Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 407 (2023) 115912
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Thermodynamics-informed neural networks for physically realistic mixed reality☆

Quercus Hernández[a], Alberto Badías[b], Francisco Chinesta[c,d], Elías Cueto[a,*]

[a] Aragon Institute of Engineering Research (I3A), Universidad de Zaragoza, Maria de Luna 3, E-50018 Zaragoza, Spain
[b] Higher Technical School of Industrial Engineering, Polytechnic University of Madrid, C. de José Gutiérrez Abascal, 2, 28006 Madrid, Spain
[c] ESI Chair and PIMM Lab, ENSAM ParisTech., 155 Boulevard de l'Hôpital, 75013, Paris, France
[d] CNRS@CREATE Ltd., CREATE Tower, #08-01, 1 Create Way, 138602, Singapore

Received 19 December 2022; received in revised form 18 January 2023; accepted 19 January 2023
Available online 9 February 2023

Abstract

The imminent impact of immersive technologies in society urges for active research in real-time and interactive physics simulation for virtual worlds to be realistic. In this context, realistic means to be compliant to the laws of physics. In this paper we present a method for computing the dynamic response of (possibly non-linear and dissipative) deformable objects induced by real-time user interactions in mixed reality using deep learning. The graph-based architecture of the method ensures the thermodynamic consistency of the predictions, whereas the visualization pipeline allows a natural and realistic user experience.

Two examples of virtual solids interacting with virtual or physical solids in mixed reality scenarios are provided to prove the performance of the method.

Keywords: Deep learning; Augmented reality; Thermodynamics; Structure preserving; Real-time simulation

1. Introduction

Computer science advances in the last decades led us to experience in a relatively short lapse of time three major technological innovations: the personal computer, the Internet, and mobile devices. Currently, we are at the beginning of a fourth paradigm of computing innovations involving immersive technologies such as Virtual Reality (VR), Augmented Reality (AR) or Mixed Reality (MR). All this is possible due to huge advances in machine learning techniques and hardware improvements applied to computer graphics and computer vision.

☆ This material is based upon work supported in part by the Army Research Laboratory and the Army Research Office under contract/grant number W911NF2210271. This work has been also partially funded by the Spanish Ministry of Science and Innovation, AEI/10.13039/501100011033, through Grant number PID2020-113463RB-C31, and by the *Primeros Proyectos Grant* from Polytechnic University of Madrid, ETSII-UPM22-PM01. The support of ESI Group through the Chairs at ENSAM Paris, the DesCartes programme under its Campus for Research Excellence and Technological Enterprise (CREATE) programme and Universidad de Zaragoza is also gratefully acknowledged.

* Corresponding author.
E-mail addresses: quercus@unizar.es (Q. Hernández), alberto.badias@upm.es (A. Badías), francisco.chinesta@ec-nantes.fr (F. Chinesta), ecueto@unizar.es (E. Cueto).

https://doi.org/10.1016/j.cma.2023.115912

It is clear that this new paradigm seeks to revolutionize technology in the next years and will have a great impact in society such as smart cities [1,2], new teaching methods [3] or economic paradigms [4]. Technology companies have already created numerous digital platforms, such as the Metaverse [5,6] or the Omniverse [7,8], in order to develop their own immersive technologies.

In most of the cases, virtual worlds are required to resemble our real world as much as possible, so many disciplines come into play (traditionally, computer graphics and computer vision). However, virtual worlds need to be dynamic rather than static so the user can responsively interact with a changing virtual world. From that perspective, physics simulation plays a major role and it is required to be real-time. On the one hand, current real-time physic engines rely on severe simplifications of the governing dynamical equations and are limited to very simple material models and constitutive phenomena. On the other hand, classical engineering methods for solid and fluid simulations, such as the finite element or finite differences methods, have the consistency of decades of theoretical research in terms of the convergence to a consistent physical solution but are too expensive to achieve real-time framerates. However, these last methods can be used to generate a rich and consistent database to train a fast AI accelerated by the recent advances in machine learning procedures.

In this work, we aim to merge the physical consistency of classical simulation methods with the speed of real-time physics engines using a deep learning approach. Although the formulation is general for a wide variety of dynamical systems, in this work we focus on nonlinear solid mechanics. The results are consistent with the laws of thermodynamics by construction and are able to achieve real-time performance in general load cases which were not previously seen by the network. In Section 2, we explore some of the related work involving real-time reality simulators and physics-informed deep learning. In Section 3 we introduce the thermodynamically-informed graph neural networks together with the vision and visualization system used to record the demo videos. Section 4 shows the application of the proposed algorithm in two different solids together with error plots, and future work, limitations and conclusions are provided in Section 5.

2. Related work

Real-time physics engines such as PhysX [9,10] or Havok [11] have mostly been developed in the videogame industry. Even if those engines allow to program custom dynamical models, they usually rely on simplified mass–spring models or rigid body dynamics to achieve high framerates in modern videogames. Multiple research lines remain open trying to leverage the physical consistency of the results with low computational requirements.

2.1. Model order reduction-based simulators

Several authors solved the mentioned problems by creating a reduced order model of the system. These methods consist of a two-stage procedure: an offline phase where the solution space is precomputed in a compressed representation, lying on a reduced-order manifold, and an online phase where the solution can be evaluated in real-time. The difference between each method lies on the specific projection technique used to compute the reduced manifold.

Classical linear methods such as Principal Component Analysis [12,13] or reduced basis [14,15] are fast and simple to implement but fail to capture more complex nonlinear phenomena. This inconvenience can be solved using nonlinear methods such as kernel-Principal Component Analysis [16], Locally Linear Embedding [17,18] or Proper Generalized Decomposition [19,20]. Those techniques have similar disadvantages: as the solution is already precomputed, they are unable to handle different mesh discretizations and fail to generalize to unseen configurations.

2.2. Deep learning based simulators

The use of deep learning in real-time simulations has been widely explored in recent manuscripts, using neural networks as powerful function approximators with fast evaluation performance. The spirit is similar to the reduced order modeling: in the offline phase a neural network is trained with a set of examples and in the online phase the network can be fast evaluated in unseen scenarios.

