PHYSICS PERCEPTION IN SLOSHING SCENES WITH GUARANTEED THERMODYNAMIC CONSISTENCY

A PREPRINT

Beatriz Moya
Aragon Institute in Engineering Research
University of Zaragoza
Zaragoza, Spain
beam@unizar.es

Alberto Badias
Aragon Institute in Engineering Research
University of Zaragoza
Zaragoza, Spain
abadias@unizar.es

David Gonzalez
Aragon Institute in Engineering Research
University of Zaragoza
Zaragoza, Spain
gonzal@unizar.es

Francisco Chinesta
ESI Group chair. PIMM Lab.
ENSAM Institute of Technology
Paris, France
francisco.chinesta@ensam.eu

Elias Cueto
Aragon Institute in Engineering Research
University of Zaragoza
Zaragoza, Spain
ecueto@unizar.es

February 25, 2022

ABSTRACT

Physics perception very often faces the problem that only limited data or partial measurements on the scene are available. In this work, we propose a strategy to learn the full state of sloshing liquids from measurements of the free surface. Our approach is based on recurrent neural networks (RNN) that project the limited information available to a reduced-order manifold to not only reconstruct the unknown information but also be capable of performing fluid reasoning about future scenarios in real-time. To obtain physically consistent predictions, we train deep neural networks on the reduced-order manifold that, through the employ of inductive biases, ensure the fulfillment of the principles of thermodynamics. RNNs learn from history the required hidden information to correlate the limited information with the latent space where the simulation occurs. Finally, a decoder returns data to the high-dimensional manifold, to provide the user with insightful information in the form of augmented reality. This algorithm is connected to a computer vision system to test the performance of the proposed methodology with real information, resulting in a system capable of understanding and predicting future states of the observed fluid in real-time.

***K*eywords** Physics perception · thermodynamics-aware deep learning · GENERIC · sloshing

1 Introduction

Simulating the world consists in the emulation of real dynamics in a virtual environment. Models within these simulations enable robots to experience perception of the world around them, thus allowing them to reason about the consequences of their actions [1]. In this sense, fluid manipulation is a difficult task in AI-enabled robotics, and trustable physics-based simulation of liquids is desired for success [2] [3] [4].

The merge between data-driven learning and knowledge about physics is becoming popular in dynamical modeling, enabling the study of complex systems of highly non-linear nature [5]. This conjunction is known as "physics-informed deep learning". Very different techniques may fall under this broad classification. Despite the common interest in introducing well-known physical knowledge into these approaches, there is a great divergence of proposals to be considered. From solving PDEs [6] [7] to learning constitutive laws based on invariants and conserved quantities [8] [9], these works establish a strong framework to work towards generalizable deep learning, extending models' capacity to learn about new situations out of the range of training.

While these techniques keep improving, they face an important difficulty. In general, experimental campaigns to obtain the sets of governing variables are not always possible. Some works propose to model fluid dynamics with data coming from images [10] [11] or sensors [12], but that information could not be enough to form a physically consistent description. In contrast, some proposals try to reconstruct the missing dynamical data. Here, we propose an image-based method combined with physics knowledge to reconstruct the dynamics of the fluid from the free surface detected in video frames.

Information comes from camera recordings. This means that 30 to 120 snapshots per second will be available, and we expect a similar feedback rate of the model to provide the user with a smooth sensation. Under such stringent rates, encoding the information available to a low-dimensional manifold is mandatory. In this condition, we can establish a bridge between real systems with their digital twins [13] [14].

The present work aims to develop a thermodynamically admissible model for fluid dynamics understanding and reasoning. The simulation engine is coupled with a computer vision system to build online digital twins of fluids. The performance of the loop must achieve real-time speed to guarantee trustable decision-making.

The information we require in the proposed physics-informed description —not only position, but also the velocity and stress fields and internal energy— is inaccessible to a commodity depth camera. We do not consider here more sophisticated systems such as particle image velocimetry, for instance. We hypothesize that the knowledge of the internal variables we need for a complete description of the fluid will come from the study of the evolution of the free surface. We first train an autoencoder from full-field computational data coming from simulations. Then, we evaluate the information of the free surface in sequences to distill from its history the dynamical information needed to find a correlation with the latent manifold. We use a recurrent neural network (RNNs) to reconstruct the dynamical state and project it to a reduced-order space where simulations are performed to achieve real-time performance. On top of that, an augmented reconstruction is provided afterward, outputting not only the state of the full fluid volume, but also velocity, internal energy, and stress fields that were not accessible in the first place.

To guarantee the thermodynamic admissibility of the simulations, we learn from data a particular formulation of the dynamics based on the so-called General Equation for the Non-Equilibrium Reversible-Irreversible Coupling (GENERIC) formalism [15]. GENERIC constitutes a generalization of Hamiltonian dynamics to dissipative phenomena. Under the scope of GENERIC acting as an inductive bias [16], or learning constraint, during the learning procedure, we ensure the accomplishment of the basic principles of energy conservation and entropy generation, thus providing a physically sound learning framework.

The paper is structured as follows. Section 2 describes the state of the art and recent works in the field. Section 3 describes the problem in detail. Section 4 exposes the method, from the projection of the dynamics to a lower dimensional manifold to the learning algorithm and its connection with partial measurements. The training and implementation details are explained in section 5. Section 6 showcases the results obtained with real-world measurements. The paper finalizes with a discussion of the results and an evaluation of the future developments that can derive from this work.

2 Works in the field

2.1 Self-supervised estimation of dynamical states

Labeling is an indisputable bottleneck for data-driven prediction. Thus, we need a framework to substitute the need for labeling with a deeper understanding of the information available. This has clear importance in robotics and visual perception and understanding based on images [4] [17] [18] [19]. Dynamical modeling, and specifically fluid dynamics, also needs to address this lack of information when performing simulation and validation. Many internal variables employed in the descriptions are not easily measurable. Whereas some works use information obtained from images [10] [11] or sensors [12], many opt for reconstructing the dynamical internal state for an accurate description [20]. Dynamics typically use strategically placed sensors that acquire data to recover the full set of quantities from sparse observations [21] [22]. Several works employ deep neural networks (DNN) for this purpose. Erichson et al. [23] propose the use of shallow neural networks for reconstructing fluid flows. Lye et al. [24] estimate the unknown input parameters in turbulent flows from observables.

Our contribution consists in a hybrid proposal between image and physics-based reconstruction using computer vision analysis.

2.2 Deep learning incorporating physics priors

Despite the use of self-supervised learning to compensate for the lack of unlabeled data, supervised methods still need a large database for learning a model. Few data could jeopardize the accuracy of the approximations. That is the case of black-box schemes, which do not succeed in learning global expressions and efficient generalizations in low data regimes [25]. Inductive biases [16] guide the network to learn specific correlations in data. On top of that, they favor convergence and reduce error bounds [26]. As a result, less data is required, and the results are more realistic and accurate.

Physics-informed deep learning is a current trend in artificial intelligence. Many approaches leverage theoretical knowledge to improve algorithm training [6] [27] [28]. Hamiltonian (thus, conservative) systems are regular test benchmarks for these techniques [29] [30]. Nevertheless, systems of utmost importance, such as those that involve Newtonian and non-Newtonian fluid dynamics, require beyond-equilibrium schemes. Thermodynamic neural networks constitute a framework that enables the study of any physical system, including those of inherent dissipative nature [31]. Yu et al. [32] apply the generalized Onsager principle to infer relationships among the state variables that define the system to ensure the fulfillment of that principle. GENERIC [15] describes the changes over time of a set of state variables—that must characterize the energy of the system— to model its evolution of energy and entropy. The dynamics can be fully described at a coarse-grained scale by learning the manifold of its time progression [33]. This learning theory has been successfully applied to model rather different and complex behaviors [34] [35] [36] [37]. In recent works, it has been coupled with DNNs to build the so-called Structure-Preserving Neural Networks (SPNN) [38] [39].

If we focus on the field of fluid simulation, DNNs are an extended tool for its emulating [40]. [41] [42] apply CNNs to characterize 2D and 3D fluid dynamics. In the case of [43], the authors distill the dynamics of unsteady flows with RNNs. Following the same spirit of the latter work, [44] employs specifically LSTMs networks in reduced order manifolds. Graph neural networks are becoming popular in this field [45]. Regarding physically informed deep learning, there are also works related to the study of fluids. [46] apply PINNs to high-speed flows. The work of [47] presents an approach for learning PDEs from Physics-informed CNNs.

There exist plenty of approaches [45] [48] [49] [4] in the context of scene understanding and interaction with fluids. In [50], the authors propose closing the learning loop with observations that help to correct the errors of the simulation. This work implements Navier Stokes in conjunction with the theory of Smooth Particle Hydrodynamics [51]. In contrast, we propose purely data-driven learning in a reduced order space imposing physical priors to achieve accuracy and generalization, as well as real-time performance. We believe in the use of the GENERIC formalism to discover fast and accurately the patterns of the dynamics for perception and interaction tasks.

2.3 Computer vision in fluids estimation

Although the algorithm is trained with computational data, the final goal is to connect it with real liquids to close the perception loop. Detection and tracking of fluids, as well as containers, may be difficult if they lack texture. The measurements obtained are usually invalid or noisy because the surfaces are not Lambertian [52]. We are interested in the detection of fluids, particularly the free surface. [53] [50] propose the use of CNNs to perform tracking of the fluids. In the work of [54], the authors propose an algorithm for filling level detection with RGD-D cameras. We propose an approach similar to the one presented in the work [55]. We convert the color image into a binary image in black and white to detect the color gradient that appears on the free surface.

3 Problem description

In this work, we propose a strategy to train simulators for physical scene understanding. Similar strategies have been developed in recent times, see [49] for instance. However, in contrast with these works, we aim to develop a technique in which quantitative—and not only qualitative—information is given. This is of utmost importance if our perception system is to be employed in an industrial framework in the form of a digital twin of an asset, for instance. Thus, it is of primary importance to ensure that the learned simulator can make predictions that adhere to known basic principles of physics, such as the laws of thermodynamics.

