows the low sensitivity of the U-Mesh to the deformation range.

`github.com/SofaDefrost/ModelOrderReduction` that works as a plugin of the SOFA framework [11]. The POD consists in three phases. First, an offline phase where all the potential movements of the beam are sampled and stored in the so-called snapshot. This offline phase is the equivalent to the data generation phase in U-Mesh and is also computationally intensive since it performs many fine simulations. In a second phase, the snapshot space is condensed in a reduced basis using singular value decomposition and a truncation [6]. This phase "corresponds" to the training of the U-Net and is generally faster. Finally, the resulting reduced model allows for faster simulations since there are fewer degrees of freedom. We applied the simple POD and the hyperreduced-POD (HPOD) to the fine beam depicted previously (see Fig. 7) in order to compare the performance of POD and U-Mesh. Note that the hyperreduced version keeps the same number of modes as the POD.

We first compare the computation times for a given accuracy. The truncature error of the POD was set such that the mean norm error obtained with POD is similar to the one obtained with U-Mesh. To reach the desired accuracy in the considered deformation range, 124 modes were preserved. The computation times are reported in table 5. On the one hand, with the selected number of modes, POD is 1.25 times faster than the full FEM model and HPOD is 12.5 times faster. The cost of the hyperreduction is a lost on the accuracy of more than 40%. On the other hand, U-Mesh is more than 200 times faster than the full FEM model.

Let us now compare the relative errors of the two methods for a given computation time. The fastest version of the reduced model is the one considering only one deformation mode and using hyperreduction. As presented in table 6, HPOD can compute deformations in $4\,ms$ but with an error that is almost 10 times larger than the one produced by U-Mesh.

4.2 Current limitations

With the current U-Mesh, it is not possible to make very accurate predictions if we apply a force somewhere out of the training data set. This limitation is also inherent to POD. In the same manner, we are restricted to the geometry used to train the network.

	Prediction time in s	e in m
FEM	0.625	0.000
POD	0.476	0.006
HPOD	0.053	0.031
U-Mesh	0.003	0.006

Table 5: Computation times of the full FEM model, of the POD with and without hyperreduction and of the U-Mesh, for a similar mean norm error e.

	Prediction time in s	Relative error in $\%$
FEM	0.625	0.00
HPOD	0.004	0.056
U-Mesh	0.003	0.006

Table 6: The mean norm error of the full FEM model, of the hyperreduced POD and of the U-Mesh are given, for a computation time in the range of the millisecond.

Another limitation of the method is the expensive offline phase. The data generation can be extremely time consuming in particular when considering large meshes or when more complex input sequences are needed. It goes without saying that the larger the data sets, the longer the training. There is however room for improvement in order to reduce these effects by performing intelligent samplings for the data set generation, such as Latin-Hypercube sampling. Moreover, we performed a little study on the sensitivity of the method to the Young's modulus variations in the training data set. We noticed that U-Mesh is more accurate for stiffer objects. This is an important result to keep in mind for the future *smart data generation.* Indeed, in a low deformation regime less data is needed to reach good accuracy.

5 Conclusion

In this paper we have proposed a U-Net architecture that can learn the relation function between an input force and an output deformation for various geometries and make predictions with high accuracy in a record amount of time suitable for haptic feedback. The take-home-message of this work is that for a given network architecture, the prediction time is nearly constant (and very short), irregardless of the size of the problem. Furthermore the accuracy of the prediction, which depends on the quality and the size of the data set, is controllable since we generate this data. We believe that such an approach has a tremendous potential for problems requiring very fast simulations of objects undergoing interactions. Such interactions could be user-driven or the result of contacts with other structures.

Yet, there are several directions to investigate to make this approach more broadly usable, in particular in the context of biomechanics, where material parameters, boundary conditions and geometries can vary from one patient to another. To this end, we will be investigating transfer learning as a means to refine a neural network pre-trained on an average model.

6 Aknowledgments

The authors would like to thank Jean-Nicolas Brunet for the writing assistance and proof reading.

References

[1] Cotin, S., Delingette, H., Ayache, N. (1999). Real-time elastic deformations of soft tissues for surgery simulation. IEEE transactions on Visualization and Computer Graphics, 5(1), 62-73.

[2] Haouchine, N., Dequidt, J., Berger, M. O., Cotin, S. (2013). Deformation-based augmented reality for hepatic surgery. Studies in health technology and informatics, 184.