For instance, several works avoid the real-time restrictions by decreasing the dimensionality of the problem by using autoencoders [21,22] in a similar fashion to the reduced order modeling methods. Other approaches are based on standard multilayer perceptrons, used as a collocation method for residue minimization [23,24]

or specific formulations for contact mechanics [25,26]. Those approaches require the prior knowledge of the governing equations. In the field of physics-informed machine learning, many methods have recently proposed to use neural networks such as solvers of partial differential equations (PDE) residues in the context of general physics [27,28] or fluid mechanics [29–31], identification and simulation of the dynamics of complex systems [32] or structure-preserving algorithms [33–35] with promising results. However, none of the mentioned methods have been implemented nor tested in an augmented or virtual reality setup, to the best of our knowledge.

This work presents a method to simulate real-time dynamics of one or more virtual systems interacting with physical objects in a mixed reality application. We require no governing equation, as it is learned from data using a thermodynamics-based formulation for non-equilibrium dynamical systems together with geometric deep learning. Furthermore, the trained graph neural network achieves real-time performance which enables a smooth user experience in the interaction of several virtual objects.

3. Methodology

3.1. Problem statement

The present work focuses on the deformation of virtual solid objects. Thus, we use the dynamical equilibrium equation of non-linear solid mechanics which balances the external and internal body forces with the acceleration of the solid. This is,

$$\nabla \boldsymbol{P} + \boldsymbol{B} = \rho \ddot{\boldsymbol{u}} \text{ in } \Omega_0, \tag{1}$$

where $\boldsymbol{B}$ represents the volumetric force applied to the body and $\boldsymbol{P}$ the first Piola–Kirchhoff stress tensor. $\Omega_0 = \Omega(t = 0)$ represents the undeformed configuration of the virtual solid. The solution is subjected to appropriate boundary conditions

$$\begin{aligned} \boldsymbol{u}(\boldsymbol{X}) &= \bar{\boldsymbol{u}} \text{ on } \Gamma_u, \\ \boldsymbol{P}\boldsymbol{N} &= \bar{\boldsymbol{t}} \text{ on } \Gamma_t, \end{aligned}$$

with Γ_u and Γ_t representing the essential (Dirichlet) and natural (Neumann) portions of the boundary $\Gamma = \partial\Omega$ of the solid. $\boldsymbol{X}$ is the undeformed position, $\boldsymbol{N}$ is the unit vector normal to $\Gamma = \partial\Omega_0$ and $\bar{\boldsymbol{u}}$, $\bar{\boldsymbol{t}}$ are the applied displacement and traction respectively. To complete the problem, some relationship between kinematic variables (displacements, strain) and dynamic variables (stresses) must be assumed. The constitutive equation is here chosen to be hyperelastic, with a strain energy function per unit volume Ψ defined such that

$$\boldsymbol{S} = \frac{\partial \Psi}{\partial \boldsymbol{E}} \tag{2}$$

where $\boldsymbol{S}$ is the second Piola–Kirchhoff and $\boldsymbol{E}$ is the Green–Lagrange strain tensor. Viscoelastic effects are also considered using variable shear relaxation modulus via Prony series. The objective of the method is to solve Eq. (1) in a real-time interactive interface with a physics-based neural network trained with high fidelity solutions.

3.2. Thermodynamics-informed graph neural networks

We use a novel deep learning method [36] which aims to learn the dynamical evolution of a physical system using a graph-based approach. Its objective is to learn not the outcome of a given simulation under different conditions such as forces or boundary conditions, but to be able to learn the *physics* taking place, such that the learned simulator is not sensitive to changes in the mesh, for instance.

The graph neural network architecture is thus constructed on top of a graph structure $\mathcal{G} = (V, \mathcal{E}, \boldsymbol{g})$, where $V = \{1, \ldots, n\}$ is a set of $|V| = n$ vertices, $\mathcal{E} \subseteq V \times V$ is a set of $|\mathcal{E}| = e$ edges and $\boldsymbol{g} \in \mathbf{R}^{F_g}$ is the global feature vector. Each vertex and edge in the graph is associated with a node in the finite element model from which data are obtained. The global feature vector defines the properties shared by all the nodes in the graph, such as constitutive properties.

To ensure translational invariance of the learned model, the position variables of the system $\boldsymbol{q}_i$, are assigned to the edge feature vector $\boldsymbol{e}_{ij} \in \mathbb{R}^{F_e}$ so the edge features represent relative distances ($\boldsymbol{q}_{ij} = \boldsymbol{q}_i - \boldsymbol{q}_j$) between nodes. The rest of the state variables are assigned to the node feature vector $\boldsymbol{v}_i \in \mathbb{R}^{F_v}$, while the external forces are included in

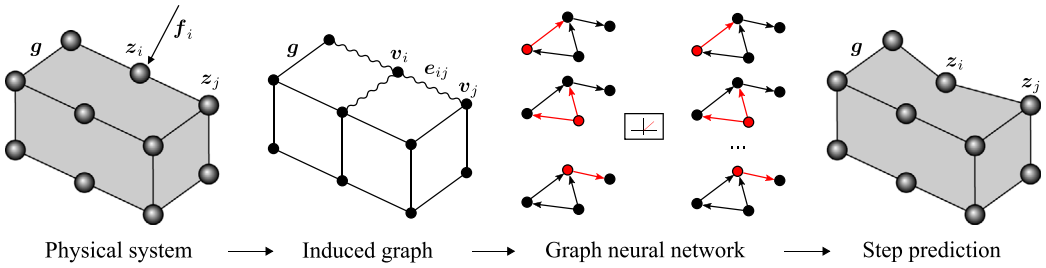

Fig. 1. Thermodynamics-informed graph neural network architecture. The system is described as a set of state variables z_i, global simulations parameters $\boldsymbol{g}$ and external boundary conditions $\boldsymbol{f}_i$. A graph $\mathcal{G}$ is constructed from the information of the physical system, defining vertex features $\boldsymbol{v}_i$, edge features $\boldsymbol{e}_{ij}$ and global features $\boldsymbol{g}$. The graph features are processed with a message-passing graph neural network. The step prediction is performed using the GENERIC integration scheme in Eq. (3), which is repeated iteratively to get the complete rollout of the simulation.

a vector $\boldsymbol{f}_i \in \mathbb{R}^{F_f}$. We employ an encode-process-decode scheme [37], built upon multilayer perceptrons (MLPs) shared between all the nodes and edges of the graph.