We have considered the problem of fluid sloshing as a proof of concept for this analysis. This is in part due to its practical interest (to construct robots able to manipulate fluids, for instance) but also for its generality: sloshing is a (nonlinear) physical phenomenon that presents several interesting and challenging characteristics. Among them,

a dissipative behavior. While there is a plethora of works devoted to learning conservative phenomena, based upon Hamiltonian or Lagrangian descriptions (see, among others, [9] [56] [57] [58] [59] [60]), very little has been investigated for learning dissipative phenomena.

We consider a system whose time evolution is described in terms of a set of state variables that ensure the full description of its thermodynamical state. In general, any physical system can be described at different levels (micro, meso, macro), that incorporate different information. At the molecular dynamics scale, physics is described by the positions and momenta of every molecule. While at this scale Newtonian laws apply and everything is conservative, the number of degrees of freedom, and also the time scale at which the phenomenon evolves, makes this type of description useless. Coarser levels incorporate less information, but also involve fewer degrees of freedom [61]. At this level, the effect of unresolved variables (those of lower levels not considered) on the evolution of resolved ones (those considered in the coarse level) introduces dissipation, by the celebrated fluctuation-dissipation theorem [62].

The present approach models Newtonian, and also non-Newtonian (shear-thinning), fluids at a hydrodynamics level in a laminar regime. At this level, the state variables required for a full description of a (possibly non-Newtonian) fluid are position $\boldsymbol{q}_j$, velocity $\boldsymbol{v}_j$, internal energy E_j, and stress fields $\boldsymbol{\tau}_j$, according to the GENERIC formalism [63]. Thus, for a fluid discretized into M particles, the set $\mathcal{S}$ of state variables is

$$\mathcal{S} = \{\boldsymbol{z} = (\boldsymbol{q}_j, \boldsymbol{v}_j, E_j, \boldsymbol{\tau}_j, j = 1, 2, \ldots, M) \in (\mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R} \times \mathbb{R}^6)^M\}. \quad (1)$$

With the help of an RGBD camera, whose detailed description can be found in Section 6.1, only partial measurements of these variables are observable. We are not interested in using complex laboratory equipment, such as particle imaging velocimetry for instance, which would greatly limit the scope of our approach. Instead, we only have access to a set of points of the fluid's free surface at each video frame, as will be detailed below. This severely limits the amount of information at our disposal and obliges us to develop a system able to recover the dynamical state. In this sense, our methodology is linked to (at least, partially) self-supervised methodologies.

As a proof of concept, we implement the learning algorithm to reason about the physics of different fluids contained in a glass subjected to arbitrary movements. This phenomenon has been first reproduced computationally to simulate the evolution of a discretized fluid under different initial conditions. It has been computationally modeled and simulated with Smooth Particle Hydrodynamics [51]. In those simulations, the geometry is not a parameter of the problem. We perform four simulations over different types of fluids, with four different initial velocities to trigger the slosh, but we consider the same volume of liquid every time. Therefore, the initial particle discretization is always the same, and valid also for the real liquid, since we fill the cup with the same volume of liquid. As a control test, we selected a liquid (glycerine) whose particle variations are not extremely chaotic. Nevertheless, the particles do experience variations in position, especially those of the free surface, which is the main interest in this work, and this evolution is to be captured by the learning algorithm proposed. Then, we evaluate the state of the fluid at discretized time steps for each simulation. The required state variables are evaluated at each particle and stored for each time step. As a result, we have a collection of snapshots that describe the state of the fluid for training of the networks. Afterward, the system must be able to predict the temporal evolution of a liquid under different, previously unseen, conditions with limited information. Each particle has a tag or an index. This indexation is employed to define de information vector in a specific order. Also, we work on a local coordinate system: the positions of the particles are referred to the center of gravity of the cup, placed at the bottom of the volume. Given a continuous discretization for a non-too-chaotic sloshing, we did not consider it necessary to force the autoencoder to be permutation invariant in this specific example. In this work, we mainly focus on the study of the dynamics of different types of fluids to perform accurate and physically sound emulations in real-time, and the distillation of information from only measurements of the free surface.

4 Methodology and architecture

The complexity of the algorithm just presented forces us to implement our system in three different steps, see Fig. 1 for a graphical sketch of the implemented architecture. The highly dimensional nature of the problem motivates the reduction of the dynamics to carry out learning on an embedded space of a much lower dimension. In the case of learning and predicting new situations from real-world data, a correlation needs to be established between the data available (the free surface) and the latent space built from full computational descriptions. We hypothesize with the existence of features (distinctive attributes) in data sequences of the free surface that relates the history of partial measurements with internal variables of the fluid. An architecture based on recurrent neural networks can unveil those correlations, map the data acquired to the latent space, and output the reconstructed state in the next time step.

To accomplish these requirements, we need to develop three different architectures. Firstly, we have to project our computational data to a lower dimensional manifold to train the algorithm efficiently and achieve real-time performance in the implementation. Secondly, we need to train a physics-informed integrator that will learn the evolution of the

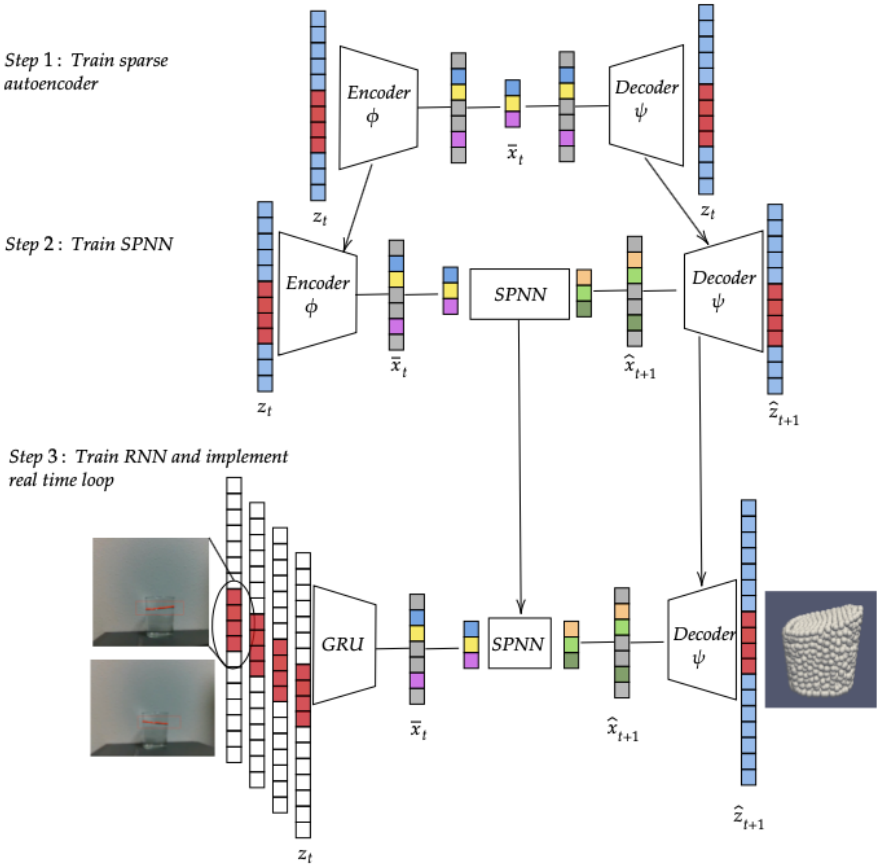

Figure 1: Sketch of the construction procedure for the deep neural network able to perceive sloshing phenomena. In a first step (first row), we must unveil the intrinsic dimensionality of the problem. To that end, we train a sparse autoencoder from computational full-field data. Once the number of variables governing the physics has been determined, we must train a structure-preserving neural network, able to integrate in time the state of the system. Thus, at step 2 in the figure, given the state of the system at time t, the decoder will output the state of the system at time $t + \Delta t$. But the main difficulty is that during runtime we do not have access to the high-dimensional state of the system, only a portion of it, represented in red in the input vector of the network. These values correspond to the position of the free surface of the fluid. Thus, the encoder must be substituted in step 3 by a GRU that takes the near history of the free surface to convert it into the reduced-order encoding of the system that will feed the thermodynamics-informed time integrator.

dynamics in the latent space. Finally, we work over an algorithm based on recurrent neural networks to substitute the encoder with a perceptron capable of projecting partial measurements to the latent space. It is worth mentioning that we chose to train in two different steps the autoencoder and the RNN and apply transfer learning. In transfer learning, we profit from the knowledge acquired in a new application. Sloshing dynamics is already a complex problem to apply model order reduction, and working with limited information from the free surface could complicate the process of learning a low dimensional manifold. Despite these considerations to optimize the algorithm, the need for deep neural networks that learn the patterns of the dynamics remains.

4.1 Model reduction based on autoencoders

In spite of the advances in terms of computational resources and calculation power, as well as the capacity of storage, model order reduction is still a necessary tool to deal with the abundance of data and the physical complexity of certain phenomena. Training a complex database could jeopardize the convergence to an optimal result. Such is the case of the database available to build the model of the present work. In addition, operating in a low-dimensional manifold could also foster the robustness and stability of the proposed learning scheme so that changes in the input do not strongly condition the stability of the solution.

Currently, the development of these techniques is focused on capturing the important features of non-linearities. Fluid dynamics are led by strong non-linear structures hard to be learned by machine learning methods. New approaches arise from the perspective of autoencoders [64] [65] as a preprocessing step to deal with turbulence and instabilities.

