Real-time biomechanical modeling of the liver using Machine Learning models trained on Finite Element Method simulations

Oscar J. Pellicer-Valero[a], María José Rupérez[b], Sandra Martínez-Sanchis[b], José D. Martín-Guerrero[a]

[a]*Intelligent Data Analysis Laboratory, Department of Electronic Engineering, Universitat de València (UV), Valencia, Spain*
E-Mail: `Oscar.Pellicer@uv.es` `jose.d.martin@uv.es`
[b]*Centro de Investigación en Ingeniería Mecánica (CIIM), Universitat Politècnica de València (UPV), Valencia, Spain*
E-Mail: `mjrupere@upvnet.upv.es`

Abstract

The development of accurate real-time models of the biomechanical behavior of different organs and tissues still poses a challenge in the field of biomechanical engineering. In the case of the liver, specifically, this would constitute a great leap forward for complex applications such as surgical simulators, computed-assisted surgery or guided tumor irradiation.

In this work, a relatively novel approach for developing such a model is presented. It consists in the use of a machine learning algorithm, which provides real-time inference, trained on tens of thousands simulations of the biomechanical behavior of the liver carried out by the finite element method on more than 100 different liver geometries.

Four different scenarios were modeled: a single liver with an arbitrary force applied, a single liver with two simultaneous forces applied, a single liver with different material properties with an arbitrary force applied, and a much more general model capable of modeling the behavior of any liver with an arbitrary force applied. For the definition of sufficient accuracy, a $3mm$ threshold was set for the Mean Euclidean Error (MEE). The results show that the Machine Learning (ML) models perform extremely well on all the scenarios, managing to keep the MEE under $1mm$ in all cases.

These results constitute a remarkable improvement in this field and may involve a prompt implementation in clinical practice.

Keywords: Machine Learning, Neural Networks, Deep Learning, Principal Components Analysis, Finite Element Method, Real Time, Liver, Coherent Point Drift, Biomechanical Modeling, Soft Tissue

1. Introduction

In the last few years, the field of biomechanical engineering has undergone a continued growth as many new technology-driven applications are being developed and introduced in the clinical practice. However, several specific applications, such as Computed Assisted Surgery (CAS), surgical simulators with haptic feedback or directed tumor irradiation (as in gating), all share a requirement for precise and real time biomechanical models of the organ they must interact with, a subject that remains as one of the biggest challenges in biomechanics.

In the case of CAS, the liver is of special interest, since it moves significantly during the respiratory cycle. High precision techniques such as biopsies, tumor ablation, cryotherapy, brachitherapy, tumor embolization, directed irradiation (gating), or vector delivery for genetic therapy (1) could all benefit from a biomechanical model of the liver able to assist clinicians during these procedures.

The first biomechanical models able to work in real time were based on mass-spring simulations (2; 3). Despite their speed, these have been progressively abandoned due to their inability to accurately model the nonlinearities which characterize biological tissue.

Models based on the Finite Elements Method (FEM), on the contrary, have a very well established mechanical and mathematical base, and allow for high accuracy simulations for any kind of geometry or material. However, the increased accuracy comes at the cost of prohibitive computational times, which hinder the application of this technique for real-time systems.

In order to accelerate these simulations, there are two main techniques that stand out in the literature, the first trying to exploit the parallelism of the problem with Graphic Processing Units (GPUs) or Central Processing Unit (CPU) clusters (4; 5), and the second attempting dimensionality reduction techniques by means of algorithms like Proper Generalized Decomposition (PGD); this algorithm tries to only solve the few most important deformation modes, allowing for very accurate real-time results in many problems (6).

In distributed or GPU models, the price to pay in exchange for real-time simulations is the use of very coarse meshes of less than 500 nodes, which lack the required accuracy for most applications (4). In the PGD-based models, the main limitation is the requirement of (approximately) linearly separable deformation modes, which may limit its application to the hyperelastic materials which normally describe the mechanical behavior of the soft biological tissue (7).

Other radically different approaches are the data-based strategies, which consist in training a Machine Learning (ML) model from simulations (as those obtained from FEM), or directly from sensor data. ML algorithms are able to automatically learn nonlinear mappings between several inputs (applied force, application area, node coordinates, etc.), and several outputs (e.g.: displacement, strain or stress fields). Although the training process is relatively slow, once trained, these algorithms provide extremely quick inference times, therefore fulfilling the requirement for real-time simulations. This strategy has been successfully applied in the literature on several organs (8; 9), being the work presented in (10) the first where this approach was applied in the case of the liver. In all instances, however, the employed meshes were of low resolution, and also, only one liver geometry was considered. Thus, in order to apply this procedure to any other liver, all the FEM simulations should be repeated on any new geometry, and a new ML model should be trained on the results, both processes being very time consuming.

As an exception, the work presented in (11) proposed a ML model that was validated on geometries from different livers. Here, though, the limitation stemmed from the fact that only a very reduced displacement set was considered.

In sight of the current state of the art, the main objective of this work is to develop a data-based model able to simulate the biomechanical behavior of any liver, subjected to any force, with sufficient accuracy and in real time.

On one hand, for the definition of sufficient accuracy, a $3mm$ threshold was set for the Euclidean error (11). On the other hand, real-time implies that the simulation is able to run at least at $25Hz$, while for haptic feedback applications, real time is considered only for frequencies higher than $300Hz$ (12).

In addition to the main ML model (applicable to any liver), three more ML models will be developed in order to show that this procedure could also be used for very high precision scenarios, multiple force interactions, or even scenarios where the material properties of the liver are variable.

A remarkable contribution included in this work is the development of an algorithm able to provide a natural parametrization of the geometry of any liver in only a few variables. Moreover, a simple yet powerful modification to the Coherent Point Drift (CPD) algorithm, which significantly improves its performance when the registered geometries differ substantially, is employed.

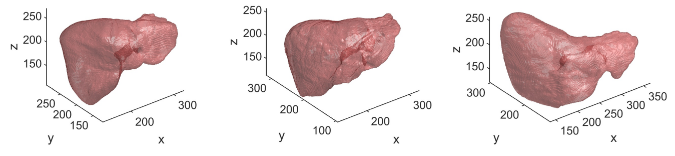
The work layout is as follows. First, Section 2 introduces the relevant details regarding the research development, focusing on data sources, data processing, FEM simulations and ML models training. Then, in Section 3, the results are presented and discussed. Finally, Section 4 summarizes the main conclusions drawn form the work, and suggests further lines for research.

2. Experimental setup

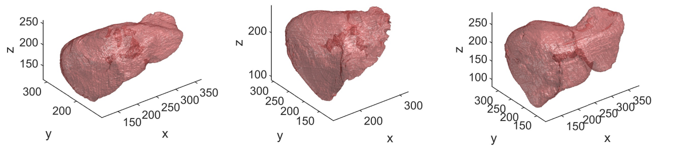
2.1. Data acquisition

The Computed Tomography (CT) images used in this project came from two main sources. The first image set (OWN) was provided by Hospital La Fe, in Valencia, and consists of a total of 24 abdominal CT images with their respective segmentation mask (a binary 3D image) for the liver (Figure 1a).

The second set ($LITS$) comes from the 2017 Liver Tumor Segmentation Challenge (LiTS)(13). This was a competition organized by MICCAI 2017 in conjunction with ISBI 2017 whose objectives were the automatic liver segmentation, tumor segmentation, and tumor load estimation. This dataset is publicly available and consists of 130 scans of abdominal CT images from six different medical centers with their respective segmentation of the liver and its tumors (Figure 1b).



(a) OWN dataset



(b) $LITS$ dataset

Figure 1: Some liver masks of the OWN (a) and the $LITS$ (b) datasets.

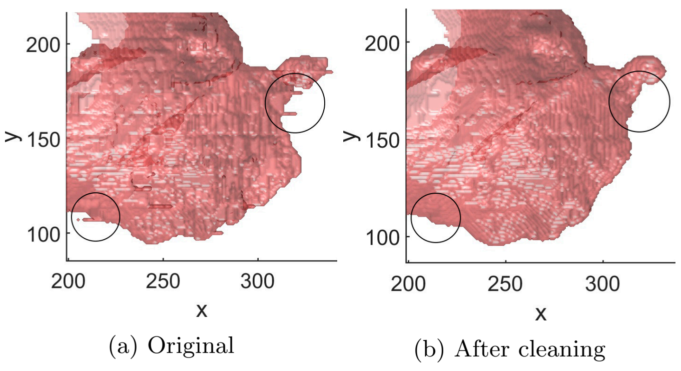
2.2. Data processing

Once all the images were acquired, they were resized, put in a same frame of reference, cleaned, and meshed. Matlab 2018a was used for all the image processing steps, while Simpleware's ScanIP was used for meshing. All these steps were completely automated and can, therefore, be easily applied to new images should they become available.