We use this architecture to learn the GENERIC structure of the evolution in time of the variables governing the virtual system [38,39]. It consists on splitting the system into a conservative and dissipative contribution. The conservative dynamics are defined using an energy potential E and a symplectic Poisson matrix $\boldsymbol{L}$, which recover the Hamiltonian formalism, whereas the dissipative dynamics are described by the entropy potential S and the dissipative or friction matrix $\boldsymbol{M}$, which accounts for the non-reversible dynamics. The time evolution of the state variables of the system $\boldsymbol{z}$ is described by the following equation

$$\frac{d\boldsymbol{z}}{dt} = \boldsymbol{L}\nabla E + \boldsymbol{M}\nabla S. \tag{3}$$

By enforcing the so called degeneracy conditions $\boldsymbol{L}\nabla S = \boldsymbol{M}\nabla E = \boldsymbol{0}$ together with the algebraic properties of the $\boldsymbol{L}$ (skew-symmetric) and $\boldsymbol{M}$ (symmetric and positive semidefinite) matrices we ensure the energy conservation and the entropy inequality. Thus, we guarantee the thermodynamical consistency of the predictions. The neural network learns the parametrization of the Poisson and friction operators in lower triangular matrices, $\boldsymbol{l}$ and $\boldsymbol{m}$ respectively, and the energy and entropy potentials, E and S. The GENERIC operators are then assembled as $\boldsymbol{L} = \boldsymbol{l} - \boldsymbol{l}^\top$ and $\boldsymbol{M} = \boldsymbol{m}\boldsymbol{m}^\top$, which enforces the skew-symmetry and positive semi-definiteness of the operators. A scheme of the algorithm is presented in Fig. 1.

We assume our virtual solids to be viscous-hyperelastic, so that the state variables for the proper description of their evolution in terms of the GENERIC formalism are the position $\boldsymbol{q}$, velocity $\boldsymbol{v}$ and the stress tensor $\boldsymbol{\sigma}$,

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}, \boldsymbol{v}, \boldsymbol{\sigma}) \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R}^6\}. \tag{4}$$

The edge feature vector contains the relative deformed position between nodes, to give a distance-based attentional flavor to the graph processing blocks and translational invariance. The velocity and stress tensor components are part of the node feature vector, concatenated to a two-dimensional one-hot vector $\boldsymbol{n}$ which represent the encastre and solid nodes respectively. The external load vector $\boldsymbol{f}_i$ is included in the node processor MLP as an external interaction. No global feature vector is needed in this case, resulting in the following feature vectors:

$$\boldsymbol{e}_{ij} = (\boldsymbol{q}_i - \boldsymbol{q}_j, \|\boldsymbol{q}_{ij}\|_2), \quad \boldsymbol{v}_i = (\boldsymbol{v}, \boldsymbol{\sigma}, \boldsymbol{n}). \tag{5}$$

Thus, the dimensions of the feature vectors are $F_e = 4$, $F_v = 11$, $F_f = 3$ and $F_g = 0$.

3.3. Vision system

For an augmented or mixed reality application, we need to include virtual objects in a real scene. For that, it is necessary to have a sensor able to get information about the physical environment and a screen device to plot the resulting image. In the present work, we use a ZED Mini stereo vision system from Stereolabs, which is able to retrieve both a depth and RGB image of the captured snapshot. We plot the resulting real-time video stream in a computer screen, but could be extended to a VR headset or AR glasses.

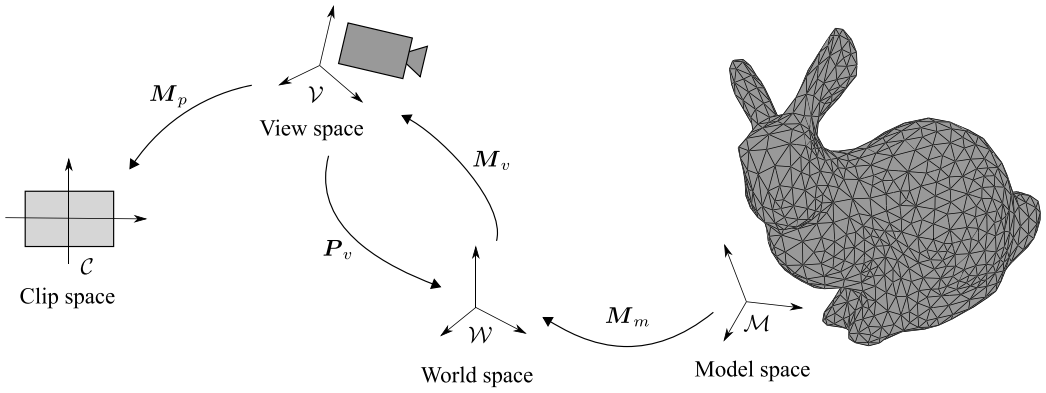

Fig. 2. Model-view-projection transformation between the model and the clip coordinates. The dynamics and interactions are computed in the model space as 3D coordinates, whereas the visualization pipeline requires a 2D clipping coordinates in a fixed range between -1 and 1.

As the laws of motion are independent with respect to any frame of reference (objectivity), we use the model space to compute the dynamical state variables of the system. However, the visualization pipeline requires the coordinates to be in a normalized range of $\mathcal{C} = [-1, 1] \times [-1, 1]$ called the clip space. The transformation between the 3D model space and the 2D clip space is achieved by the definition of the Model-View-Projection matrix (see Fig. 2). From now, the point coordinates are supposed to be in homogeneous coordinates.