Given a set of snapshots that live in a smooth finite-dimensional manifold $\mathcal{M} \in \mathbb{R}^D$, with $D = 13M$, as discussed in Eq. (1), we aim to find a projection to a manifold of latent variables $\mathcal{N} \in \mathbb{R}^d$ of much lower dimensionality, $d \ll D$. Autoencoders are an alternative to perform such reduction by finding patterns in the spatiotemporal structure of data. They are neural networks based on unsupervised learning which consist of two parts: an encoder that maps data to an embedded space, and a decoder that reconstructs latent information to the original space. The output of the decoder $\hat{\boldsymbol{z}}_t = \hat{\boldsymbol{z}}(t)$ has to be equal to the input of the encoder $\boldsymbol{z}_t = \boldsymbol{z}(t)$. These two structures are trained by backpropagating the error of the reconstruction,

$$\begin{aligned} \text{Encoder } \phi : \mathcal{M} \subset \mathbb{R}^D &\to \mathbb{R}^d \\ \boldsymbol{z} &\mapsto \boldsymbol{x}, \end{aligned}$$

$$\begin{aligned} \text{Decoder } \psi : \mathcal{N} \subset \mathbb{R}^d &\to \mathbb{R}^D \\ \boldsymbol{x} &\mapsto \hat{\boldsymbol{z}}. \end{aligned}$$

These schemes have shown good performance unveiling non-linear features to provide low and accurate representations. In this work, we decided to use sparse autoencoders (SAE) so the sparsity in the bottleneck distills the real low dimensionality of the problem. This is accomplished automatically by the sparse autoencoder by employing a L1-norm penalization [66]. This can be interpreted, in the light of scientific machine learning, as a very practical form of imposing parsimony—in other words, Occam's razor—to the learned model [8].

The backpropagated loss has two terms. With N_{snap} as the number of snapshots introduced in the algorithm, the first loss term $\mathcal{L}^{\text{sae}}_{\text{mse}}$ analyses the reconstruction error between the ground truth and the result of the decoder,

$$\mathcal{L}^{\text{sae}}_{\text{mse}} = \frac{1}{N_{\text{snap}}} \sum_{i=1}^{N_{\text{snap}}} (\boldsymbol{z}_i - \hat{\boldsymbol{z}}_i)^2. \tag{2}$$

It is evaluated with the mean squared error (MSE) between the output and the input. Secondly, a regularizer term $\mathcal{L}^{\text{sae}}_{\text{reg}}$ is introduced to enforce the sparsity of the solution for the latent state $\boldsymbol{x}_i$,

$$\mathcal{L}^{\text{sae}}_{\text{reg}} = \sum_{i=1}^{N_d} |\boldsymbol{x}_i|, \tag{3}$$

where N_d is the dimension of the latent vector of the autoencoder. The size of the bottleneck is fixed a priori, and the number of non-vanishing entries (i.e., the intrinsic dimensionality of data) will be determined without user intervention during the training period.

The contribution of the regularization is weighted with a coefficient $\lambda^{\text{sae}}_{\text{reg}}$ to control its influence in the training process,

$$\mathcal{L}^{\text{sae}} = \mathcal{L}^{\text{sae}}_{\text{mse}} + \lambda^{\text{sae}}_{\text{reg}} \mathcal{L}^{\text{sae}}_{\text{reg}}. \tag{4}$$

Although we have normalized the dataset, we decided to embed each group of state variables (position, velocity, internal energy, and stress tensor separated in normal $\boldsymbol{\sigma}$ and shear $\boldsymbol{\tau}$ components) separately to capture all the important features of each group of variables. In other words, we define five different autoencoders. The latent variables coming from each autoencoder are merged to form the latent space of the dynamics. This decision has been taken for optimization and accuracy reasons. The five encoders have the same structure. For all of them, the encoder and decoder have the same architecture, but inverted. They have been defined as fully connected and non-recursive layers that follow a feed-forward scheme.

4.2 Learning the dynamical evolution based on Structure-Preserving Neural Networks

We have already mentioned the fact that one of our primary interests is to develop a technique that satisfies known basic principles of physics. Thus, the learned time integrator should fulfill the first principles of physics to provide credible predictions of future events to help in decision-making. By structure-preserving neural networks (SPNN) we refer to a class of techniques that are constructed to satisfy some a priori known properties of the problem such as equivariance [71] or energy conservation [57] [60]. In their most general form, they can be applied to conservative as well as dissipative problems, in which the principles of thermodynamics are satisfied by construction [39] [38]. In the

case of SPNN, we work from a thermodynamical perspective to drive learning to admissible scenarios that will ensure the consistency of the results. For this purpose, we employ inductive biases that come from thermodynamic priors.

When the phenomenon studied is dissipative, GENERIC is a particularly convenient formalism to describe its evolution in time. It presents a formulation to model the evolution of a vector of variables $\boldsymbol{z}$ from the analysis of the energy (Hamiltonian) potential in conjunction with a second (Massieu) potential that captures the dissipative nature of the dynamics. It is expressed in terms of the Poisson and friction brackets, which can be reformulated as matrix operators, as

$$\frac{d\boldsymbol{z}}{dt} = \boldsymbol{L}(\boldsymbol{z})\frac{\partial E(\boldsymbol{z})}{\partial \boldsymbol{z}} + \boldsymbol{M}(\boldsymbol{z})\frac{\partial S(\boldsymbol{z})}{\partial \boldsymbol{z}}. \tag{5}$$

The product of the gradient of energy and the symplectic matrix, $\boldsymbol{L}\nabla E$, models the conservative part of the time evolution, while the entropy gradient and the friction matrix $\boldsymbol{M}\nabla S$ capture the occurring dissipative effects beyond equilibrium. To discern which part of the evolution is governed by conservative phenomena and which one by dissipation, an additional condition must be imposed, the so-called degeneracy conditions,

$$\boldsymbol{L}\frac{\partial S}{\partial \boldsymbol{z}} = \boldsymbol{0}, \quad \boldsymbol{M}\frac{\partial E}{\partial \boldsymbol{z}} = \boldsymbol{0},$$

that state that energy has nothing to do with dissipation (it is conserved in closed systems) and entropy is not responsible for reversible phenomena.

In fact, these degeneracy conditions guarantee the conservation of energy,

$$\dot{E}(\boldsymbol{z}) = \nabla E(\boldsymbol{z}) \cdot \dot{\boldsymbol{z}} = \nabla E(\boldsymbol{z}) \cdot \boldsymbol{L}(\boldsymbol{z})\nabla E(\boldsymbol{z}) + \nabla E(\boldsymbol{z}) \cdot \boldsymbol{M}\nabla S(\boldsymbol{z}) = 0, \tag{6}$$

and the production of entropy,

$$\dot{S}(\boldsymbol{z}) = \nabla S(\boldsymbol{z}) \cdot \dot{\boldsymbol{z}} = \nabla S(\boldsymbol{z}) \cdot \boldsymbol{L}(\boldsymbol{z})\nabla E(\boldsymbol{z}) + \nabla S(\boldsymbol{z}) \cdot \boldsymbol{M}\nabla S(\boldsymbol{z}) \geq 0, \tag{7}$$

respectively.

These requirements are guaranteed if we choose $\boldsymbol{L}$ and $\boldsymbol{M}$ to be skew-symmetric and symmetric, positive semidefinite, respectively. For the sake of clarity, it is worth mentioning that GENERIC preserves these properties in the full-order as well as in the reduced-order manifolds, enabling the learning of its structure in the latent space. In other words, it works equally well for $\boldsymbol{z}$ and $\boldsymbol{x}$, provided that $\boldsymbol{x}$ expresses the dynamics with accuracy.

We work on a discretized context of data samples, and learning is performed over a discrete expression of GENERIC. In this case, we consider a Forward Euler approximation of the time derivative that describes the dynamical evolution of the system. From this expression, the discrete degeneracy conditions to be imposed can be straightly derived:

$$\frac{\boldsymbol{x}_{n+1} - \boldsymbol{x}_n}{\Delta t} = \mathsf{L}(\boldsymbol{x}_{n+1})\mathsf{DE}(\boldsymbol{x}_{n+1}) + \mathsf{M}(\boldsymbol{x}_{n+1})\mathsf{DS}(\boldsymbol{x}_{n+1}), \tag{8}$$

where L, M, DE and DS represent the discretized versions of $\boldsymbol{L}$, $\boldsymbol{M}$, ∇E and ∇S, respectively, and the subscript n refers to time $t = n\Delta t$ and $n+1$ indicates time $t + \Delta t$, respectively.

The SPNN follows a feed-forward flow which consists of a set of fully-connected layers that learn the gradients of energy and entropy as well as the matrices L and M. The learning scheme enforces the skew-symmetry and symmetry of L and M respectively and the degeneracy conditions. Given pairs of consecutive snapshots, the neural network learns the integrator of the dynamical problem. The input is the state vector of latent variables at time t and $t + \Delta t$, $\boldsymbol{x}_n$ and $\boldsymbol{x}_{n+1}$. The output coming from the net is a vector that contains the predicted L, M, and gradients of energy and entropy of the current state. During runtime, the time integration is consecutively performed with these operators.