Firstly, the masks were resized (using cubic interpolation) so that the voxels had a size of $1mm$ in all three dimensions. Secondly, some masks were flipped along some axis or some axes were swapped, so that they all shared the same axis configuration. Thirdly, all masks were moved to a $400 \times 400 \times 400$ image and their centroid set to the position $(200, 200, 200)$. All this processing was necessary due to the multiple sources that the images came from. Additionally, images with a voxel size coarser than $2.5mm$ along any dimension were discarded.

At this point, it was decided to discard some outlying liver geometries (15 out of 152) which would probably confound the model. The discarded livers were chosen by visual inspection before any further processing was performed.

The next step was to clean the images to avoid meshing issues or later convergence problems. On one hand, some segmentations suffered from artifacts (Figure 2 left) and noise which was not a result of the actual geometry of the liver, but rather a result of the employed automatic segmentation method. To solve this issue, an *opening* followed by a *closing* morphological operation was applied, using a spherical structuring element with a radius of three voxels.



(a) Original (b) After cleaning

Figure 2: Cleaning of the binary mask based on an *opening-closing* morphological operation. Circles highlight areas where some of the artifacts appear.

On the other hand, only a few masks had a segmented hepatic tree. Therefore, in order to homogenize all images, it was decided to fill in the cavities left by the segmented ducts, by means of further morphological operations (Figure 3).

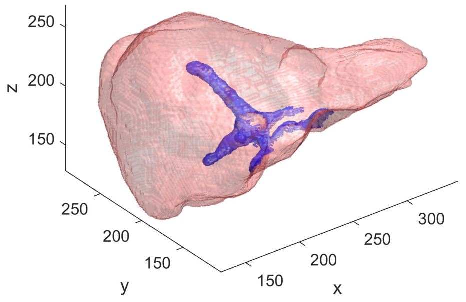


Figure 3: The hepatic tree (in blue) is automatically detected and filled in.

The final step was meshing all livers. This could be automated by using ScanIP scripting capabilities. The resulting meshes had $11,736 \pm 3,599$ nodes (Figure 4).

2.3. FEM simulations

FEM is a numerical method for finding approximate solutions for a particular field ϕ (such as the deformation field) on an arbitrarily shaped geometry (such as that of any liver) given a particular set of boundary conditions (restrictions on how the liver interacts with its surroundings). This is achieved by discretizing this geometry in a set of finite elements and finding the solutions of the field only for the nodes that comprise it, thus reducing the degrees of freedom of the problem (14).

The usual formulation for the elastic problem is based on variational methods. If an energy balance is applied on the body of interest, Equation (1) is obtained:

$$\Pi_p = \Pi_s - W_p \tag{1}$$

where Π_p is the total potential energy of the system, Π_s stands for the energy stored in the deformed structure and W_p represents the work exerted by the forces acting upon it.

In virtue of the Minimum Total Potential Energy Theorem, the total potential energy Π_p will be minimum at the equilibrium, namely, for a particular displacement field $\{u\}$ (which will be solution) for which all the differential equations and boundary conditions are simultaneously satisfied. Therefore, if the solution is found when Π_p reaches a minimum, then it must also be true that its derivative with respect to the system parameters (or degrees of freedom) $\{u\}$ must also be null, as Equation (2) shows:

$$\frac{\delta \Pi_p(\{u\})}{\delta(\{u\})} = 0 \tag{2}$$

Since the field $\{u\}$ has been discretized, the solution must only be found for a finite number of points u_i.

$$\frac{\delta \Pi_p(u_i)}{\delta(u_i)} = 0 \tag{3}$$

Developing Equation (3), a solution for u_i can be found.

When a solid body is subjected to a large deformation under a comparatively small load, the relationship of positions in deformed and undeformed configurations is described by a deformation gradient tensor $\mathbf{F}$:

$$\mathbf{F} = \sum_{\alpha=1}^{3} \lambda_\alpha \mathbf{n}_\alpha \otimes \mathbf{N}_\alpha := \begin{bmatrix} \mathbf{F}_{11} & \mathbf{F}_{12} & \mathbf{F}_{13} \\ \mathbf{F}_{21} & \mathbf{F}_{22} & \mathbf{F}_{23} \\ \mathbf{F}_{31} & \mathbf{F}_{32} & \mathbf{F}_{33} \end{bmatrix} \tag{4}$$

Where λ_1, λ_2 and λ_3 are the stretches in the three principal directions; and $\lambda = 1 + dL/L$ with L being the undeformed length. $\mathbf{N}_1, \mathbf{N}_2, \mathbf{N}_3$ and $\mathbf{n}_1, \mathbf{n}_2, \mathbf{n}_3$ are material vector triads and spatial vector triads, respectively. The left Cauchy Green deformation tensor, $\mathbf{B}$, describes the strain, while the Cauchy stress tensor T, describes the stress:

$$\mathbf{B} = \mathbf{F}\mathbf{F}^T = \sum_{\alpha=1}^{3} \lambda_\alpha^2 \mathbf{n}_\alpha \otimes \mathbf{n}_\alpha \tag{5}$$

$$\mathbf{T} = -p\mathbf{1} + 2\frac{\delta W}{\delta I_1}\mathbf{B} - 2\frac{\delta W}{\delta I_2}\mathbf{B}^{-1} \tag{6}$$

Where:

$$I_1 = tr\mathbf{B} \tag{7}$$

$$I_2 = \frac{1}{2}[(tr\mathbf{B})^2 - tr(\mathbf{BB})] \tag{8}$$

$$I_3 = det\mathbf{B} \tag{9}$$

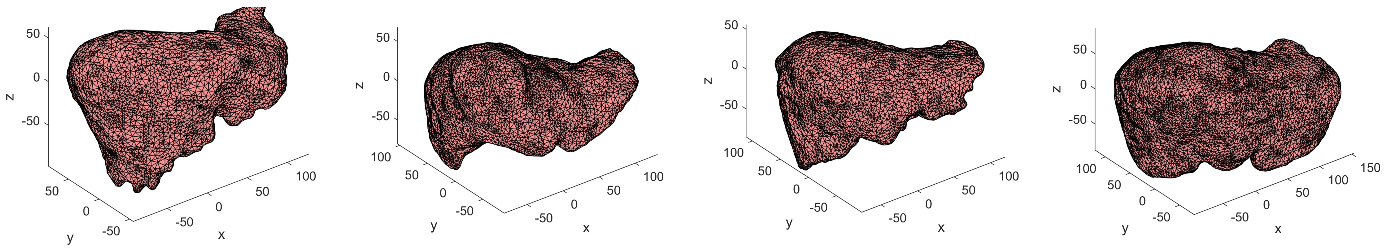


Figure 4: Examples of FE liver meshes.

Where I_1, I_2 and I_3 are strain invariants. The hydrostatic pressure p is constitutively indeterminate, and hence it is obtained form the underlying equilibrium and boundary conditions of the particular problem. Based on the isotropic assumption, the strain energy density function W is expressed as function of the strain invariants (15; 16):

$$W = W(I_1, I_2, I_3) \tag{10}$$

Alternatively, on the basis of the Valanis and Landel hypothesis (17), W can also be expressed directly as a function of the three principal stretches, namely λ_1, λ_2 and λ_3. A rigorous mathematical interpretation is available (18) to show that both properties are equivalent.

2.3.1. Biomechanical model

The next step was to choose a suitable constitutive model for the liver. The main tissue found in this organ is the parenchyma, which in its most general form can be considered a visco-poro-hiperelastic material. Moreover, the hepatic tree is comprised of a different material, which should be independently characterized. Finally, some authors also take into account the presence of a collagen capsule wrapping the parenchyma, known as Glisson capsule (19).

Concerning the characterization of the parenchyma's hyperelastic behavior, most recent studies have found a first order Ogden model (20) specially well suited, requiring only two empirical parameters (21) as elastic constants for its construction. Regarding viscoelasticity, the bibliography on the topic shows that traction tests at different deformation velocities give rise to different Young moduli (20), thus proving it usefulness. Regarding the porous properties of the parenchyma (which should be able to model the capilarization of the liver), its inclusion in a FEM model still remains marginal (21). Moreover, some authors model the hepatic tree as a set of truss elements (4), or as a different mesh with its own mechanical properties (22). Finally, with respect to the Glisson capsule, few authors (23) consider its inclusion in FEM models due to the difficulty of segmenting it in medical images and/or estimating its material properties.

Following the latest trends, a first order Ogden model was used to model the mechanical behavior of the liver parenchyma. Viscoelasticity was not taken into account based on the hypothesis that the applied forces were slow enough for such effects not to be of importance. Thus, the performed simulations were static. Finally, the model was considered homogeneous and the porosity effects were embedded into the elastic properties of the Ogden model. Regarding the hepatic tree or the Glisson capsule, both were assimilated by the parenchyma due to the lack of segmentation masks for these elements.