- Model: The dynamics are computed in a local frame of reference called the model space $\mathcal{M}$. The position, orientation and scale of the model can be defined as a set of transformation matrices $\boldsymbol{T}_m$, $\boldsymbol{R}_m$ and $\boldsymbol{s}_m$ respectively with respect to the world space $\mathcal{W}$, considered to be the usual Euclidean space $\mathbb{R}^3$.

$$\boldsymbol{x}_{\text{world}} = \boldsymbol{M}_m \boldsymbol{x}_{\text{model}} = \boldsymbol{T}_m \boldsymbol{R}_m \boldsymbol{s}_m \boldsymbol{x}_{\text{model}}. \tag{6}$$

- View: In a similar fashion, the viewing camera also has a model matrix defining its position in the world space, which is commonly designed as the extrinsic parameters $\boldsymbol{P}_v$ of the camera pose. This information can be obtained using a monocular pose estimator or triangulating with a stereo vision system. Thus, the camera or view space $\mathcal{V}$ can be determined by a set of transformations from the world space using the camera extrinsic parameters given by a rotation $\boldsymbol{R}_v$ and translation $\boldsymbol{T}_v$ matrices.

$$\boldsymbol{x}_{\text{view}} = \boldsymbol{M}_v \boldsymbol{x}_{\text{world}} = \boldsymbol{P}_v{}^{-1} \boldsymbol{x}_{\text{world}} = (\boldsymbol{T}_v \boldsymbol{R}_v)^{-1} \boldsymbol{x}_{\text{world}}. \tag{7}$$

- Projection: The last transformation is in charge of projecting the 3D coordinates into the 2D clip space $\mathcal{C}$. First, in order to get a realistic visualization, we use a perspective projection $\boldsymbol{M}_p$ based on the camera viewing frustum

$$\boldsymbol{M}_p = \begin{pmatrix} \cot\frac{\alpha}{2} & 0 & 0 & 0 \\ 0 & \cot\frac{\alpha}{2} & 0 & 0 \\ 0 & 0 & -\frac{z_{\text{far}}+z_{\text{near}}}{z_{\text{far}}-z_{\text{near}}} & -\frac{2z_{\text{far}}z_{\text{near}}}{z_{\text{far}}-z_{\text{near}}} \\ 0 & 0 & -1 & 0 \end{pmatrix}, \tag{8}$$

  where α is the field of view of the camera and z_{near} and z_{far} are the minimum and maximum distance of the clipping plane. This creates a normalized 3D viewing box of the camera, which is then projected in the 2D clip space by the vertex shader. Thus, the final coordinates can be computed using the following equation:

$$\boldsymbol{x}_{\text{clip}} = \boldsymbol{M}_p \boldsymbol{x}_{\text{view}} = \boldsymbol{M}_p \boldsymbol{M}_v \boldsymbol{M}_m \boldsymbol{x}_{\text{model}}. \tag{9}$$

By defining the Model-View-Projection matrix as $\boldsymbol{M}_p \boldsymbol{M}_v \boldsymbol{M}_m$ we can compute the clip coordinates directly from the model coordinates computed by the thermodynamics-informed neural network.

3.4. Visualization system

The visualization of the resulting image is performed using OpenGL and the GLU library. This pipeline requires the definition of two spatial functions, run sequentially on the GPU, called the vertex and fragment shaders. The vertex shader defines the vertex position of the entities to display whereas the fragment (or texture) shader define the RGBA colors for each rasterized pixel.

Each vertex position is computed using Eq. (9) from the deformed configuration coordinates and the polygons are drawn based on the connectivity matrix of each solid. The color of each vertex is computed directly from the neural network prediction using a fixed colormap. Additionally, a basic lighting model was added to the fragment shader to increase the realism of the virtual objects using the Phong shading. This model computes each RGB pixel intensity as

$$\boldsymbol{I} = k_a \boldsymbol{i}_a + k_d(\boldsymbol{l} \cdot \boldsymbol{n})\boldsymbol{i}_d + k_s(\boldsymbol{r} \cdot \boldsymbol{v})^{\beta}\boldsymbol{i}_s, \tag{10}$$

where k_a, k_d and k_s are the ambient, diffuse and specular reflection constants, β is the shininess constant of the material and $\boldsymbol{i}_a$, $\boldsymbol{i}_d$ and $\boldsymbol{i}_s$ are the RGB color intensities of the ambient, diffuse and specular components. The illumination varies depending on the geometry of the scene and camera position, where $\boldsymbol{n}$ is the surface normal vector, $\boldsymbol{l}$ and $\boldsymbol{r}$ are the directions of the light source and its perfect reflection, $\boldsymbol{v}$ is the viewing direction and "$\cdot$" is the dot product.

We have also implemented a depth or z-buffer in the fragment shader which compares on each pixel the depth of virtual and real objects on the scene and renders only the closest to the camera. This accounts for every occlusion that the vision system may encounter. The depth of the scene seen by the camera is computed directly using the stereo vision system.

3.5. Collision and contact

We use a high-fidelity hand tracker from MediaPipe [40] which provides an accurate localization of the finger tips. By using the inverse transformation of Eq. (9), we can compute the 3D coordinates of the finger tips in the model space and compute the distance to each node of the virtual object. When this distance is less than a small threshold, collision is detected and a prescribed force is applied to the model. A similar procedure is applied to the collision of several virtual objects.

The code is implemented in Python using the PyOpenGL wrapper for the visualization and Pytorch Geometric [41] for the deep learning training and evaluation. The videos are generated using a standard desktop computer with a single Nvidia RTX2070 GPU and provided as supplementary material. The code for the implementation of the thermodynamics-informed neural networks and the video sequences are publicly available at https://github.com/quercushernandez.

4. Results

4.1. Bending beams

The first system is composed of two interacting viscoelastic beams. Both identical beams are assembled with a small gap between each other, allowing for a contact interaction, as depicted in Fig. 3.