The loss that carries the information to train and guide the SPNN is composed by two different terms. Firstly, the final output coming from the integration must match the ground truth. The accuracy is evaluated by measuring the mean squared error of these quantities,

$$\mathcal{L}_{\text{mse}}^{\text{SPNN}} = \frac{1}{N_{\text{snap}}} \sum_{i=1}^{N_{\text{snap}}} (x_i - \hat{x}_i)^2. \tag{9}$$

Secondly, training is also governed by the degeneracy conditions. They are taken into account as a loss coming from the sum of the squared values of both conditions,

$$\mathcal{L}_{\text{deg}}^{\text{SPNN}} = \frac{1}{N_{\text{snap}}} \sum_{i=1}^{N_{\text{snap}}} (\mathsf{L}_i\mathsf{DS}_i)^2 + (\mathsf{M}_i\mathsf{DE}_i)^2. \tag{10}$$

Finally, the MSE loss is weighted with the hyperparameter $\lambda_{\text{mse}}^{\text{SPNN}}$ to control its influence in the global loss function of the network,

$$\mathcal{L}^{\text{SPNN}} = \lambda_{\text{mse}}^{\text{SPNN}}\mathcal{L}_{\text{mse}}^{\text{SPNN}} + \mathcal{L}_{\text{deg}}^{\text{SPNN}}. \tag{11}$$

4.3 Recurrent neural networks for state reconstruction

The latent space is built from computational data obtained from simulations, for which we have a full description of the fluid states. Computational data includes internal variables important in our description, whereas they are unmeasurable by ordinary means from camera inputs. We hypothesize that, although internal variables are not measurable, their influence can be unveiled from the dynamical evolution of the free surface. From this information, we could analyze the features of the history of the dynamics where these internal variables will arise.

Following the spirit of self-supervised learning, in the sense of unveiling not-given information, we propose an approach based on recurrent neural networks (RNNs) to reconstruct the state of the fluid. RNNs are structures that take into consideration the history of data working over sequences instead of learning from discrete and individual snapshots. RNNs are widely used, especially in the fields of natural language processing, speech recognition, or economics. Vanilla RNN often encounters, however, vanishing and exploding gradients [67]. Gated Recurrent Units (GRU) [68] and Long Short-Term Memory (LSTM) units [69] are architectures capable of dealing with these problems. They include *gates*, or flows of information, to keep important information in long or complex sequences while forgetting irrelevant features.

The basic idea behind the GRU architecture is to accumulate information from previous layers, see Fig. 2. The hidden state $\boldsymbol{h}_t$ represents a summary of the features identified in previous sequences. $\boldsymbol{g}_t^{\text{update}}$ is the output of the update gate. This gate selects which information from the hidden state and the input sequence passes to the next step, modeled with a sigmoid activation function. In contrast, the reset gate $\boldsymbol{r}_t$ reflects the past information that should be avoided. A new memory cell $\boldsymbol{n}_t$, defined as the reset information, stores only the relevant information from the past. The output is the final hidden state $\boldsymbol{h}_t$ that accumulates the relevant information of past states and features learned from the current input sequence:

$$
\begin{aligned}
\boldsymbol{g}_t^{\text{update}} &= \sigma(\boldsymbol{x}_t U^z + \boldsymbol{h}_{t-1} W^z), \\
\boldsymbol{r}_t &= \sigma(\boldsymbol{x}_t U^r + \boldsymbol{h}_{t-1} W^r), \\
\boldsymbol{n}_t &= \tanh(\boldsymbol{x}_t U^h + (\boldsymbol{r}_t \boldsymbol{h}_{t-1}) W^h), \\
\boldsymbol{h}_t &= (1 - \boldsymbol{g}_t^{\text{update}}) \boldsymbol{h}_{t-1} + \boldsymbol{g}_t^{\text{update}} \boldsymbol{n}_t.
\end{aligned}
$$

GRUs have two main gates: an update gate, to update the hidden state, and a reset gate, which evaluates whether the previous cell state is relevant. In contrast with LSTMs, GRUs do not have a forget gate, which controls what is considered important to be remembered or to be rendered futile, and have fewer parameters. Despite having a simpler structure, GRU's performance is similar to LSTM in certain tasks. GRUs have proven to train faster and more efficiently with smaller datasets and shorter sequences [70].

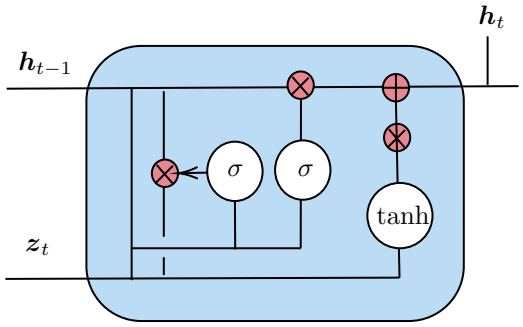

Figure 2: Representation of a GRU cell. The three main paths indicated represent the update and reset gates, the new memory cell, and their connection to update the new hidden state transmitted to the next later.

The input of the network consists of a sequence of measurements of the position of the free surface of the fluid. Since it is firstly trained with computational data, we select the particles of the discretization that belong to the free surface at each time step. The batch of sequences is introduced in the network to pass through GRU recurrent layers doing a projection *from-many-to-one*, i.e., introducing a sequence to obtain a single vector as output. The output vector of the GRU layers passes through a final forward fully connected layer with linear activation. The result of this process $\hat{\boldsymbol{x}}_t$ must match the latent state vector corresponding to the last snapshot $\boldsymbol{x}_t$ of the given sequence. The loss $\mathcal{L}_{\text{mse}}^{\text{GRU}}$ evaluates

the MSE between the predicted latent state and the ground truth,

$$\mathcal{L}_{\text{mse}}^{\text{GRU}} = \frac{1}{N_{\text{snap}}} \sum_{i=1}^{N_{\text{snap}}} (\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i)^2. \tag{12}$$

5 Training and validation

We train the algorithm from computational simulations performed in Abaqus CAE (Dassault Systèmes). It includes an add-on to perform conversion into particles and apply SPH in the simulations [72]. Each of the performed simulations is 2 seconds long, the time at which the fluid reaches the equilibrium state. We evaluate the state of the fluid at discrete time steps, equally spaced by time increments of 0.005 seconds. As a result, we have 1600 snapshots available for training. This dataset is split into two subsets: 80% for training and 20% for testing. The same training subsets are employed for training the three networks. The particles are labeled, and the data is stored for each time step into a vector following the order of the labeling. The geometry is not a parameter of the problem, and we have a constant discretization of the fluid. Nevertheless, the method is aimed to be able to learn the patterns of the dynamics, especially of those of the free surface, to emulate appropriately the sloshing perceived.

Once the three nets have been trained, we assemble the algorithm to feed the simulation loop with only a sequence of limited data. Instead of providing all the state variables obtained from the simulations, we pretend to only have the position at some points of the free surface. The sequence will be projected to the latent space, the SPNN will learn the dynamics, and the decoder will output results of the simulation augmenting the information originally given. The decoder provides the whole reconstruction of the fluid as well as the velocities, stresses and internal energy.

5.1 Hyperparameters and characteristics of each net

Each snapshot consists of a state vector of the position, velocity, internal energy, and stresses (shear and normal) evaluated at each particle of the discretized fluid. The fluid is discretized into 2134 particles. Thus, the global dimensionality is 27742.

The global SAE is subdivided into five different SAE, one for each subset of state variables. The nets have been initialized following the Kaiming method in which, briefly speaking, the weight initialization follows a Gaussian, and biases are set to 0 [73]. In addition, we apply linear activations in the first and last layers and ReLU for hidden layers. The optimizer chosen for these nets is Adam. A scheduler is programmed for updating the learning rate after 1000 and 3000 epochs.

We adapt the architectures according to the input dimension and the complexity or noisy nature of the values, and encoder and decoder have a symmetric structure:

- Position: Input size is $D = 6402$ and output size $d = 20$. It is composed by $N_h = 2$ hidden layers of size 120.
- Velocity: Given the complexity of the velocity, we built a net of input size $D = 6402$, output size $d = 20$, $N_h = 4$ hidden layers, and hidden size 200.
- Internal energy: In the case of energy, input size is $D = 2134$, output size $d = 10$, and there are $N_h = 3$ hidden layers which consist of 40 neurons each.
- Normal stress: The normal stress tensor components are identical. Thus, the input shape of the net is $D = 2134$, the output shape is $d = 20$, and it is composed of $N_h = 3$ hidden layers of 200 neurons.
- Shear stress: This net had input size $D = 6402$, $N_h = 3$ hidden layers of 200 neurons, and output size $d = 20$.

The specific learning parameters such as learning rate (lr), weight decay (wd) and sparse weights (λ^{sae}) are defined for each SAE. Table 1 shows the parameters of each net. Given the complexity of the patterns learned, we require low learning rates.

After 10000 epochs, SAEs converge to optimal results. Taking into account the sparsity imposed to improve the reduction, the final dimension of each net is $d_{\text{position}} = 3$, $d_{\text{velocity}} = 3$, $d_{\text{energy}} = 2$, $d_\sigma = 3$ and $d_\tau = 2$. Thus, the final shape of the reduced space is $d_{\text{latent space}} = 13$. The latents obtained are the output of the RNN, and the input of the SPNN. This substantial reduction will not only reduce computing time and storage, but also improve the convergence to a solution in subsequent trainings.

The recurrent neural network relates partial measurements of the fluid to the latent space that we have built. We consider that the only information accessible by ordinary means in real-time is information related to the free surface of the fluid. Specifically, we pretend we can only measure the position at some points of the free surface.

Table 1: Training parameters for each SAE.

	lr	wd	λ^{sae}
Position (q)	10^{-4}	10^{-6}	10^{-3}
Velocity(v)	10^{-4}	10^{-5}	10^{-3}
Internal energy (e)	10^{-4}	10^{-5}	10^{-4}
Normal stress (σ)	10^{-4}	10^{-5}	5×10^{-3}
Shear stress (τ)	10^{-3}	10^{-6}	5×10^{-3}

We take each full-order snapshot of the database to find the particles that represent the free surface of the liquid evaluating their height over their neighbors. Those points do not necessarily follow a balanced distribution since they are not uniformly distributed on the free surface. Given the density of the surface particle set, a simple linear interpolation is performed to obtain a uniform free surface grid. This step facilitates comparison between sequences in future predictions. As a result, we have equally discretized free surfaces of 21 points, which results in a vector of size 42. Despite reconstructing a 3D representation of fluids, we only consider a 2D data projection in the recurrent autoencoder to be consistent with the information that will come from the camera. Although we reconstruct the depth map of the images, depth measurements are still noisy. Thus, we decided to rely on 2D data, perpendicular to the depth, which represents the vertical movement of the liquid, that is more stable. Those coordinates represent the direction of the movement of the glass, and the vertical height of the liquid due to the sloshing effect.