The deformation energy density function W for an Ogden elastic model is given by Equation (11):

$$W = \sum_{k=1}^{N} \frac{\mu_k}{a_k} (\lambda_1^{a_k} + \lambda_2^{a_k} + \lambda_3^{a_k}) \tag{11}$$

where N is the order of the model, μ_k and a_k are empirical parameters of the material and $\lambda_1, \lambda_2, \lambda_3$ are the principal stretches.

The values for the material parameters μ_k and a_k were obtained from a set of 30 material properties described in (20). A default material was chosen as the median of all material properties in said paper:

$$a_1 = 10.06, \mu_1 = 4.1kPa$$

For the compressibility modulus (K_0), a value of 100 times the value of μ_1 was selected to ensure a quasi-incompressible behavior, which is a common choice for soft tissue modeling.

2.3.2. Boundary conditions

From the perspective of the boundary conditions (BC), the liver is in contact with multiple organs and structures, thus rendering the task of finding anatomically correct BCs extremely challenging.

The most usual solution found in the literature is to resort to simplified BCs, where some nodes are considered fixed and the rest are left free. Following this trend, many authors fix the nodes in contact with the cava vein or with the falciform ligament (21; 24). Others do not consider the falciform ligament as a restriction and resort to only fixing the cava vein (22).

Typically, even more simplified BCs are considered, especially when the objective is to prove the feasibility of a new method rather than to make an anatomically correct simulation of the organ and its interactions within the body. E.g.: in (25) and (23) palpation was simulated with the livers laid against a flat surface, hence sufficing to fix the nodes in contact with this surface.

For this work, the liver was considered attached only to the cava vein. Therefore, a null displacement boundary condition was applied to the liver nodes in contact with it (Figure 5).

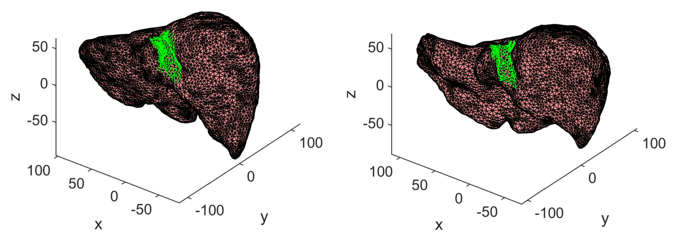


Figure 5: Some livers with BCs applied (in green).

2.3.3. *Applied forces*

The last step before launching the simulations consisted in defining the forces to be applied to the liver. In the literature, cylindrical indenters are typically pressed on to the surface of the organ, which also allows for a direct comparison of the simulation results with previous ex vivo identation tests (23). Other authors consider indenters with an infinitely small radius, hence the forces becoming nodal forces (25).

The forces in this work were also considered to be nodal, and they were applied on a random node of the liver, with a random orientation, and with a magnitude of $0.4N$ (a similar magnitude to the forces applied in (23) or (25)). The objective was to provide a simple but challenging set of forces. Nonetheless, for a real-world application, the simulations could be performed with any kind of forces originating from any sort of surgical tool deemed necessary.

2.3.4. *Performed simulations*

Following the proposed objectives, the mechanical behavior of the liver was simulated in four different scenarios.

Table 1: Simulated scenarios

ID	*Random forces*	*Two forces*	*Multiple mater.*	*Multiple livers*	*Simulated forces*
1	Yes	No	No	No	400
2	Yes	Yes	No	No	1,500 (×2)
3	Yes	No	Yes	No	3,000
4	Yes	No	No	Yes	10,200

In Table 1, a summary of the setup for each simulated scenario is presented. In the first scenario (the simplest one), only one liver geometry was considered, upon which 400 different forces, each one with a random position and orientation, were applied, hence adding up to a total of 400 deformed livers, stemming from 400 different FEM simulations.

The second and third scenarios also employed only one liver. However, in the second scenario two random simultaneous forces were considered in each simulation (instead of just one), while in the third scenario multiple material properties were contemplated. In contrast to the first scenario, more simulations were needed to account for the growing casuistry.

Finally, the fourth scenario posed the biggest challenge, as it was designed to work on any liver geometry.

All the simulations were performed with FEBio (26) and automated with Matlab. The analysis type was static (since no viscoelasticity was finally considered), and the force was applied in ten steps to help the FEM converge when large deformations are present. Consequently, for each simulation ten results were obtained, with every variable being identical except for the magnitude of the force, thus allowing for a tenfold growth in the number of simulations at no additional cost.

2.4. *Training of the ML models*

2.4.1. *Feature selection for the ML models*

Before being able to use the simulation data to train a ML model, it should be expressed in terms of an input matrix X and an output matrix Y. Then, a ML algorithm would be trained to learn the mapping $X \rightarrow Y$ as accurately as possible.

The input matrix X was of dimensions: $(S * N * T) \times F$ where S stands for the number of simulations (e.g., 400 for the first scenario), N stands for the number of nodes ($\sim 12,000$), T stands for the number of successfully simulated time steps (usually ten), and F is the number of features. Thereby, every row (each sample) corresponded to a node of the mesh, and each column contained several features, which are described as follows:

- x, y, z: Coordinates of the considered node
- f^x, f^y, f^z: Force vector
- n^x, n^y, n^z: Coordinates of the considered node to the node where force was applied.
- $d^x_{min}, d^y_{min}, d^z_{min}, d^{euc.}_{min}$: Distance from the considered node to the closest fixed node in x, y, z and total Euclidean distance.
- $d^x_{mean}, d^y_{mean}, d^z_{mean}, d^{euc.}_{mean}$: Distance from the considered node to the centroid of all the fixed nodes in x, y, z and total Euclidean distance.
- $d^x_{load}, d^y_{load}, d^z_{load}, d^{euc.}_{load}$: Distance from the considered node to the node where the force was applied.
- $a_1, a_2, ..., a_{27}$: Additional features that parametrize the liver geometry (to be explained in Section 2.4.2).

In addition to the previous features, for the second scenario (where two simultaneous forces were applied) further features related to the second force, with the exact same meaning as for the first force, were included: $(d^x_{min})_2, (d^y_{min})_2, (d^z_{min})_2, (d^{euc.}_{min})_2, (d^x_{mean})_2, (d^y_{mean})_2, (d^z_{mean})_2, (d^{euc.}_{mean})_2, (d^x_{load})_2, (d^y_{load})_2, (d^z_{load})_2, (d^{euc.}_{load})_2$.

Moreover, this allowed for the number of simulations to be artificially doubled just by swapping the values of the features associated to the first force with those associated to the second one.

Finally, the output matrix Y contained as many rows as X, as well as the following three columns:

- d^x, d^y, d^z: Node displacement in x, y, z.

2.4.2. Liver geometry parametrization

For the ML algorithm to be able to generalize to different livers, the whole liver geometry should somehow be introduced into each sample of X. To address this challenge, two consecutive subproblems must be solved:

a) Express each liver geometry as a vector of N features, such that each feature has the same meaning for all livers.

b) Apply dimensionality reduction techniques that reduce the vector from N to n features such that $n \ll N$.

Regarding the first subproblem, a typical approach consists in directly using the binary mask flattened into a vector. However, this method assumes that two voxels in the same location from two different livers have the same meaning or, in other words, represent the same feature, which is often not true.

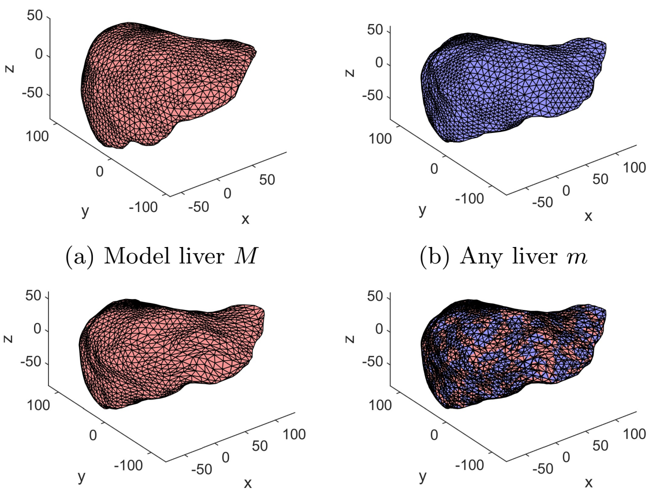
Here, a novel and effective approach is proposed. Firstly, the nodes of a model liver M (which was chosen beforehand for subjectively being the most regular), were registered by a soft registration algorithm to the nodes of each liver m to obtain M_r (M registered). Secondly, the displacement field that the nodes of M underwent to become M_r was obtained. Namely: $field = M_r - M$.

Then, M_r could be flattened to express it as vector, where each element has the same meaning for all livers: how much each node of M must be displaced in x, y or z in order to become M_r (where M_r is an approximation of the original geometry m).

The bottom right plot of Figure 6 shows that this registration process was successful, since M_r approximates m satisfactorily. A slightly coarser mesh was used to speed up the registration process.