The dimensions of the beams are $H = 10$, $W = 10$ and $L = 40$. The finite element mesh from which data are obtained consists of $N_e = 500$ hexahedral linear brick elements and $N = 756$ nodes. The constitutive parameters of the hyperelastic strain energy potential are $C_{10} = 1.5 \cdot 10^5$, $C_{01} = 5 \cdot 10^3$, $D_1 = 10^{-7}$ and $\bar{g}_1 = 0.3$, $\bar{g}_2 = 0.49$, $\tau_1 = 0.2$, $\tau_2 = 0.5$ respectively for the two-term Prony series. A distributed load of $F = 10^5$ is applied in 52 different positions with a perpendicular direction to the solid surface. Each simulation is composed of $N_T = 20$ time increments of $\Delta t = 5 \cdot 10^{-2}$ s. Both beams are assembled in 90° with a gap of $g = 10$.

The graph neural network vertex and edge MLPs have two layers of $F_h = 100$ neurons each, with 10 message passing sequential blocks. The training was performed for $N_{\text{epoch}} = 1800$ epochs and learning rate $l_r = 10^{-4}$.

Fig. 4 shows a real-time video sequence generated using the presented algorithm. The interaction of both the real objects (finger tips) and the virtual objects are simulated smoothly at more than 30 frames per second. It is important to highlight that during a video sequence the trained neural network is able to generalize to previously

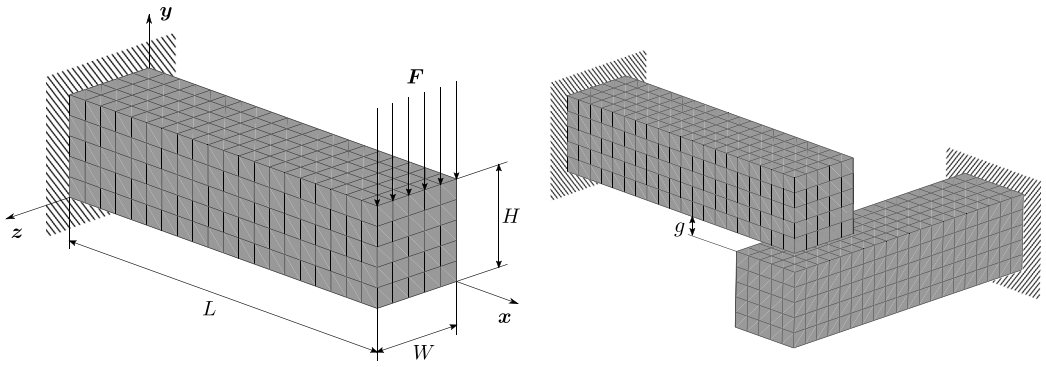

Fig. 3. Interacting beams geometry scheme. Both beams are angled with a gap between both, enabling their interaction.

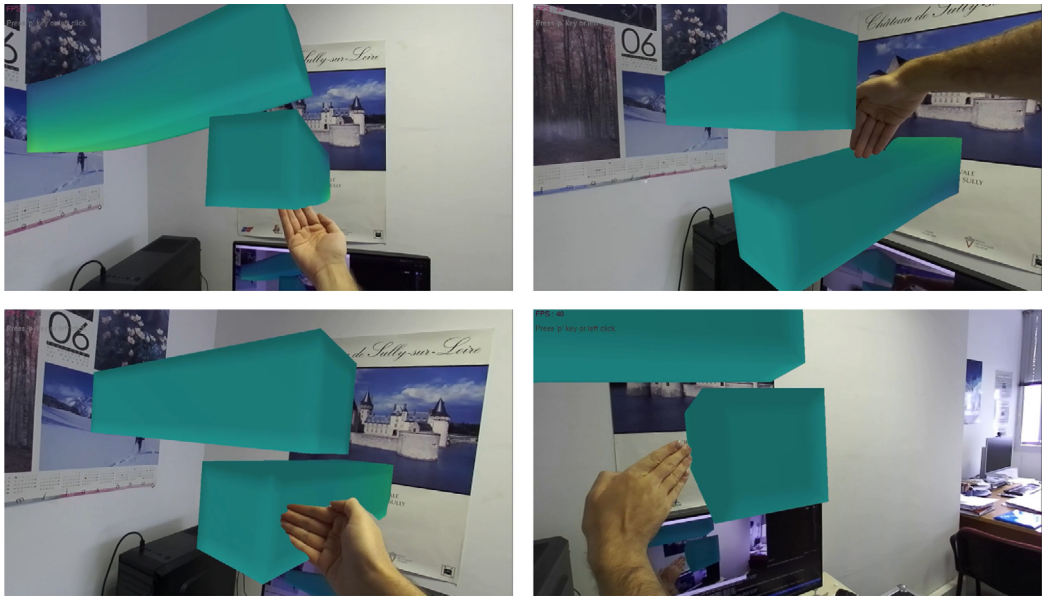

Fig. 4. Frames extracted from the interacting beams sequence. Color encodes the x-x stress field component associated with the displacement imposed by contact with a real object.

unseen configurations. The quantitative errors by the neural network in the real-time rollout predictions are shown in Fig. 7a, which remain below 1% in position and velocity and 10% in the stress tensor field.

The computational cost of the high fidelity FEM simulations used for the training of the neural network is 15 s per simulation, up to 795 s for the whole dataset, and 243 Mb in memory storage. Conversely, the training time of the neural network is 4.5 h and the mean evaluation time is 9 ms with a memory storage of 12.3 Mb. Thus, the use of a deep learning approach allows a drastic reduction of the online computation time with the inconvenient of larger offline training time. It is also worth noting that the network parameters are more than 10 times lighter in terms of memory storage.

4.2. Stanford bunny

The second example is a bunny mesh from the Stanford 3D scanning repository, which is a more complex geometry as the one shown in the previous example. It is a standard reference model in computer graphics research. The model is represented in Fig. 5.