Once we have the information related to the free surface at each time step, we assemble the sequences. We consider sequences of 16 snapshots. It is the minimum sequence size to guarantee that the features of the dynamics are correctly captured to find a mapping to the low dimensional manifold. With smaller sequences, the RNN does not learn good projections to the latent space. Since it is a short, although complex, sequence, GRU is optimal for this case.

Even though the time step of the data in our database is 0.005 seconds, the camera streams depth measurements at a frequency of 60 Hz. Therefore, to assemble the sequences, we must choose snapshots equally spaced by approximately 0.015 seconds.

The input size of the net is: batch size $\times$ sequence length $\times$ vector size. The net consists of three GRU hidden layers of 26 neurons, and there is a last feed-forward fully connected layer to connect the last GRU layer to the latent space of size 13. This last layer has linear activation. The optimizer for training this net is Adam, and parameters are set to $lr = 10^{-3}$, $wd = 10^{-5}$. The learning rate is updated by a scheduler at 1000 and 3000 epochs.

We reach good results after 10000 epochs. Training loss is 1.19×10^{-3}, and test loss reaches a value of 2.3×10^{-3}.

The SPNN learns the integration scheme of the dynamics in the low manifold. The latent variables are the input of the net, so the input size is 13. Providing that L and M are skew symmetric and symmetric, respectively, we only learn the upper elements of the main diagonal. Thus, instead of learning the full matrices of dimension $d \times d$, we learn $d \cdot (d-1)/2$ elements for L and $d \cdot (d+1)/2$ elements for M. Therefore, considering that the gradients have size d, the final output size is $d_{out} = d \cdot (d-1)/2 + d \cdot (d+1)/2 + d + d = 195$.

We have reduced the complexity of the dynamics thanks to applying model order reduction. Despite it, the dynamics and the latent evolution are still complicated for training. We require a structure of $N_h = 13$ hidden layers of size 195.

The SPNN has been also initialized following the Kaiming method. We have applied ReLU activations for hidden layers, and linear activations in the first and last layers. The initial parameters are set to $lr = 10^{-3}$, $wd = 10^{-5}$. The optimizer selected for training is also Adam, and the scheduler updates the learning rate after 1500, 2400, and 4000 epochs. Lower learning rates were required, again, due to the complexity of the features to be learned. The weight assigned to the MSE loss is $\lambda_{\text{mse}}^{spnn} = 10^3$, to give more importance to the reconstruction.

We train the SPNN for 5000 epochs. At that point, training and test losses are $3.2 \times 10{-}3$ and $1.42 \times 10{-}2$, respectively.

5.2 Initial validation

The performance of the method is tested with an input sequence extracted from the simulation of initial velocity 0.2 m/s. Given this sequence, the information is mapped to the latent state. Once we obtain the latent variables corresponding to that sequence, the SPNN integrates the dynamics until the fluid reaches the steady-state. We could provide information of sequences at each time step, but data acquisition is not always accessible in real scenarios (occlusion, connection problems, ...). In those cases, the simulation should continue until new information is provided. To this end, we test the

Table 2: Loss comparison among SAE, kPCA and POD

	Error AE	**Error POD**	**Error kPCA**
$\boldsymbol{q}$	0.149×10^{-4}	0.257×10^{-4}	0.141×10^{-4}
$\boldsymbol{v}$	4.1×10^{-4}	19×10^{-4}	6.25×10^{-4}
e	0.472×10^{-4}	0.64×10^{-4}	0.342×10^{-4}
$\boldsymbol{\sigma}$	5.1×10^{-4}	20×10^{-4}	3.371×10^{-4}
$\boldsymbol{\tau}$	0.798×10^{-4}	19×10^{-4}	3.36×10^{-4}

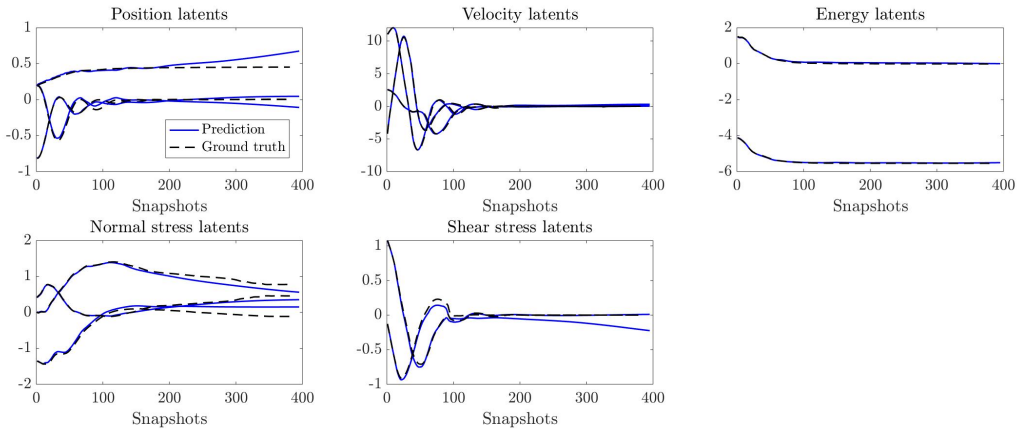

Figure 3: Simulation results. Learning of the dynamics in the latent manifold. Dashed lines represent the time evolution of the latents that aimed to be emulated. Lines in blue represent the result of the SPNN in the latent manifold.

ability of the method to continue integrating, and the stability of the simulation and results, by providing only one first snapshot.

Table 2 shows the MSE of the autoencoder proposed to reconstruct each group of state variables. These results have been compared with those obtained with POD [74] taking 10 modes, and kPCA [75] with 4 modes. Modes are selected concerning the evolution of the eigenvalues obtained from each method. The AE achieves the same or improved levels of accuracy as POD and kPCA. Figure 3 plots the simulation results in the reduced-order space. The initial state has been projected to the latent manifold to emulate the evolution of its behavior. Fig. 5 plots the time evolution of some state variables in the high-dimensional space for 21 randomly selected particles. Finally, Fig. 4 shows the comparison between the ground truth and the projection of the results to the high order manifold at three steps. This figure also includes the RMSE error of the reconstruction of each snapshot and the Hausdorff distance between the ground truth and the result. The Hausdorff distance (HD) [76] evaluates the closeness of two sets by analyzing the largest distance between one set of points to another. If the HD is low, it resembles a high degree of similarity

$$HD = \max\{\sup_{x \in X} d(x, Y), \sup_{y \in Y} d(y, X)\},$$

being X and Y the two sets to be compared. In this context, the HD evaluates how well the shape of the result matches the shape of ground truth to have an indicator of the accuracy in the reconstruction. Thus, it compares the maximum (sup) distance from the ground truth to the output $\sup_{x \in X} d(x, Y)$, and vice versa $\sup_{y \in Y} d(y, X)$. Of these two distances, the maximum is the HD of the mismatch. After analyzing the results obtained in the computational phase, we decide to test the loop in a real scenario for the reconstruction of real fluids.

6 Tests with real-world data

Once the proposed strategy has been tested on computational data, it is extended to real-world problems. An RGBD camera is used to track the free surface of the fluid. This information is converted into sequences to predict the next state of the fluid. Finally, we not only have a reconstruction of the fluid, but also an augmented representation of it, by outputting the velocity, internal energy, and stress fields.

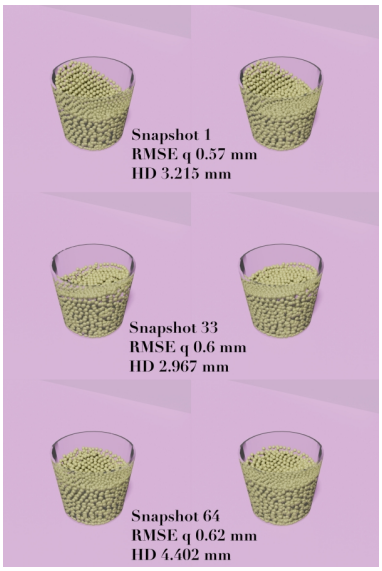

Figure 4: Comparison of the reconstruction of the integration provided by the SPNN (right) with the ground truth (left). The selected snapshots correspond to peaks of the sloshing dynamics of glycerine. Specifically, we present the comparison for snapshots 1, 33, and 64 of the collection. The height of the cup is $7cm$, and it is filled up to $5.6cm$ approximately.

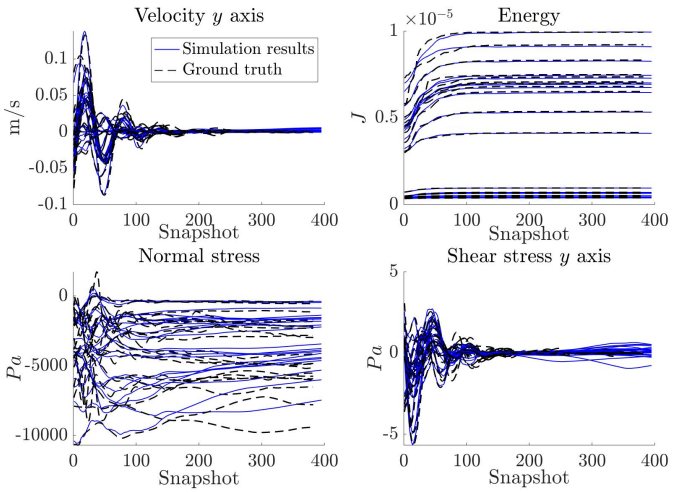

Figure 5: Time evolution of selected state variables evaluated at 21 random particles. The graph shows a comparison between the simulated fields with the ground truth for the validation simulation of the algorithm.