For the non-rigid registration, a modified version of CPD was employed. CPD is a point set registration algorithm that finds the spatial transformation that best aligns two point sets and/or finds the correspondence between the points of both sets (27). CPD can perform both rigid registration (the transformation is limited to some combination of translation, rotation and scaling, amounting to a total of six parameters), and non rigid registration (a non linear transformation of any number of parameters).

The proposed modification consists in applying scheduled changes to the regularization parameters β (related to the width of a Gaussian smoothing filter) and λ (tradeoff between fit and regularization), which are gradually



(a) Model liver M (b) Any liver m

(c) M registered to m: M_r (d) m and M_r superimposed

Figure 6: An example of liver registration

reduced as the algorithm converges, thus refining the registration by going from low to high spatial frequencies. This change, although relatively trivial, allows the algorithm to significantly outperform its unmodified counterpart, specially for problems where the input and output shapes differ substantially.

The values used for these two regularization parameters can be found in Table 2, and a visualization of the convergence process in two different cases can be checked in in Figure 7.

Table 2: Regularization values β and λ for CPD depending on the relative tolerance of the algorithm; when the minimum relative tolerance is surpassed, the next stage is activated

Stage	*Minimum relative tolerance to switch to next stage*	β	λ
1	-	8	6
2	5×10^{-3}	2	1.5
3	1×10^{-3}	1.2	0.9
4	1×10^{-4}	0.6	0.45
5	1×10^{-5}	0.3	0.225

Concerning the second subproblem, PCA was used to reduce the dimensionality of this set of geometry characterizing vectors to only a few parameters. PCA makes a transformation of a possibly correlated data set into a non correlated orthogonal base, whose variables are called principal components (PCs), and are ordered by amount of variance explained. If the N variables in the original space are highly redundant, then it is possible to compress high dimensional data to only a few variables n (such that $n \ll N$) by applying PCA and taking the first n PCs as the new variables.

Figure 8 shows the reconstruction of the original liver geometry given the first 27 PCs.

Figure 9 shows (for all the livers aggregated) the Intersection over Union (IoU) and the DICE scores (which are

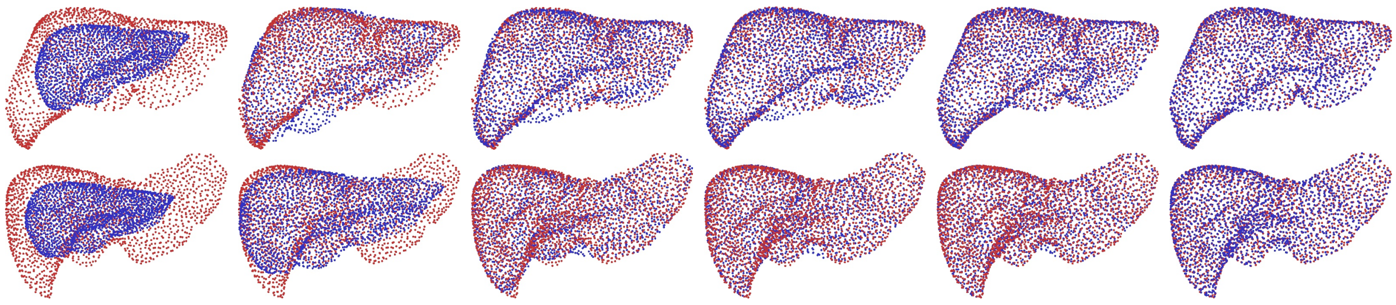


Figure 7: Convergence of the modified CPD algorithm for two different livers. First image to the left represents the initial state, while the rest were captured at the end of each stage, right before switching to the next. As it can be seen, the registration is extremely successful even when the geometries differ significantly. The blue point cloud corresponds to the nodes of the model liver M, as they mutate to become M_r, an approximation of the arbitrary-shaped liver m, whose nodes are given by the red point cloud.

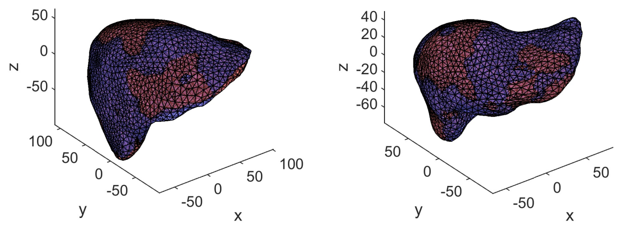


Figure 8: Reconstruction (in red) of the original geometry (in blue) given only the first 27 PCs for two different livers

both intersection scores), as well as the accumulated relative variance given the first n PCs (the rest were set to zero).

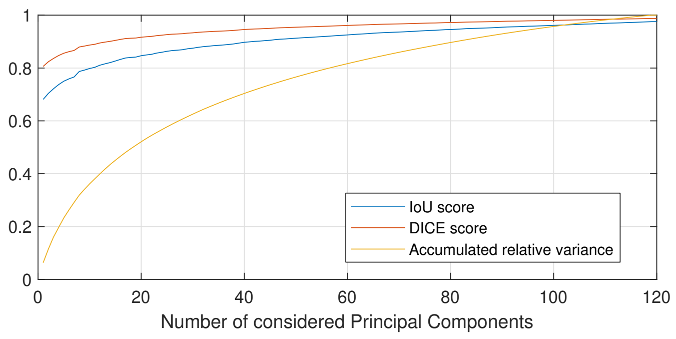


Figure 9: IoU score, DICE score and accumulated relative variance depending on the number of PCs used to reconstruct the geometry

Therefore, these 27 PCs will be the 27 features $(a_1, a_2, ..., a_{27})$, that will appear in matrix X to help parametrize the liver geometry. If the ML algorithm is powerful enough, it should be able to learn the meaning of each parameter and use it to internally reconstruct an approximation of the shape of each liver. The precise number of PCs to include in X was set to 27 by means of hyper-parameter optimization (view Section 2.4.3).

Finally, the proposed approach was compared to the few similar approaches found in the literature. Some works on the problem of liver geometry parametrization can be found in (28) or in (29). In the first paper, the authors use Locally Linear Embedding (LLE) on meshes generated by applying a rigid transformation (six parameters) to a single initial mesh. LLE successfully compresses this transformation into four parameters, while keeping most of the variance of the geometry. In the second paper, a similar experiment is conducted, but this time using Kernel PCA (kPCA) instead of LLE, and applying an affine transformation (twelve parameters) to the initial mesh, instead of a rigid transformation. Although the results were very promising in both papers, the first subproblem is not directly addressed in any of the them, thus rendering both approaches ineffective for arbitrary liver meshes.

The method here proposed can be thought as a generalization of the previous methods, in the sense that the proposed transformation is not limited to being rigid or affine; instead, it is a general nonlinear transformation. This approach can therefore be applied to any geometry, not only to simple transformations of a single model mesh.

2.4.3. ML model and hyper-parameter optimization

Once the input and output matrices (X and Y, respectively) were built, the next step was to train a ML algorithm able to approximate Y given X with sufficient accuracy, and in real time.

Regarding the kind of ML algorithm to use, two main contenders were tested: a Feedforward NN for regression and a Random Forest (RF) regressor (which is an ensemble of Regression Trees) (30). Even though the second algorithm was successfully employed in (11), for this particular problem the NNs showed a vastly superior performance in the preliminary tests, and were therefore finally chosen to conduct all the experiments.

The feedforward NNs used in this work are supervised learning algorithms able to learn nonlinear mappings (models) between certain inputs (such as the coordinates of the node, the force orientation, the distance to the force, etc.) and certain outputs (such as the displacement in x, y, z of a node).

Figure 10 offers a visual representation of the algorithm (for a single hidden layer, and ignoring biases) and introduces some notation. To compute the output of the network $\hat{Y}$ given the input X, Equations (12) and (13) are used:

$$Z = X \times W_1 \qquad (12)$$

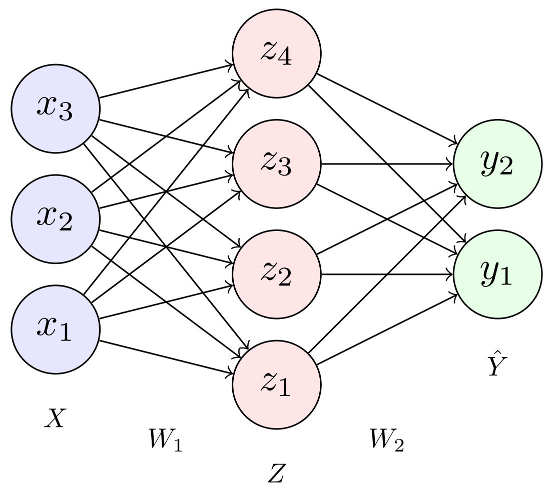


Figure 10: A feedforward neural network comprised of an input layer X, a hidden layer Z, an output layer $\hat{Y}$ and the weights (W_1, W_2)

$$\hat{Y} = f(Z) \times W_2 \tag{13}$$

where f is a nonlinear activation function, W_1 and W_2 are the weights of the network, or in other words, the parameters that the NN must learn, and X, Y contain the inputs and the actual outputs in matrix form, respectively. Thus, a NN can be seen as stack of linear transformations with nonlinear activation functions in between. The NN could have as many hidden layers as needed, but only one has been considered in Figure 10 for the sake of simplicity.