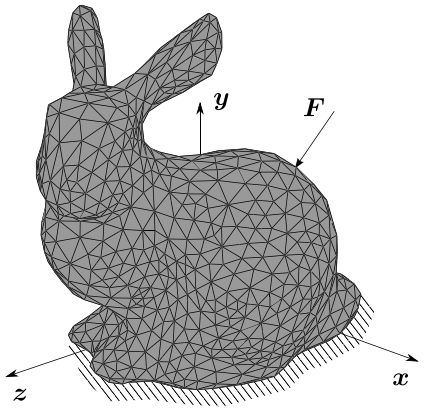

Fig. 5. Bunny mesh geometry with a concentrated application force and bottom encastre.

The finite element mesh from which data are obtained consisted of $N_e = 4941$ tetrahedral linear elements and $N = 1352$ nodes. The constitutive parameters of the hyperelastic strain energy potential are $C_{10} = 2.6 \cdot 10^{-1}$ and $D_1 = 4.9 \cdot 10^{-2}$. Similarly as the previous case, a concentrated load of $F = 1$ is applied in 100 different positions with a perpendicular direction to the solid surface. The body was fixed to the ground plane by disabling displacements and rotations at the lower model nodes. Each simulation is composed of $N_T = 20$ time increments of $\Delta t = 5 \cdot 10^{-2}$ s.

The graph neural network vertex and edge MLPs have two layers of $F_h = 100$ neurons each, with 10 message passing sequential blocks. The training was performed for $N_{\text{epoch}} = 1800$ epochs and learning rate $l_r = 10^{-4}$.

Fig. 6 shows a real-time video sequence generated using the bunny model, also with a minimum framerate of 30 frames per second. In this case the stress field errors reported in Fig. 7b are higher due to the stress peaks at the single-node force application, which causes that the elasticity phenomena remain very local in space.

In this case, the computational cost of the high fidelity FEM simulations is 27 s per simulation, up to 2700 s for the whole dataset and 833 Mb in memory storage. The dataset and edge count is much larger in this example, which increases the training time of the neural network to 20 h and the evaluation time to 11 ms with the same memory storage as the previous example. Thus, the data compression is even bigger in this case. To address the high training time and errors due to stress peaks, several future improvements are discussed in the next section.

5. Conclusions

We presented a real-time augmented reality simulator, which enables a user to interact with virtual deformable solids. The predictions are computed using the GENERIC structure of the system learnt with a message passing graph neural network. The enforcement of such physics constraints guarantees the fulfillment of the first and second laws of thermodynamics. The resulting algorithm has a wide variety of applications not only in the entertainment industry, but also in engineering design or manufacturing, where the visualization of augmented data superimposed in a real or virtual object might redefine the next generation of industry 4.0 and digital twins.

Our method has several limitations which might be addressed as future work. The scalability to bigger meshes is a challenging task, as graph neural networks can suffer from over-squashing or bottlenecks [42,43]. The use of graph reduction techniques via graph autoencoders [44] or U-nets [45] configurations can significantly reduce the computational requirements in larger meshes. This would also reduce the stress peaks discontinuities due to concentrated forces. The message passing algorithm itself is not able to handle domains where the boundary conditions are very far away from the prescribed forces. This can be mitigated by implementing a global attention vector so that certain dynamical information is reached to all the nodes of the domain instantly.

The visualization and data acquisition are possible due to the graphics acceleration. Even if the current work was implemented in a desktop computer as a proof of concept, only a small fraction of all the computational resources were used, so it can be extended to AR/VR headsets or to modern mobile devices. For the same reason, higher

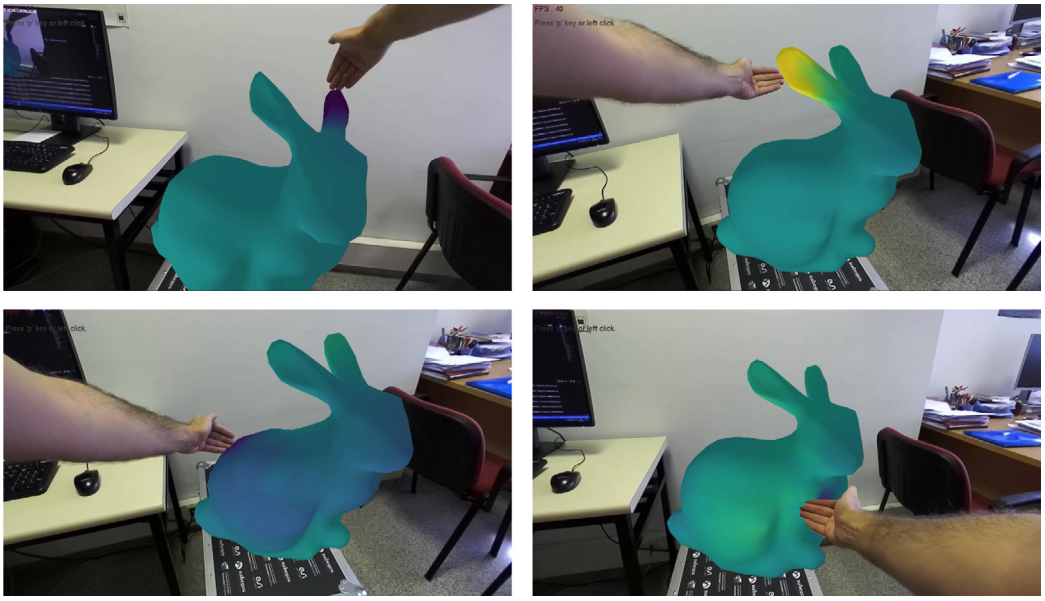

Fig. 6. Frames extracted from the bunny sequence. Color encode the x displacement field component imposed by contact with a real object.