6.1 Data acquisition review

Firstly, we need to detect the free surface of the fluid and acquire the 3D coordinates to feed the algorithm with real data. We make use of a stereo camera for this purpose, although many of the presented results could equally be obtained with a standard camera. The model of our stereo camera is RealSense D415 (`https://www.intelrealsense.com/depth-camera-d415/`). This model of stereo camera provides both intrinsic and extrinsic parameters as well as depth measurements. Thus, the projection of the 2D image into the 3D world, and vice versa, is straightforward. The free surface is expressed in pixel coordinates u, v. The software provides the intrinsic parameters f_x, f_y, c_x, c_y, s, which are the focal length, the optical center coordinates, and the skew coefficient respectively. In addition to the intrinsic parameters K, we can also estimate (using Simultaneous Localization and Mapping techniques [77]), the rotation $\boldsymbol{R}$

and translation $\boldsymbol{t}$ components to complete the projection to the real world coordinates X, Y, Z of the point $\boldsymbol{p}_w$,

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix},$$

$$\tilde{\boldsymbol{x}}_s = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]\boldsymbol{p}_w.$$

A sketch of the camera system is depicted in Fig. 6.

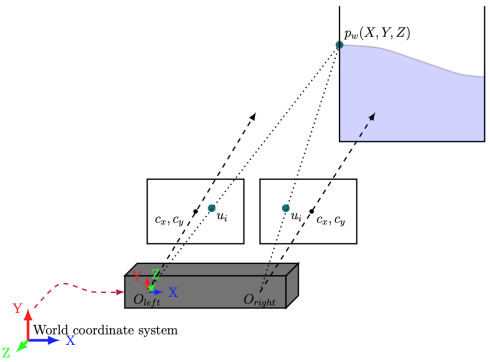

Figure 6: Representation of the data acquisition system. The free surface is the tracked element of the algorithm to perform learning and the simulation of the dynamics.

It is worth mentioning that the original frame does not provide good enough depth measurements. It reports holes where the algorithm did not successfully measure the depth of the pixels. Measurements related to transparent objects are often invalid or noisy since their surfaces are not Lambertian, which is the main assumption of the measurement algorithm incorporated in the stereo camera. In other words, instead of reflecting light evenly in all directions, they also refract light, resulting in unmeasurable conditions for the technique defined. Our approach consists in applying some filters to enhance depth streaming. Firstly, we apply a decimation filter to reduce the complexity of the measurements to foster stability. Then, the frame is mapped to a disparity map where the spatial filter, to preserve the edges, and the temporal filter, to promote data persistency, are applied. This result is projected back to the depth map where the hole-filling filter is finally applied. The filtered depth map outputs a full depth field from which we can evaluate the position of the features of the glass and the free surface (see Fig. 7). This procedure is fully detailed in the reference [78].

We binarize the color frame, also streamed by the camera, to convert the image into a black and white picture. The free surface appears as a gradient in the black and white image, see Fig. 8. Since we prioritize the speed in the data streaming, we define an area for performing this analysis instead of forcing the recognition across the full image. The points of the free surface are detected, tracked, projected from frame coordinates to 3D, and stored.

6.2 Reconstruction and integration from video streaming

We assemble the data obtained in the data acquisition step into sequences. These sequences feed the algorithm trained with computational data. The RNN projects the sequences to the latent space, the SPNN integrates the dynamics, and the decoder outputs the next state of the free surface and the position of the whole set of particles. Velocity, internal energy, σ, and τ evaluated at each particle are also provided. Therefore, we reconstruct the complete state of the fluid at the next time step only from the free surface. The video stream for validation consists of 800 frames, which is a recording of 12 seconds.

Figure 10 shows the results of the algorithm compared to the real video streaming. We perform the reconstruction and integration over the whole sequence, i.e. no cuts were applied to the streaming and the method is applied continuously. We apply the three steps (RNN projection, integration, and decoding) over the full video in 3.42 seconds on an ordinary laptop (Macbook Pro 2013-3 GHz Intel Core i7), achieving (much more than) the real-time performance proposed. Some snapshots of the sloshing were selected and plotted in the first row of Fig. 6.2. The snapshots shown represent the peaks, which are the most critical states in manipulation, and some intermediate states between the peaks. The rest of the pictures correspond to the augmented information obtained with this method, which has been possible thanks to the physics-aware simulation framework.

Figure 7: Color and depth stream before (up) and after (down) applying the required filters to reconstruct the depth map of the streaming.

Figure 8: Representation of the color frame and its conversion to a binarized image to seek the free surface. The area defined for searching is represented in the color frame as well as the points of the free surface detected in the black and white image.

All results of the integration are stable, realistic, and close to the real result. We analyze objectively the results by evaluating the root mean squared error (RMSE) between the real $\boldsymbol{y}$ and the predicted $\hat{\boldsymbol{y}}$ free surfaces in n snapshots of the video streaming. Ultimately, we feed the algorithm with the free surface in t (from the video), and we compare the integration result $(t+1)$ with the free surface in $t+1$ (from the video),

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2}.$$

The evolution of the error along the video is represented in Fig. 11. The error remains under 5 mm in the whole sequence of the length of the video and stays lower than 3 mm in the vast majority of it. We also evaluate the HD between the free surface that comes from the camera and the simulation. These results reflect the closeness between the free surfaces, for which there is not a larger deviation than 4–5 mm, even in the higher peaks of the sloshing. In some cases, higher deviations in the HD come from distortions in the detection of the free surface (like in the first snapshot of water presented).

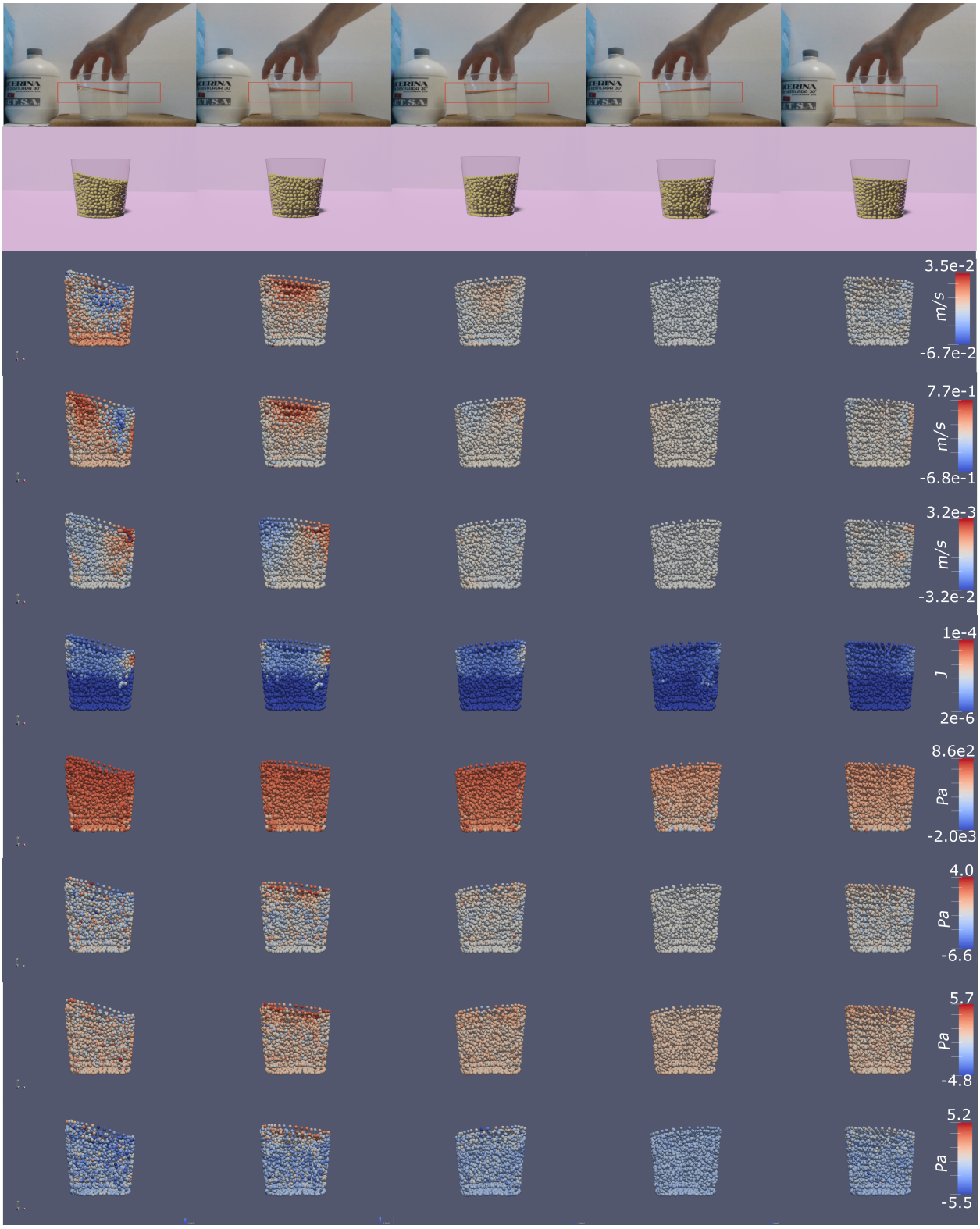

Figure 9: Results for a 12 seconds video of a glass of glycerine. Eight snapshots of the sloshing sequence were selected for comparison. The selected snapshots have index 560, 565, 568, 572, 578 from left to right. The second row corresponds to the fluid reconstruction and prediction provided the previous snapshot. From row three to ten we show the additional information obtained from the reconstruction and simulation (velocity, energy and stress fields, respectively).