To train the NN to approximate a certain function, some parameters (or weights) $\theta = \{W_1, W_2\}$ must be found in order to minimize a certain cost function $J(\theta)$ such as the Mean Squared Error (MSE):

$$J(\theta) = grandmean((\hat{Y}(\theta) - Y)^2) \tag{14}$$

where *grandmean* represents the mean across all elements of the squared errors matrix $(\hat{Y}(\theta) - Y)^2$, and the exponentiation is applied element-wise.

To minimize J with respect to the parameters θ, Gradient Descent (GD) is typically used (31):

$$\theta = \theta - \mu \nabla_\theta J(\theta) \tag{15}$$

GD is an iterative algorithm that at each step pushes the parameters θ a small amount μ in the direction opposite to gradient ∇_θ of $J(\theta)$ with respect to θ, thus finally converging to a minimum for J (31).

NNs are extremely powerful nonlinear approximators that tend to overfit the training set, therefore jeopardizing the generality of the model to new samples that the network has not trained with. To prevent this problem, multiple techniques can be used, such as: adding $L2$ regularization to the parameters, adding gaussian noise to the inputs, or by using *dropout* (32). For all the tasks related to NN training, the Python library Keras (33) running on top of Tensorflow (34) was employed.

The objective at this point was to optimize the hyper-parameters of the NN model (such as the NN architecture, the learning rate, the batch size, etc., as well as other variables that affect performance such as the number of PCs to use) in order to reduce the error in the validation set.

For the first three scenarios, the validation set contained a randomly chosen $\sim$ 10% of all simulations, with a different applied force each. Thus, the performance of these models was evaluated on forces that the NN had not been trained with.

For the last scenario, where multiple liver geometries were considered, the validation set contained all the simulations corresponding to a randomly chosen $\sim$ 12.5% of all livers. Hereby, it is possible to assess the behavior of the model for livers that it has never seen before.

To obtain the final results (view Section 3), k-fold cross validation was employed, where $k = 10$ for the first three scenarios and $k = 8$ for the last scenario.

The hyper-parameter optimization was manually conducted, until the optimal values shown in Table 3 were reached. Rectified Linear Unit (ReLU) activation functions and MSE cost function were always employed, and noise injection (which consists in adding a random normal noise to the inputs as a way of regularizing the network) was applied on the last scenario only.

Regarding the training, the learning rate was reduced tenfold after the second epoch without improvement on validation loss for the first three scenarios, while for the last one, the learning rate was reduced by a factor of 1.5 after each epoch. For all scenarios, training was stopped if the loss in the validation set stopped improving.

Finally, it must be noted that all the data coming from the simulations was randomly sampled before being used to train the NN. The percentage of data remaining after the sampling is also shown in Table 3. This processing was performed to speed up the training process at practically no cost in performance. In fact, the simulation data is very redundant due to two main reasons. First, for nodes positioned away from the node of application of the force, the field is very similar between neighboring nodes. Second, each force was applied in ten steps (at an increasing magnitude), thus providing very similar results between consecutive steps.

3. Results and discussion

In this section, the results for each of the four models will be presented and it will be discussed if both sufficient accuracy and real-time inference were achieved. All final metrics have been computed using all the samples from all k validations sets concatenated.

3.1. Scenario 1: Base model

The first scenario is the simplest one, since only one liver, one material and one arbitrary force is considered (although the force may have any orientation, any magnitude, and be applied to any node). Only 360 simulations

Table 3: Optimal values found for the hyper-paramaters of the model in each scenario.

Scenario	*Architecture*	*Dropout*	*Batch size*	*Noise variance*	*L2 regulariz.*	*Learning rate*	*Number of PCs*	*% sampled*
1 (base case)	$(500, 300, 100, 50)$	$(0.3, 0.5, 0.6, 0.7)$	512	0	0	2.5×10^{-4}	0	50%
2 (two forces)	$(500,) \times 5$	$(0.5,) \times 5$	256	0	3×10^{-5}	1.25×10^{-4}	0	50%
3 (30 materials)	$(500,) \times 5$	$(0.5,) \times 5$	512	0	0	2.5×10^{-4}	0	25%
4 (102 livers)	$(500,) \times 5$	$(0.5,) \times 5$	4096	0.1	2×10^{-4}	1×10^{-3}	27	12%

were employed to train this model, while the remaining 40 made up the validation set.

The numerical results for all scenarios have been compiled in Table 4. A naive model (which always outputs a constant value calculated as the average of all the training outputs) was included for comparison as well.

Analyzing the first row of Table 4 (which corresponds to this scenario), the first column shows the mean absolute error (MAE) for each output coordinate (x, y, z), followed by the mean euclidean error (MEE), which represents the mean distance between the predicted and the real displacement fields. All these errors are extremely low, staying around $0.12mm$ for all three coordinates, and below $0.25mm$ for the MEE. For reference, both the discretization error of the original mask as well as the maximum ScanIP's mesh error are around $0.5mm$ ($\frac{1}{2}$ the voxel size). Also, the maximum displacements for this particular scenario are above $20mm$, as it can be seen in Figure 12, which will later be discussed.

Continuing with Table 4, the following two columns show the percentage of samples with an Euclidean error (EE) below $1mm$ and $3mm$ (which was set as the objective threshold). As it can be observed, the results are also very good in this regard, since 99.96% of the samples manage to stay below the $3mm$ error limit, while 98.26% stay below $1mm$. Finally, the last three columns show the correlation coefficients for all three output coordinates, which for this scenario are almost unitary, proving the notable performance of the proposed model.

Figure 11 shows the absolute error (AE) distributions in x, y and z, as well as the Euclidean error (EE) distribution. As it can be noted, coordinate errors manage to stay under $0.25mm$, while Euclidean errors remain below $0.6mm$, excluding outliers. These errors are well below the maximum systematic error of $1mm$.

Finally, Figure 12 shows how the predicted Euclidean displacements (in red) follow very closely the actual euclidean displacements (in blue), even when large displacements are present.

3.2. Scenario 2: Two simultaneous forces

For the second scenario, two simultaneous random forces were applied in each of the 1,500 simulations. This is a rather complex situation, since the non-linearities prevent the resulting deformation field from being a simple addition of the independent effects of each force.

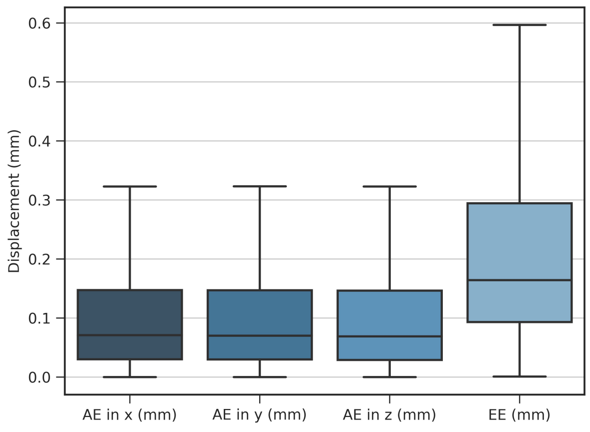


Figure 11: Scenario 1: Box plot of the absolute errors (AE) in x, y, z, as well as the euclidean error (EE). Outliers are excluded.

As it can be seen from the second row of Table 4, the results are still accurate, achieving a 99.84% of the samples within the error threshold of $3mm$. It is worth mentioning that the naive model performs much worse in this scenario as it did on the previous one, suggesting that the deformation magnitudes are significantly larger, which makes them more difficult to predict. The box plots in Figure 13 show that the euclidean error distribution mostly stays below $1mm$.

Finally, form the scatter plot (**??**), it can be checked that, in fact, the deformations are larger as compared to the previous case. Nevertheless, the model still achieves a very low dispersion in the predictions.

3.3. Scenario 3: 30 materials

The third scenario was designed to test the learning abilities of the model against the change in material properties. For each of the 3,000 simulations, one of 30 different parameter sets was randomly sampled, thus amounting to a total of ~ 100 simulations per material.

Once again, the third row of Table 4 shows that very good results were achieved. The percentage of samples below the $3mm$ mark falls slightly to 98.46% due to the presence of many more outliers. Indeed, a few very elastic materials give rise to extreme deformations when the force acts upon certain parts of the liver. Figures 14 and 15 confirm these conclusions; in fact, the box plot of Euclidean

Table 4: Results for all scenarios: Mean Absolute Error (MAE); Mean Euclidean Error (MEE); percentage of predictions with a Euclidean Error (EE) below $1mm$ and $3mm$; and correlation coefficient between predicted and actual values.