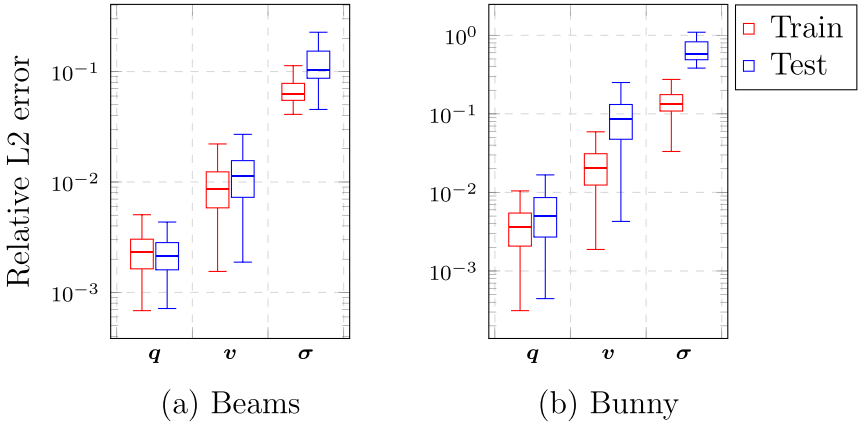

Fig. 7. Box plots for the relative L2 error for all the rollout snapshots of the bunny dataset in both train and test cases. The state variables represented are position ($\boldsymbol{q}$), velocity ($\boldsymbol{v}$), and stress tensor ($\boldsymbol{\sigma}$).

framerates might be achieved by fine-tuning and optimizing the proposed network structure or porting the code to a higher-performance language such as C++. Occlusions in real-time augmented reality software still remain as an open problem in the computer vision community [46,47]. The stereo depth estimation is an approximation which might cause image artifacts in singular camera poses. For instance, some works handle the problem by using deep learning [48] but it is out of the scope of this manuscript.

This work is just a small step forward towards the new immersive technologies which can potentially deeply change our society. It is a multidisciplinary problem where computer vision, computer graphics, machine learning and computational mechanics must meet to define new algorithms for a new digital revolution.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Elias Cueto reports financial support was provided by Spanish Ministry of Science. Elias Cueto

reports financial support was provided by ESI Group. Francisco Chinesta reports financial support was provided by ESI Group. Elias Cueto reports financial support was provided by Army Research Office.

Data availability

Data will be made available on request.

References

[1] Zaheer Allam, Ayyoob Sharifi, Simon Elias Bibri, David Sydney Jones, John Krogstie, The metaverse as a virtual form of smart cities: opportunities and challenges for environmental, economic, and social sustainability in urban futures, Smart Cities 5 (3) (2022) 771–801.

[2] Vivek Veeraiah, P Gangavathi, Shahanawai Ahamad, Suryansh Bhaskar Talukdar, Ankur Gupta, Veera Talukdar, Enhancement of meta verse capabilities by IoT integration, in: 2022 2nd International Conference on Advance Computing and Innovative Technologies in Engineering, ICACITE, IEEE, 2022, pp. 1493–1498.

[3] Pericles 'asher' Rospigliosi, Metaverse or simulacra? Roblox, minecraft, meta and the turn to virtual reality for education, socialisation and work, Interact. Learn. Environ. 30 (1) (2022) 1–3.

[4] Fei-Yue Wang, Rui Qin, Xiao Wang, Bin Hu, Metasocieties in metaverse: Metaeconomics and metamanagement for metaenterprises and metacities, IEEE Trans. Comput. Soc. Syst. 9 (1) (2022) 2–7.

[5] Stylianos Mystakidis, Metaverse, Encyclopedia 2 (1) (2022) 486–497.

[6] Sascha Kraus, Dominik K Kanbach, Peter M Krysta, Maurice M Steinhoff, Nino Tomini, Facebook and the creation of the metaverse: radical business model innovation or incremental transformation? Int. J. Entrepreneurial Behav. Res. (2022).

[7] Mathias Hummel, Kees van Kooten, Leveraging nvidia omniverse for in situ visualization, in: International Conference on High Performance Computing, Springer, 2019, pp. 634–642.

[8] Xiao Li, Baris Can Yalcin, Olga-Orsalia Christidi-Loumpasefski, Carol Martinez Luna, Maxime Hubert Delisle, Gonzalo Rodriguez, James Zheng, Miguel Angel Olivares Mendez, Exploring NVIDIA omniverse for future space resources missions, 2022.

[9] Anderson Maciel, Tansel Halic, Zhonghua Lu, Luciana P Nedel, Suvranu De, Using the PhysX engine for physics-based virtual surgery with force feedback, Int. J. Med. Robotics Comput. Assist. Surg. 5 (3) (2009) 341–353.

[10] Antonio D'Andrea, Monica Reggiani, Andrea Turolla, Davide Cattin, Roberto Oboe, A PhysX-based framework to develop rehabilitation using haptic and virtual reality, in: 2013 IEEE International Symposium on Industrial Electronics, IEEE, 2013, pp. 1–6.

[11] Lei Wang, Zhenwen Wang, Hua Xu, A method for 3D rock fracturing simulation based Havok, in: 2015 6th IEEE International Conference on Software Engineering and Service Science, ICSESS, IEEE, 2015, pp. 898–901.

[12] Gal Berkooz, Philip Holmes, John L. Lumley, The proper orthogonal decomposition in the analysis of turbulent flows, Annu. Rev. Fluid Mech. 25 (1) (1993) 539–575.

[13] Ritesh R. Rama, Sebastian Skatulla, Carlo Sansour, Real-time modelling of diastolic filling of the heart using the proper orthogonal decomposition with interpolation, Int. J. Solids Struct. 96 (2016) 409–422.

[14] Christophe Prud'Homme, Dimitrios V Rovas, Karen Veroy, Luc Machiels, Yvon Maday, Anthony T Patera, Gabriel Turinici, Reliable real-time solution of parametrized partial differential equations: Reduced-basis output bound methods, J. Fluids Eng. 124 (1) (2002) 70–80.

[15] Andrea Manzoni, Filippo Salmoiraghi, Luca Heltai, Reduced basis isogeometric methods (RB-IGA) for the real-time simulation of potential flows about parametrized NACA airfoils, Comput. Methods Appl. Mech. Engrg. 284 (2015) 1147–1180.