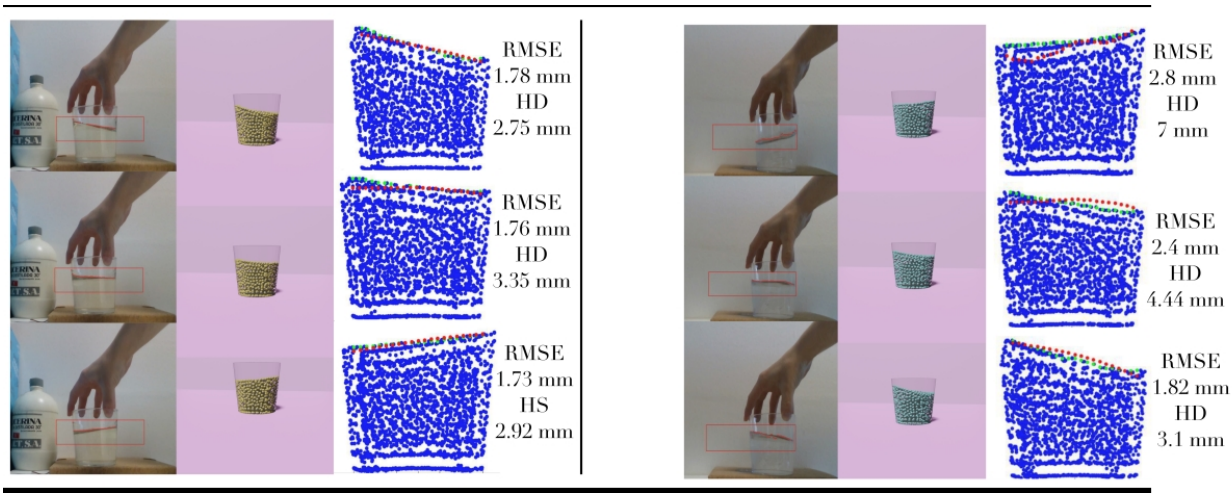

Figure 10: Detail of the comparison of glycerine (left) and water (right) with the prediction. The third column of both liquids compares the predicted fluid volume (in blue), the free surface of the liquid volume (green) and the target free surface (in red). The RMSE and the Hausdorff distance (HD) that correspond to each snapshot are indicated.

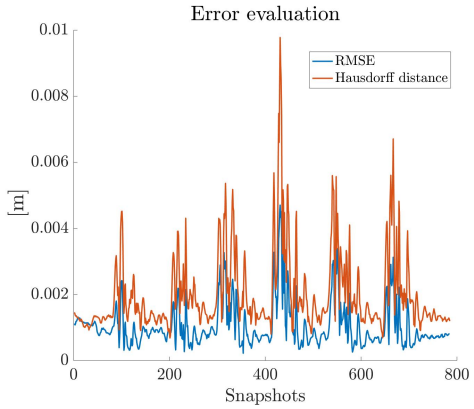

Figure 11: Evolution of the mean squared error during the perception process of sloshing in a glass of glycerine.

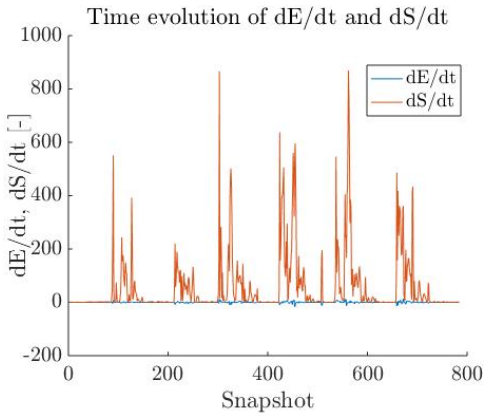

Figure 12: Time derivatives of energy and Entropy along the video. The time derivative of Energy oscillates around zero, ensuring energy conservation. Entropy production is also ensured since the time derivative is always positive.

Fig. 12 showcases finally the compliance of the principles of thermodynamics in the predictions. The time derivative of energy makes little oscillations due to the numerical approximation around 0, which means that we ensure the conservation of energy. In addition, the time derivative of entropy remains always positive, fulfilling its production.

7 Conclusions and future work

We have presented an approach for the physical perception of sloshing phenomena. It is based on physics-informed learned simulators connected to the real world employing commodity RGBD cameras. The algorithm has been trained with computational data to build a physically sound reduced-order manifold to learn the evolution of the dynamics dictated by the General Equation for the Non-Equilibrium Reversible-Irreversible Coupling (GENERIC). This thermodynamic framework ensures the physical consistency, accuracy, and realism of the results to promote informed decision-making.

Provided the physics-aware approach for learning, only four simulations per fluid were needed to perform training that accurately mimics computational and real behaviors. This approach has been completed with the development of a self-supervised technique to recover information of the dynamics that is unmeasurable by ordinary means. Advanced sensors and tools, such as PIV cameras, are available to evaluate data that cannot be extracted from a simple video stream. Nevertheless, we are still unable to measure important information for a full physical description. The suggested methodology fills in the gaps of information for the simulation of future dynamical states.

Since the main purpose of the approach is to connect real-world systems with AI-guided simulators, we test the implementation of the integration scheme coupled with a data-acquisition system. The free surface is tracked by an RGBD camera, and the information obtained is used for fluid prediction. Notably, real-time is successfully achieved due to the reduction obtained through the application of autoencoders: 12 seconds of real-world time are analyzed in slightly more than 3 seconds, allowing performing decision making or using control algorithms. In addition, information is provided to the user by using augmented reality, i.e., by outputting the reconstructed fluid volume and a set of variables that may be important for decision-making on top of the video stream.

Nonetheless, physics perception must achieve generality. It is unmanageable to train a model for each casuistic. Consequently, transfer learning must be extended to world and scene reasoning. Starting from a model such as the one proposed, incremental learning could set the base to extend learning and build hybrid twins that learn from evolving scenarios. Finally, the permutation-invariant condition is a desirable characteristic to work with unordered meshes [79] [80] [81]. The consideration of this condition would help to achieve a higher degree of adaptivity and complexity previewed in future works.

Acknowledgments

The work presented has been partially supported by the Spanish Ministry of Economy and Competitiveness through Grant number PID2020-113463RB-C31 and by the Regional Government of Aragon and the European Social Fund, research group T88. The authors also thank the support of ESI Group through the project UZ-2019-0060.

References

[1] C. K. Liu and D. Negrut, "The role of physics-based simulators in robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, 2020.

[2] C. J. Bates, I. Yildirim, J. B. Tenenbaum, and P. Battaglia, "Modeling human intuitions about liquid flow with particle-based simulation," *PLoS computational biology*, vol. 15, no. 7, p. e1007210, 2019.

[3] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18 327–18 332, 2013.

[4] C. Schenck and D. Fox, "Spnets: Differentiable fluid dynamics for deep neural networks," in *Conference on Robot Learning*. PMLR, 2018, pp. 317–335.

[5] L. Li, S. Hoyer, R. Pederson, R. Sun, E. D. Cubuk, P. Riley, K. Burke *et al.*, "Kohn-sham equations as regularizer: Building prior knowledge into machine-learned physics," *Physical review letters*, vol. 126, no. 3, p. 036401, 2021.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations," *arXiv preprint arXiv:1711.10561*, 2017.

[7] L. Yang, X. Meng, and G. E. Karniadakis, "B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data," *Journal of Computational Physics*, vol. 425, p. 109913, 2021.

[8] Z. Liu and M. Tegmark, "AI Poincaré: Machine Learning Conservation Laws from Trajectories," *arXiv preprint arXiv:2011.04698*, 2020.

[9] K. L. Course, T. W. Evans, and P. B. Nair, "Weak form generalized hamiltonian learning," *arXiv preprint arXiv:2104.05096*, 2021.

[10] Z. Bai, S. L. Brunton, B. W. Brunton, J. N. Kutz, E. Kaiser, A. Spohn, and B. R. Noack, "Data-driven methods in fluid dynamics: Sparse classification from experimental data," in *Whither Turbulence and Big Data in the 21st Century?* Springer, 2017, pp. 323–342.

[11] P. Rodríguez-Ocampo, M. Ring, J. Hemandez-Fontes, J. Alcérreca-Huerta, E. Mendoza, G. Gallegos-Diez-Barroso, and R. Silva, "A 2d image-based approach for cfd validation of liquid mixing in a free-surface condition." *Journal of Applied Fluid Mechanics*, vol. 13, no. 5, 2020.

[12] K. Bieker, S. Peitz, S. L. Brunton, J. N. Kutz, and M. Dellnitz, "Deep model predictive flow control with limited sensor data and online learning," *Theoretical and Computational Fluid Dynamics*, pp. 1–15, 2020.

[13] A. Sancarlos, M. Cameron, A. Abel, E. Cueto, J.-L. Duval, and F. Chinesta, "From rom of electrochemistry to ai-based battery digital and hybrid twin," *Archives of Computational Methods in Engineering*, vol. 28, no. 3, pp. 979–1015, 2021.

[14] B. Moya, I. Alfaro, D. Gonzalez, F. Chinesta, and E. Cueto, "Physically sound, self-learning digital twins for sloshing fluids," *PLoS One*, vol. 15, no. 6, p. e0234569, 2020.

[15] M. Grmela and H. C. Öttinger, "Dynamics and thermodynamics of complex fluids. i. development of a general formalism," *Physical Review E*, vol. 56, no. 6, p. 6620, 1997.

[16] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, "Relational inductive biases, deep learning, and graph networks," *arXiv preprint arXiv:1806.01261*, 2018.

[17] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2146–2153.

[18] M. Nava, A. Paolillo, J. Guzzi, L. M. Gambardella, and A. Giusti, "Uncertainty-aware self-supervised learning of spatial perception tasks," *arXiv preprint arXiv:2103.12007*, 2021.

[19] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *IEEE robotics and automation letters*, vol. 5, no. 2, pp. 2372–2379, 2020.