Scenario	*Algorithm*	*MAE (mm)*			*MEE (mm)*	*% of samples*		*Correlation coefficient*		
		x	y	z		*EE* $< 1mm$	*EE* $< 3mm$	x	y	z
1 (base case)	NN	0.1173	0.1176	0.1224	0.2389	98.2615%	**99.9551**%	0.9973	0.9974	0.9959
	Naive	1.5828	1.5777	1.4020	3.0661	32.5021%	64.2548%	-	-	-
2 (two forces)	NN	0.1862	0.1825	0.1977	0.3816	93.0551%	**99.8382**%	0.9967	0.9967	0.9949
	Naive	2.3247	2.2804	2.0247	4.4828	20.9003%	50.1540%	-	-	-
3 (30 materials)	NN	0.1621	0.1616	0.1548	0.3299	94.7381%	**98.4644**%	0.9911	0.9904	0.9900
	Naive	2.1622	2.1178	1.7933	4.2052	35.0012%	63.7634%	-	-	-
4: *Mean* (102 livers)	NN	0.4130	0.4732	0.4119	0.8643	72.4243%	**95.5784**%	0.9790	0.9814	0.9702
	Naive	1.4444	1.6767	1.2993	2.9534	35.3185%	66.6061%	-	-	-
4: *Median* (102 livers)	NN	0.3377	0.3360	0.3315	0.6619	79.0198%	**99.0115**%	0.9828	0.9874	0.9731
	Naive	1.6341	1.8307	1.2278	3.1332	33.7861%	63.3039%	-	-	-

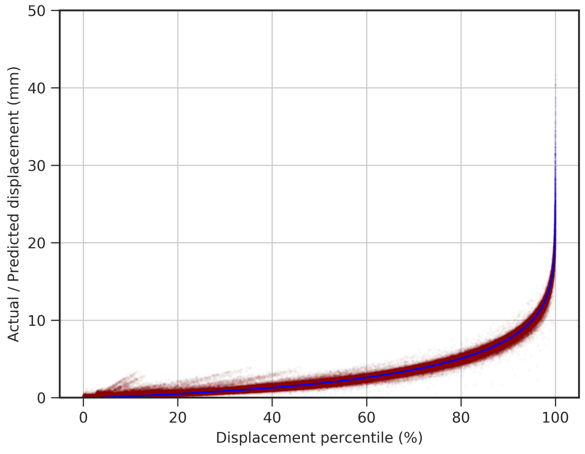


Figure 12: Scenario 1: Actual euclidean displacement (in blue) and predicted euclidean displacement (in red) for all samples, sorted by ascending actual euclidean displacement.

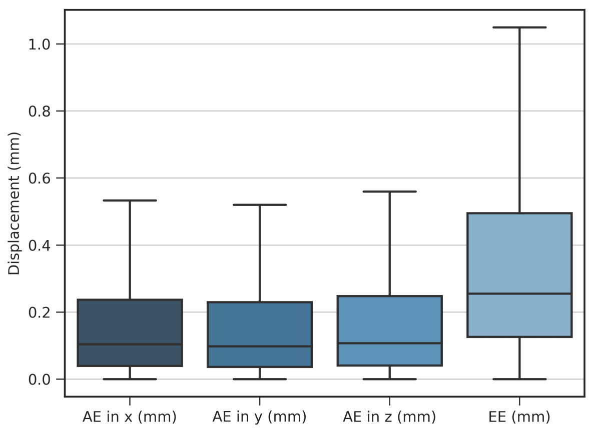


Figure 13: Scenario 2: Box plot of the absolute errors (AE) in x, y, z, as well as the euclidean error (EE). Outliers are excluded.

errors verifies that the performance is even better than in the first scenario (if outliers are excluded).

3.4. Scenario 4: 102 livers

The fourth scenario constitutes the main outcome of this paper. The objective was to train a model able to model the biomechanical behavior of any liver presenting any geometry. To this end, the validation sets were composed of all the simulations belonging to a $\sim 12.5\%$ of all livers. Hence, the results show how the model performs when tested on new livers.

For this scenario, the results were aggregated differently. On one hand, for each of the 102 livers in the validation set, the same metrics as in all the previous cases were computed. In fact, the last row of Table 4 shows these metrics for a median liver (chosen as the liver with a z correlation in the median). Figures 16 and 17 were obtained for this median liver. Even though the results have worsened, 99.01% of the samples still manage to stay below the allowed error threshold of $3mm$, which is an excellent finding, considering that the liver under consideration represents the median behavior of a model which was not trained with that particular geometry.

On the other hand, the results were aggregated for all livers by computing the the mean, which can be seen in the fourth row of Table 4. Here, the values are not as ideal due to the influence of some outlying liver geometries for which the model did not perform as well. To further prove this point, Figure 18 shows the distribution of MEE over all the livers of all validation sets. As it can be observed,

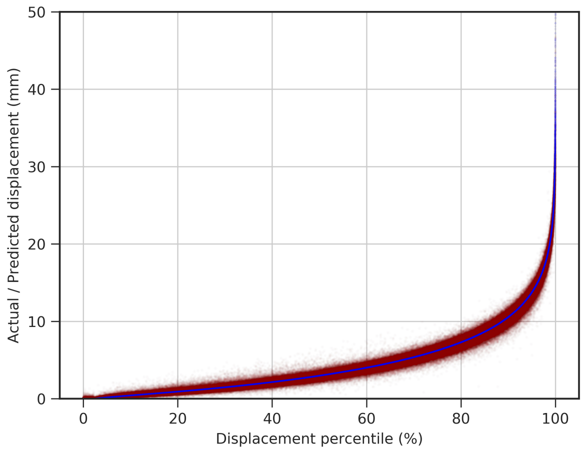


Figure 14: Scenario 2: Actual euclidean displacement (in blue) and predicted Euclidean displacement (in red) for all samples, sorted by ascending actual euclidean displacement.

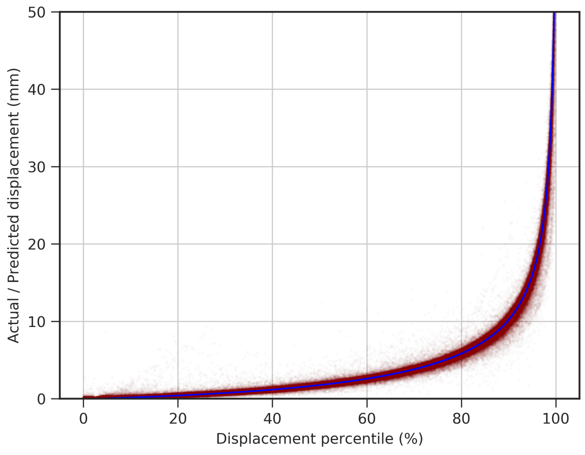


Figure 16: Scenario 3: Actual euclidean displacement (in blue) and predicted euclidean displacement (in red) for all samples, sorted by ascending actual euclidean displacement.

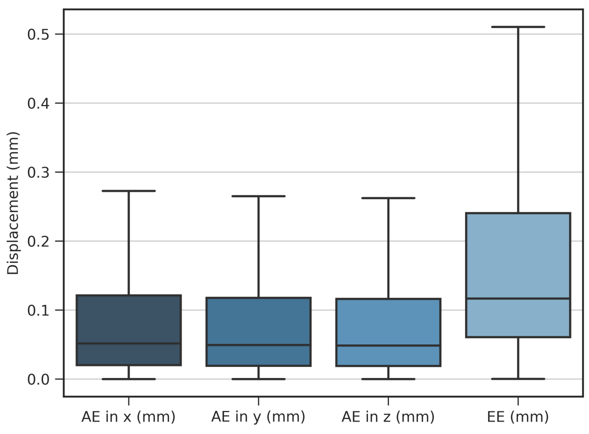


Figure 15: Scenario 3: Box plot of the absolute errors (AE) in x, y, z, as well as the euclidean error (EE). Outliers are excluded.

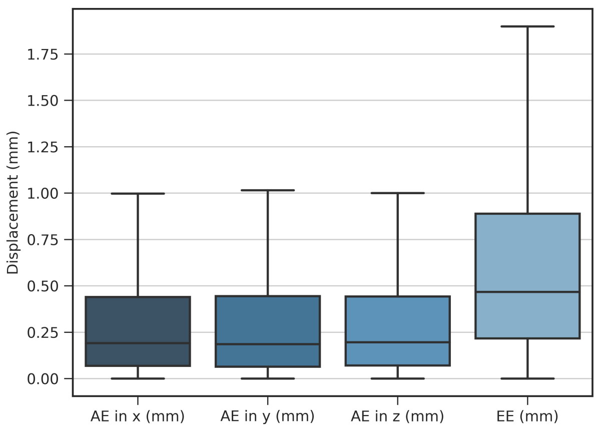


Figure 17: Scenario 4: Box plot of the absolute errors (AE) in x, y, z, as well as the euclidean error (EE). Outliers are excluded.

the distribution peaks at a MEE of around 0.6 to $0.7mm$ (precisely where the median liver lied), but some outlying livers with poorer performance drag the mean of the MEEs distribution towards higher values.