[16] Bernhard Schölkopf, Alexander Smola, Klaus-Robert Müller, Kernel principal component analysis, in: International Conference on Artificial Neural Networks, Springer, 1997, pp. 583–588.

[17] Sam T. Roweis, Lawrence K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.

[18] Beatriz Moya, David González, Icíar Alfaro, Francisco Chinesta, Elias Cueto, Learning slosh dynamics by means of data, Comput. Mech. 64 (2) (2019) 511–523.

[19] Alberto Badías, David González, Iciar Alfaro, Francisco Chinesta, Elias Cueto, Local proper generalized decomposition, Internat. J. Numer. Methods Engrg. 112 (12) (2017) 1715–1732.

[20] Alberto Badías, David González, Icíar Alfaro, Francisco Chinesta, Elías Cueto, Real-time interaction of virtual and physical objects in mixed reality applications, Internat. J. Numer. Methods Engrg. 121 (17) (2020) 3849–3868.

[21] Lawson Fulton, Vismay Modi, David Duvenaud, David IW Levin, Alec Jacobson, Latent-space dynamics for reduced deformable simulation, in: Computer Graphics Forum, Vol. 38, No. 2, Wiley Online Library, 2019, pp. 379–391.

[22] Peter Yichen Chen, Jinxu Xiang, Dong Heon Cho, Yue Chang, GA Pershing, Henrique Teles Maia, Maurizio Chiaramonte, Kevin Carlberg, Eitan Grinspun, CROM: Continuous reduced-order modeling of PDEs using implicit neural representations, 2022, arXiv preprint arXiv:2206.02607.

[23] Stefania Fresca, Giorgio Gobat, Patrick Fedeli, Attilio Frangi, Andrea Manzoni, Deep learning-based reduced order models for the real-time simulation of the nonlinear dynamics of microstructures, Internat. J. Numer. Methods Engrg. 123 (20) (2022) 4749–4777.

[24] Alban Odot, Ryadh Haferssas, Stéphane Cotin, DeepPhysics: A physics aware deep learning framework for real-time simulation, Internat. J. Numer. Methods Engrg. 123 (10) (2022) 2381–2398.

[25] Javier Romero, Dimitrios Tzionas, Michael J. Black, Embodied hands: Modeling and capturing hands and bodies together, 2022, arXiv preprint arXiv:2201.02610.

[26] Cristian Romero, Dan Casas, Maurizio M Chiaramonte, Miguel A Otaduy, Contact-centric deformation learning, ACM Trans. Graph. 41 (4) (2022) 1–11.
[27] Maziar Raissi, George Em Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, J. Comput. Phys. 357 (2018) 125–141.
[28] Maziar Raissi, Paris Perdikaris, George E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[29] Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, Ricardo Vinuesa, Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations, Phys. Fluids 34 (7) (2022) 075117.
[30] Hamidreza Eivazi, Ricardo Vinuesa, Physics-informed deep-learning applications to experimental fluid mechanics, 2022, arXiv preprint arXiv:2203.15402.
[31] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, George Em Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mech. Sinica (2022) 1–12.
[32] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, Peter Battaglia, Learning to simulate complex physics with graph networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.
[33] Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, Peter Battaglia, Hamiltonian graph networks with ode integrators, 2019, arXiv preprint arXiv:1909.12790.
[34] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, Léon Bottou, Symplectic recurrent neural networks, 2019, arXiv preprint arXiv:1909.13334.
[35] Quercus Hernández, Alberto Badías, David González, Francisco Chinesta, Elías Cueto, Structure-preserving neural networks, J. Comput. Phys. 426 (2021) 109950.
[36] Quercus Hernández, Alberto Badías, Francisco Chinesta, Elías Cueto, Thermodynamics-informed graph neural networks, 2022, arXiv preprint arXiv:2203.01874.
[37] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al., Relational inductive biases, deep learning, and graph networks, 2018, arXiv preprint arXiv:1806.01261.
[38] Hans Christian Öttinger, Miroslav Grmela, Dynamics and thermodynamics of complex fluids. II. Illustrations of a general formalism, Phys. Rev. E 56 (6) (1997) 6633.
[39] Miroslav Grmela, Hans Christian Öttinger, Dynamics and thermodynamics of complex fluids. I. Development of a general formalism, Phys. Rev. E 56 (6) (1997) 6620.
[40] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, Matthias Grundmann, Mediapipe hands: On-device real-time hand tracking, 2020, arXiv preprint arXiv:2006.10214.
[41] Matthias Fey, Jan Eric Lenssen, Fast graph representation learning with PyTorch Geometric, 2019, arXiv preprint arXiv:1903.02428.
[42] Uri Alon, Eran Yahav, On the bottleneck of graph neural networks and its practical implications, 2020, arXiv preprint arXiv:2006.05205.
[43] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, Michael M Bronstein, Understanding over-squashing and bottlenecks on graphs via curvature, 2021, arXiv preprint arXiv:2111.14522.
[44] Thomas N. Kipf, Max Welling, Variational graph auto-encoders, 2016, arXiv preprint arXiv:1611.07308.
[45] Hongyang Gao, Shuiwang Ji, Graph u-nets, in: International Conference on Machine Learning, PMLR, 2019, pp. 2083–2092.
[46] Mathias Parger, Chengcheng Tang, Yuanlu Xu, Christopher David Twigg, Lingling Tao, Yijing Li, Robert Wang, Markus Steinberger, UNOC: Understanding occlusion for embodied presence in virtual reality, IEEE Trans. Vis. Comput. Graphics (2021).
[47] Wei Yan, Augmented reality instructions for construction toys enabled by accurate model registration and realistic object/hand occlusions, Virtual Real. 26 (2) (2022) 465–478.
[48] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al., Lookingood: Enhancing performance capture with real-time neural re-rendering, 2018, arXiv preprint arXiv:1811.05029.