[3] Haferssas, R., Jolivet, P., Nataf, F. (2017). An Additive Schwarz Method Type Theory for Lions's Algorithm and a Symmetrized Optimized Restricted Additive Schwarz Method. SIAM Journal on Scientific Computing, 39(4), A1345-A1365.

[4] Niroomandi, S., Alfaro, I., Cueto, E., Chinesta, F. (2008). Real-time deformable models of non-linear tissues by model reduction techniques. Computer methods and programs in biomedicine, 91(3), 223-231.

[5] Ryckelynck, D. (2005). A priori hyperreduction method: an adaptive approach. Journal of computational physics, 202(1), 346-366.

[6] Goury, O., Duriez, C. (2018). Fast, Generic, and Reliable Control and Simulation of Soft Robots Using Model Order Reduction. IEEE Transactions on Robotics, (99), 1-12.

[7] Bhattacharjee, S., Matou, K. (2016). A nonlinear manifold-based reduced order model for multiscale analysis of heterogeneous hyperelastic materials. Journal of Computational Physics, 313, 635-653.

[8] Niroomandi, S., Alfaro, I., Cueto, E., Chinesta, F. (2010). Model order reduction for hyperelastic materials. International Journal for Numerical Methods in Engineering, 81(9), 1180-1206.

[9] Niroomandi, S., Gonzalez, D., Alfaro, I., Bordeu, F., Leygue, A., Cueto, E., Chinesta, F. (2013). Realtime simulation of biological soft tissues: a PGD approach. International journal for numerical methods in biomedical engineering, 29(5), 586-600.

[10] Johnsen, S. F., Taylor, Z. A., Clarkson, M. J., Hipwell, J., Modat, M., Eiben, B., Han, L. , Hu, Y. , Mertzanidou, T. , Hawkes, D. J. Ourselin, S. (2015). NiftySim: A GPU-based nonlinear finite element package for simulation of soft tissue biomechanics. International journal of computer assisted radiology and surgery, 10(7), 1077-1095.

[11] Allard, J., Cotin, S., Faure, F., Bensoussan, P. J., Poyer, F., Duriez, C., Delingette, H., Grisoni, L. (2007). SOFA - an open source framework for medical simulation. In MMVR 15-Medicine Meets Virtual Reality (Vol. 125, pp. 13-18).

[12] Comas, O., Taylor, Z. A., Allard, J., Ourselin, S., Cotin, S., Passenger, J. (2008). Efficient nonlinear FEM for soft tissue modelling and its GPU implementation within the open source framework SOFA. In International Symposium on Biomedical Simulation (pp. 28-39). Springer, Berlin, Heidelberg.

[13] Miller, K., Joldes, G., Lance, D., Wittek, A. (2007). Total Lagrangian explicit dynamics finite element algorithm for computing soft tissue deformation. Communications in numerical methods in engineering, 23(2), 121-134.

[14] Lorente, D., Martínez-Martínez, F., Rupérez, M. J., Lago, M. A., Martínez-Sober, M., Escandell-Montero, P., Martínez-Martínez, J.M. , Martinez Sanchis, S., Serrano-López, A. , Monserrat, C, Martín-Guerrero, J. D. (2017). A framework for modelling the biomechanical behaviour of the human liver during breathing in real time using machine learning. Expert Systems with Applications, 71, 342-357.

[15] Luo, R., Shao, T., Wang, H., Xu, W., Zhou, K., Yang, Y. (2018). DeepWarp: DNN-based Nonlinear Deformation. arXiv preprint arXiv:1803.09109.

[16] Roewer-Despres, F., Khan, N., Stavness, I. Towards finite element simulation using deep learning, 15th International Symposium on Computer Methods in Biomechanics and Biomedical Engineering (2018).

[17] Amundarain, A., Borro, D., Garca-Alonso, A., Gil, J. J., Matey, L., Savall, J. (2004). Virtual reality for aircraft engines maintainability. Mechanics and Industry, 5(2), 121-127.

[18] Barbi, J., James, D. L. (2008). Six-dof haptic rendering of contact between geometrically complex reduced deformable models. IEEE Transactions on Haptics, 1(1), 39-52.

[19] Ronneberger, O., Fischer, P., Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical image computing and computer-assisted intervention (pp. 234-241).

[20] Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain.

[21] Kingma, D. P., Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

[22] Kourounis, D., Fuchs, A., Schenk, O. (2018). Toward the next generation of multiperiod optimal power flow solvers. IEEE Transactions on Power Systems, 33(4), 4005-4014.