3.5. *Real time performance*

Next, the ability of the model to work in real time will be assessed. To make an inference (get the predicted output $\hat{Y}$) with a feedforward NN, two steps are required: build the matrix X, and propagate it through the network.

Building X is extremely quick, since most of its features can be computed offline, except for the distance to the node where the force was applied, as well as the force itself. Nonetheless, calculating these features can be considered immediate.

Propagating X through the NN consists of a series of matrix multiplications followed by non-linear activations. Matrix multiplication can be done very efficiently using GPUs, allowing for an improvement in speed of several orders of magnitude with respect to a CPU. The ReLU non-linearities used in this paper are also extremely simple, and have a negligible impact on the final cost.

Oddly enough, the highest computational burden would be the cost of transferring the X matrix from computer memory to the GPU memory, but for such small X matrices, this is is not a problem either.

Finally, for a real application, it would not be necessary to compute the displacement field for all nodes, but rather for those which are of interest, such as the visible surface of the liver.

In a 2013 laptop equipped with a two core i5 processor and a low end GT 840M GPU, the time required for building X and propagating it through the NN to get the displacement of a liver is around $2ms$ for the simpler first scenario, and around $5ms$ for the last scenario (for which the network architecture is more complex). Using slightly

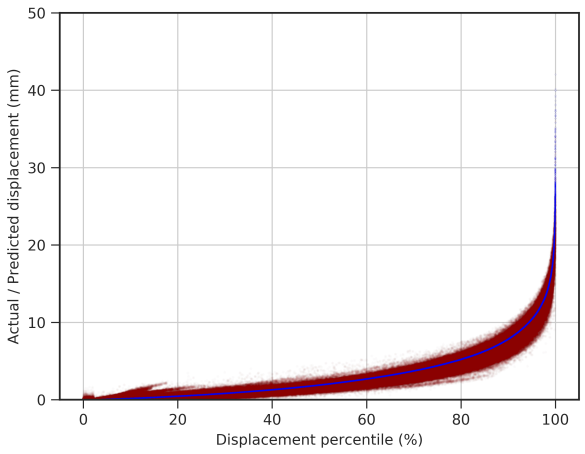


Figure 18: Scenario 4: Actual euclidean displacement (in blue) and predicted euclidean displacement (in red) for all samples, sorted by ascending actual euclidean displacement.

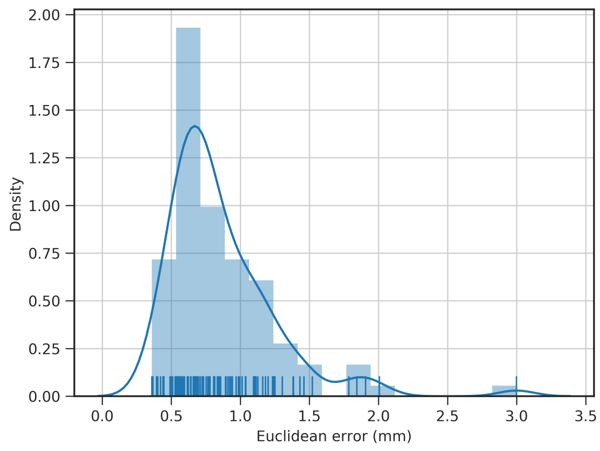


Figure 19: Scenario 4: Distribution of MEEs for all the 102 livers in the eight validation sets.

better hardware and a more polished implementation, it would be trivial to achieve inference times in the order of hundreds or even tenths of microseconds, thus enabling the model for its use in haptic feedback systems, which require working frequencies above $500Hz$.

3.6. Interactive liver manipulator

To conclude the results section, an interactive liver manipulator will be presented. It has been developed to visually assess the correct behavior of the model, as well as to prove its real time capabilities.

Figure 19 shows a video of this software being used on the last scenario, on the median liver. It must be emphasized that this is a liver that the model was not trained with. Watching the simulation, it is evident that the displacement field that the liver undergoes given arbitrary forces corresponds with the intuitive expectations, thus proving that the behavior is (at least visually) correct. Furthermore, it can be seen form the video that the time required for inference is usually around the previously stated $5ms$.

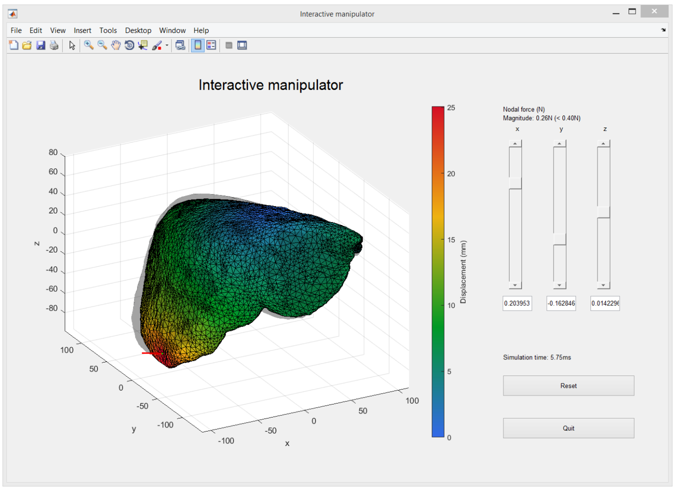


Figure 20: **(Online version: VIDEO FILE: `figures/results/video.mp4`)** Video showing the interactive manipulator simulating the mechanical behavior of the liver of median validation performance given an arbitrary force.
(Print version) Capture of the interactive manipulator simulating the mechanical behavior of the liver of median validation performance given an arbitrary force.

4. Conclusion

Summarizing, the main objective of building a general model able to simulate the mechanical behavior of any liver, given an arbitrary force with sufficient accuracy and in real time has been achieved. However, other options exist too. For instance, the model in the first scenario achieves higher levels of accuracy, as compared to any liver in the last scenario. In practice, for applications requiring such precision, it would be sensible to simulate a particular liver geometry under a few hundred forces and train a NN on top of it, as part of the pre-operative process. For instance, simulating the mechanical behavior of a liver under 200 different forces (which would be almost as accurate as with 360 forces), and training a NN model on these simulations, takes around two hours in a server with an eight-core Intel Xeon E5-2620v4 CPU and a mid tier Nvidia Maxwell GPU.

Furthermore, the capability of the method to generalize from situations where multiple forces or different material properties are at play has been shown.

Finally, a novel geometry parametrization algorithm was developed, which allows the NN model to generalize to unknown geometries. Moreover, a simple but effective modification to the CPD has been suggested, which enables this algorithm to achieve excellent registration results even when the registered geometries are drastically different.

5. Further work

Although the proposed methods proved successful, some areas of improvement can be detected.

First, the geometry parametrization technique, despite effectively allowing the NN to generalize to any liver geometry, is still approximate. A possible further research line would be to use convolutional layers to input the original segmentation mask directly into the NN.

Regarding convolutional NNs, a topic of active research is the development of automatic segmentation algorithms, which take the CT or Magnetic Resonance (MR) image and directly compute a segmentation mask.

Finally, finding accurate boundary conditions and constitutive models is still a challenge in biomechanics. Particularly, the improvement of techniques for in-vivo identification of elastic parameters is fundamental for the development of high-accuracy patient-specific models.

Acknowledgments

This work has been funded by the Spanish Ministry of Economy and Competitiveness (MINECO) through research projects TIN2014-52033-R, also supported by European FEDER funds.