[20] C. Rao, H. Sun, and Y. Liu, "Physics-informed deep learning for computational elastodynamics without labeled data," *Journal of Engineering Mechanics*, vol. 147, no. 8, p. 04021043, 2021.

[21] J. L. Callaham, K. Maeda, and S. L. Brunton, "Robust flow reconstruction from limited measurements via sparse representation," *Physical Review Fluids*, vol. 4, no. 10, p. 103907, 2019.

[22] L. Sun and J.-X. Wang, "Physics-constrained bayesian neural network for fluid flow reconstruction with sparse and noisy data," *Theoretical and Applied Mechanics Letters*, vol. 10, no. 3, pp. 161–169, 2020.

[23] N. B. Erichson, L. Mathelin, Z. Yao, S. L. Brunton, M. W. Mahoney, and J. N. Kutz, "Shallow neural networks for fluid flow reconstruction with limited sensors," *Proceedings of the Royal Society A*, vol. 476, no. 2238, p. 20200097, 2020.

[24] K. O. Lye, S. Mishra, and D. Ray, "Deep learning observables in computational fluid dynamics," *Journal of Computational Physics*, vol. 410, p. 109339, 2020.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] A. K. Maier, C. Syben, B. Stimpel, T. Würfl, M. Hoffmann, F. Schebesch, W. Fu, L. Mill, L. Kling, and S. Christiansen, "Learning with known operators reduces maximum error bounds," *Nature Machine Intelligence*, vol. 1, no. 8, pp. 373–380, 2019. [Online]. Available: `https://doi.org/10.1038/s42256-019-0077-5`

[27] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature communications*, vol. 9, no. 1, pp. 1–10, 2018.

[28] J. Ayensa-Jiménez, M. H. Doweidar, J. A. Sanz-Herrera, and M. Doblaré, "Prediction and identification of physical systems by means of physically-guided neural networks with meaningful internal layers," *Computer Methods in Applied Mechanics and Engineering*, vol. 381, p. 113816, 2021.

[29] P. Jin, Z. Zhang, I. G. Kevrekidis, and G. E. Karniadakis, "Learning poisson systems and trajectories of autonomous systems via poisson neural networks," *arXiv preprint arXiv:2012.03133*, 2020.

[30] J. S. Hesthaven, C. Pagliantini, and N. Ripamonti, "Rank-adaptive structure-preserving reduced basis methods for hamiltonian systems," *arXiv preprint arXiv:2007.13153*, 2020.

[31] F. Masi, I. Stefanou, P. Vannucci, and V. Maffi-Berthier, "Material modeling via thermodynamics-based artificial neural networks," in *Workshop on Joint Structures and Common Foundations of Statistical Physics, Information Geometry and Inference for Learning*. Springer, 2020, pp. 308–329.

[32] H. Yu, X. Tian, Q. Li *et al.*, "Onsagernet: Learning stable and interpretable dynamics using a generalized onsager principle," *arXiv preprint arXiv:2009.02327*, 2020.

[33] R. Ibañez, D. Borzacchiello, J. V. Aguado, E. Abisset-Chavanne, E. Cueto, P. Ladeveze, and F. Chinesta, "Data-driven non-linear elasticity: constitutive manifold construction and problem discretization," *Computational Mechanics*, vol. 60, no. 5, pp. 813–826, 2017.

[34] D. González, F. Chinesta, and E. Cueto, "Thermodynamically consistent data-driven computational mechanics," *Continuum Mechanics and Thermodynamics*, vol. 31, no. 1, pp. 239–253, 2019.

[35] ——, "Learning non-markovian physics from data," *Journal of Computational Physics*, vol. 428, p. 109982, 2021.

[36] C. Ghnatios, I. Alfaro, D. González, F. Chinesta, and E. Cueto, "Data-driven generic modeling of poroviscoelastic materials," *Entropy*, vol. 21, no. 12, p. 1165, 2019.

[37] B. Moya, D. González, I. Alfaro, F. Chinesta, and E. Cueto, "Learning slosh dynamics by means of data," *Computational Mechanics*, vol. 64, no. 2, pp. 511–523, 2019.

[38] Q. Hernandez, A. Badías, D. González, F. Chinesta, and E. Cueto, "Deep learning of thermodynamics-aware reduced-order models from data," *Computer Methods in Applied Mechanics and Engineering*, vol. 379, p. 113763, 2021.

[39] Q. Hernandez, A. Badias, D. González, F. Chinesta, and E. Cueto, "Structure-preserving neural networks," *Journal of Computational Physics*, vol. 426, p. 109950, 2021.

[40] B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler, "Deep fluids: A generative network for parameterized fluid simulations," in *Computer Graphics Forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 59–70.

[41] J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin, "Accelerating eulerian fluid simulation with convolutional networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3424–3433.

[42] T. P. Miyanawala and R. K. Jaiman, "An efficient deep learning technique for the navier-stokes equations: Application to unsteady wake flow dynamics," *arXiv preprint arXiv:1710.09099*, 2017.

[43] S. R. Bukka, R. Gupta, A. R. Magee, and R. K. Jaiman, "Assessment of unsteady flow predictions using hybrid deep learning based reduced-order models," *Physics of Fluids*, vol. 33, no. 1, p. 013601, 2021.

[44] S. Wiewel, M. Becher, and N. Thuerey, "Latent space physics: Towards learning the temporal evolution of fluid flow," in *Computer graphics forum*, vol. 38, no. 2. Wiley Online Library, 2019, pp. 71–82.

[45] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia, "Learning to simulate complex physics with graph networks," in *International Conference on Machine Learning*. PMLR, 2020, pp. 8459–8468.

[46] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 2020.

[47] H. Gao, L. Sun, and J.-X. Wang, "Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain," *Journal of Computational Physics*, vol. 428, p. 110079, 2021.

[48] Y. Li, T. Lin, K. Yi, D. Bear, D. Yamins, J. Wu, J. Tenenbaum, and A. Torralba, "Visual grounding of learned physical models," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5927–5936.

[49] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum, "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning," *Advances in neural information processing systems*, vol. 28, pp. 127–135, 2015.

[50] C. Schenck and D. Fox, "Perceiving and reasoning about liquids using fully convolutional networks," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 452–471, 2018.

[51] J. J. Monaghan, "Smoothed particle hydrodynamics," *Annual review of astronomy and astrophysics*, vol. 30, no. 1, pp. 543–574, 1992.

[52] S. J. Koppal, *Lambertian Reflectance*. Boston, MA: Springer US, 2014, pp. 441–443. [Online]. Available: `https://doi.org/10.1007/978-0-387-31439-6_534`

[53] C. Schenck and D. Fox, "Detection and tracking of liquids with fully convolutional networks," *arXiv preprint arXiv:1606.06266*, 2016.

[54] C. Do, T. Schubert, and W. Burgard, "A probabilistic approach to liquid level detection in cups using an rgb-d camera," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2075–2080.

[55] S. Eppel, "Tracing liquid level and material boundaries in transparent vessels using the graph cut computer vision approach," *arXiv preprint arXiv:1602.00177*, 2016.

[56] T. Bertalan, F. Dietrich, I. Mezić, and I. G. Kevrekidis, "On learning hamiltonian systems from data," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, no. 12, p. 121107, 2019.

[57] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 15 379–15 389.

[58] P. Jin, A. Zhu, G. E. Karniadakis, and Y. Tang, "Symplectic networks: Intrinsic structure-preserving networks for identifying hamiltonian systems," *arXiv preprint arXiv:2001.03750*, 2020.

[59] P. Toth, D. J. Rezende, A. Jaegle, S. Racanière, A. Botev, and I. Higgins, "Hamiltonian generative networks," *arXiv preprint arXiv:1909.13789*, 2019.

[60] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ode-net: Learning hamiltonian dynamics with control," *arXiv preprint arXiv:1909.12077*, 2019.

[61] P. Español, *Statistical Mechanics of Coarse-Graining*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 69–115. [Online]. Available: `http://dx.doi.org/10.1007/978-3-540-39895-0_3`

[62] R. Kubo, "The fluctuation-dissipation theorem," *Reports on progress in physics*, vol. 29, no. 1, p. 255, 1966.

[63] P. Espanol, M. Serrano, and H. C. Öttinger, "Thermodynamically admissible form for discrete hydrodynamics," *Physical review letters*, vol. 83, no. 22, p. 4542, 1999.

[64] K. Taira, M. S. Hemati, S. L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S. T. Dawson, and C.-A. Yeh, "Modal analysis of fluid flows: Applications and outlook," *AIAA journal*, vol. 58, no. 3, pp. 998–1022, 2020.

[65] N. B. Erichson, M. Muehlebach, and M. W. Mahoney, "Physics-informed autoencoders for lyapunov-stable fluid flow prediction," *arXiv preprint arXiv:1905.10866*, 2019.

[66] A. Ng *et al.*, "Sparse autoencoder," *CS294A Lecture notes*, vol. 72, no. 2011, pp. 1–19, 2011.

[67] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*. PMLR, 2013, pp. 1310–1318.

[68] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[69] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[70] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[71] E. Celledoni, M. J. Ehrhardt, C. Etmann, R. I. McLachlan, B. Owren, C.-B. Schönlieb, and F. Sherry, "Structure preserving deep learning," *arXiv preprint arXiv:2006.03364*, 2020.

[72] M. Smith, ABAQUS/Standard User's Manual, Version 6.9. United States: Dassault Systèmes Simulia Corp, 2009.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[74] H. V. Ly and H. T. Tran, "Modeling and control of physical processes using proper orthogonal decomposition," *Mathematical and computer modelling*, vol. 33, no. 1-3, pp. 223–236, 2001.

[75] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural computation*, vol. 10, no. 5, pp. 1299–1319, 1998.

[76] D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[77] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[78] D. T. Anders Grunnet-Jepsen, "Depth post-processing for intel® realsense™ depth camera d400 series."

[79] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[80] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[81] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.