References

[1] M. A. Clifford, F. Banovac, E. Levy, K. Cleary, Assessment of hepatic motion secondary to respiration for computer assisted interventions, Computer Aided Surgery 7 (5) (2002) 291–299. doi:10.1002/igs.10049.
URL http://doi.wiley.com/10.1002/igs.10049

[2] L. Nedel, D. Thalmann, Real time muscle deformations using mass-spring systems, in: Proceedings. Computer Graphics International (Cat. No.98EX149), Vol. 1998-Janua, IEEE Comput. Soc, 1998, pp. 156–165. doi:10.1109/CGI.1998.694263.
URL http://ieeexplore.ieee.org/document/694263/

[3] A. Duysak, J. J. Zhang, V. Ilankovan, Efficient modelling and simulation of soft tissue deformation using mass-spring systems, International Congress Series 1256 (C) (2003) 337–342. doi:10.1016/S0531-5131(03)00423-0.
URL https://www.sciencedirect.com/science/article/pii/S0531513103004230

[4] F. Faure, C. Duriez, H. Delingette, J. Allard, B. Gilles, S. Marchesseau, H. Talbot, H. Courtecuisse, G. Bousquet, I. Peterlik, S. Cotin, SOFA: A Multi-Model Framework for Interactive Physical Simulation, in: Soft Tissue Biomechanical Modeling for Computer Assisted Surgery Springer, 2012, pp. 283–321. doi:10.1007/8415_2012_125.
URL http://link.springer.com/10.1007/8415{_}2012{_}125

[5] I. Peterlík, C. Duriez, S. Cotin, Modeling and real-time simulation of a vascularized liver tissue., in: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2012, Vol. 15, 2012, pp. 50–7. doi:10.1007/978-3-642-33415-3_7.
URL http://www.ncbi.nlm.nih.gov/pubmed/23285534

[6] F. Chinesta, A. Leygue, F. Bordeu, J. V. Aguado, E. Cueto, D. Gonzalez, I. Alfaro, A. Ammar, A. Huerta, PGD-Based Computational Vademecum for Efficient Design, Optimization and Control, Archives of Computational Methods in Engineering 20 (1) (2013) 31–59. doi:10.1007/s11831-013-9080-x.
URL http://link.springer.com/10.1007/s11831-013-9080-x

[7] Y. C. Fung, R. Skalak, Biomechanics: Mechanical Properties of Living Tissues, Journal of Biomechanical Engineering 103 (4) (1981) 231. doi:10.1115/1.3138285.
URL http://biomechanical.asmedigitalcollection.asme.org/article.aspx?articleid=1395445

[8] A. Jahya, M. Herink, S. Misra, A framework for predicting three-dimensional prostate deformation in real time, The International Journal of Medical Robotics and Computer Assisted Surgery 9 (4) (2013) e52–e60. doi:10.1002/rcs.1493.
URL http://doi.wiley.com/10.1002/rcs.1493

[9] D. Deo, S. De, PhyNeSS: A Physics-driven Neural Networks-based Surgery Simulation system with force feedback, in: World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, IEEE, 2009, pp. 30–34. doi:10.1109/WHC.2009.4810896.
URL http://ieeexplore.ieee.org/document/4810896/

[10] K. Morooka, X. Chen, R. Kurazume, S. Uchida, K. Hara, Y. Iwashita, M. Hashizume, Real-time nonlinear FEM with neural network for simulating soft organ model deformation., Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention 11 (Pt 2) (2008) 742–9. doi:10.1007/978-3-540-85990-1-89.
URL http://www.ncbi.nlm.nih.gov/pubmed/18982671

[11] D. Lorente, F. Martínez-Martínez, M. Rupérez, M. Lago, M. Martínez-Sober, P. Escandell-Montero, J. Martínez-Martínez, S. Martínez-Sanchis, A. Serrano-López, C. Monserrat, J. Martín-Guerrero, A framework for modelling the biomechanical behaviour of the human liver during breathing in real time using machine learning, Expert Systems with Applications 71 (2017) 342–357. doi:10.1016/j.eswa.2016.11.037.
URL https://linkinghub.elsevier.com/retrieve/pii/S0957417416306728

[12] S. Cotin, H. Delingette, N. Ayache, A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation, The Visual Computer 16 (8) (2000) 437–452. doi:10.1007/PL00007215.
URL http://link.springer.com/10.1007/PL00007215

[13] P. Christ, LiTS - Liver Tumor Segmentation Challenge, LITS-Challenge.
URL https://competitions.codalab.org/competitions/17094

[14] G. Strang, G. J. Fix, An analysis of the finite element method, Prentice-hall Englewood Cliffs, NJ, 1973.

[15] R. S. Rivlin, Large Elastic Deformations of Isotropic Materials. IV. Further Developments of the General Theory, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 241 (835) (1948) 379–397. arXiv:/www.jstor.org/stable/91430, doi:10.1098/rsta.1948.0024.
URL http://rsta.royalsocietypublishing.org/cgi/doi/10.1098/rsta.1948.0024

[16] R. S. Rivlin, Large Elastic Deformations of Isotropic Materials. I. Fundamental Concepts, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 240 (822) (1948) 459–490. doi:10.1098/rsta.1948.0002.
URL http://rsta.royalsocietypublishing.org/cgi/doi/10.1098/rsta.1948.0002

[17] K. C. Valanis, R. F. Landel, The Strain-Energy Function of a Hyperelastic Material in Terms of the Extension Ratios, Journal of Applied Physics 38 (7) (1967) 2997–3002. doi:10.1063/1.1710039.
URL http://aip.scitation.org/doi/10.1063/1.1710039

[18] L. R. G. Treloar, The physics of rubber elasticity, Clarendon Press, 2005.

[19] A. Brunon, K. Bruyère-Garnier, M. Coret, Mechanical characterization of liver capsule through uniaxial quasi-static tensile tests until failure, Journal of Biomechanics 43 (11) (2010) 2221–2227. doi:10.1016/j.jbiomech.2010.03.038.
URL https://www.sciencedirect.com/science/article/pii/S0021929010001831?via{%}3Dihub

[20] C. D. Untaroiu, Y.-C. Lu, Material characterization of liver parenchyma using specimen-specific finite element models., Journal of the mechanical behavior of biomedical materials 26 (2013) 11–22. doi:10.1016/j.jmbbm.2013.05.013.
URL http://www.ncbi.nlm.nih.gov/pubmed/23800843

[21] S. Marchesseau, S. Chatelin, H. Delingette, S. Marchesseau, S. Chatelin, H. Delingette, B. Model, Y. Payan, J. Ohayon, Non linear Biomechanical Model of the Liver, HAL.
URL https://hal.inria.fr/hal-01536406

[22] R. Plantefève, I. Peterlik, N. Haouchine, S. Cotin, Patient-Specific Biomechanical Modeling for Guidance During Minimally-Invasive Hepatic Surgery., Annals of biomedical engineering 44 (1) (2016) 139–53. doi:10.1007/s10439-015-1419-z.
URL http://link.springer.com/10.1007/s10439-015-1419-z

[23] K. Lister, G. Zhan, P. Jaydev, Desai, Development of in vivo Constitutive Models for Liver: Application to Surgical Simulation, Ann Biomed Eng 39 (March 2011) (2011) 1060–1073.

[24] R. Plantefève, I. Peterlik, H. Courtecuisse, R. Trivisonne, J.-P. Radoux, S. Cotin, Atlas-Based Transfer of Boundary Conditions for Biomechanical Simulation, in: Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention, Vol. 17, 2014, pp. 33–40. doi:10.1007/978-3-319-10470-6_5.
URL http://link.springer.com/10.1007/978-3-319-10470-6{_}5

[25] S. Niroomandi, I. Alfaro, E. Cueto, F. Chinesta, Accounting for large deformations in real-time simulations of soft tissues based on reduced-order models, Computer Methods and Programs in Biomedicine 105 (1) (2012) 1–12. doi:10.1016/j.cmpb.2010.06.012.
URL http://dx.doi.org/10.1016/j.cmpb.2010.06.012

[26] S. A. Maas, B. J. Ellis, G. A. Ateshian, J. A. Weiss, FEBio: finite elements for biomechanics., Journal of biomechanical engineering 134 (1) (2012) 011005. doi:10.1115/1.4005694.
URL http://www.ncbi.nlm.nih.gov/pubmed/22482660

[27] A. Myronenko, X. Song, Point set registration: coherent point drift., IEEE transactions on pattern analysis and machine intelligence 32 (12) (2010) 2262–75. arXiv:0905.2635, doi:10.1109/TPAMI.2010.46.
URL http://ieeexplore.ieee.org/document/5432191/

[28] D. González, E. Cueto, F. Chinesta, Computational Patient Avatars for Surgery Planning., Annals of biomedical engineering 44 (1) (2016) 35–45. doi:10.1007/s10439-015-1362-z.
URL http://link.springer.com/10.1007/s10439-015-1362-z

[29] D. González, J. V. Aguado, E. Cueto, E. Abisset-Chavanne, F. Chinesta, kPCA-Based Parametric Solutions Within the PGD Framework, Archives of Computational Methods in Engineering 25 (1) (2018) 69–86. doi:10.1007/s11831-016-9173-4.
URL http://link.springer.com/10.1007/s11831-016-9173-4

[30] L. Breiman, Classification And Regression Trees, Routledge, 2017. arXiv:arXiv:1011.1669v3, doi:10.1201/9781315139470.
URL https://www.taylorfrancis.com/books/9781315139470

[31] S. Ruder, An overview of gradient descent optimization algorithms (2016) 1–14arXiv:1609.04747.
URL http://arxiv.org/abs/1609.04747

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, The Journal of Machine Learning Research (JMLR) 15 (2014) 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html

[33] F. Chollet, Keras: Deep Learning library for Theano and TensorFlow, GitHub Repository (2015) 1–21.

[34] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, X. Zheng, TensorFlow: A system for large-scale machine learning, arXivarXiv:1605.08695.
URL http://arxiv.org/abs/1605.08695