Learning Kinematic Formulas from Multiple View Videos

Liangchen Song[1*], Sheng Liu[1*], Celong Liu[2], Zhong Li[2], Yuqi Ding[3], Yi Xu[2], and Junsong Yuan[1]
[1]University at Buffalo [2]OPPO US Research Center, InnoPeak Technology Inc. [3]Louisiana State University
{lsong8,sliu66,jsyuan}@buffalo.edu

ABSTRACT

Given a set of multiple view videos, which records the motion trajectory of an object, we propose to find out the objects' kinematic formulas with neural rendering techniques. For example, if the input multiple view videos record the free fall motion of an object with different initial speed v, the network aims to learn its kinematics: $\Delta = vt - \frac{1}{2}gt^2$, where Δ, g and t are displacement, gravitational acceleration and time. To achieve this goal, we design a novel framework consisting of a motion network and a differentiable renderer. For the differentiable renderer, we employ Neural Radiance Field (NeRF) since the geometry is implicitly modeled by querying coordinates in the space. The motion network is composed of a series of blending functions and linear weights, enabling us to analytically derive the kinematic formulas after training. The proposed framework is trained end to end and only requires knowledge of cameras' intrinsic and extrinsic parameters. To validate the proposed framework, we design three experiments to demonstrate its effectiveness and extensibility. The first experiment is the video of free fall and the framework can be easily combined with the principle of parsimony, resulting in the correct free fall kinematics. The second experiment is on the large angle pendulum which does not have analytical kinematics. We use the differential equation controlling pendulum dynamics as a physical prior in the framework and demonstrate that the convergence speed becomes much faster. Finally, we study the explosion animation and demonstrate that our framework can well handle such black-box-generated motions.

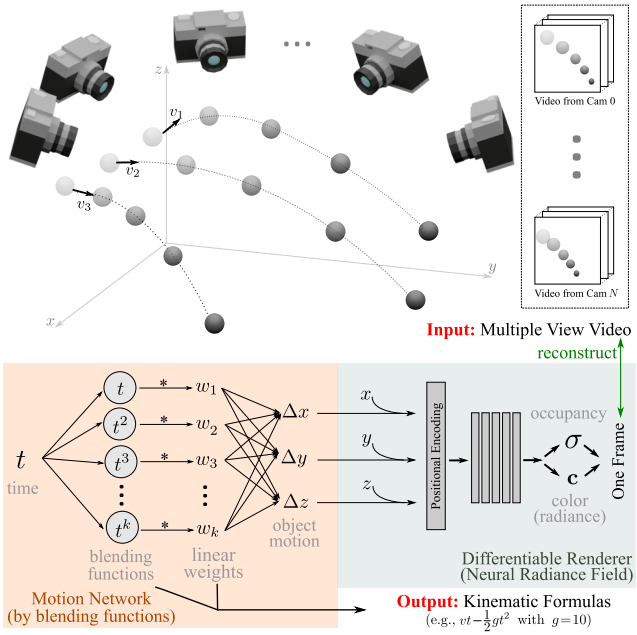

Figure 1: Illustration of the problem studied in the paper. When the training is finished, the kinematic formulas can directly derived from the motion network.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; *Physical simulation.*

KEYWORDS

Physics informed deep learning; differentiable rendering; multi-view reconstruction

ACM Reference Format:
Liangchen Song, Sheng Liu, Celong Liu, Zhong Li, Yuqu Ding, Yi Xu, and Junsong Yuan. 2021. Learning Kinematic Formulas from Multiple View Videos . In *Proceedings of the 29th ACM International Conference on Multimedia (MM '21), October 20–24, 2021, Virtual Event, China.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3474085.3479235

[*]Both authors contributed equally to this research.

MM '21, October 20–24, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8651-7/21/10…$15.00
https://doi.org/10.1145/3474085.3479235

1 INTRODUCTION

In 1971, along with the third crewed mission to land on the Moon, a video[1] is recorded to demonstrate free fall: Astronaut David Scott released a hammer and a feather simultaneously. After watching the video, we humans are convinced by Galileo's discovery that free falling objects fall with the same acceleration. As artificial intelligence has advanced greatly in recent years, a question naturally arises: Can machines get wise to the fact from their observations as well?

Learning or inferring physical status from vision data has been widely studied over the past decades, such as fluid simulation and capture [3], Particle Imaging Velocimetry [1, 9] and Particle Tracking Velocimetry [12, 25]. Previous works are mainly concerned with *complex* physical phenomenons with *a lot of* knowledge about the imaging system. Instead, we explore the possibility of finding out *simple* physical patterns with *little* knowledge about the imaging system. Specifically, we focus on the kinematics of rigid body in this

[1]https://moon.nasa.gov/resources/331/the-apollo-15-hammer-feather-drop/

work, which is *simple*, while for the imaging system we only assume camera intrinsic and extrinsic parameters as available knowledge, which is *little*.

Though our scope is limited to simple kinematics, the learning task is by no means a cakewalk. The first challenge is correctly building the physical model of the rigid body to be studied. To study the kinematics, we should be aware of the motion of the object thus a temporal-consistent reconstruction of the object is needed. The second challenge is the gap between physical object positions and the observed video frames. That is, even if the physical model of the object is known, we still need a non-trivial render-then-compare step to judge whether the learned physical status is accurate.

To address the above challenges, we propose a framework consisting of a differentiable rendering module and blending function (also known as basis functions) based motion modeling network. Specifically, we adopt the recently developed Neural Radiance Field (NeRF) [13] for learning the physical model of the object. In NeRF, the object geometry is represented implicitly by querying a multi-layer perceptron (MLP), thus the challenge of temporal consistency can be addressed by simply freezing the MLP network.

Moreover, for the second challenge, since the object is modeled by MLP, we construct a motion network and connect it to NeRF. The motion network consists of blending functions and linear weights for combining them. In this way, the overall rendering flow remains differentiable, so the motion network can be trained in an end-to-end manner by comparing the rendered frames with observed, which bridges the aforementioned gap effectively.

A general overview of the problem and our framework is demonstrated in Fig. 1. We design three experiments with synthetic data to validate the effectiveness and extensibility of our proposed framework. First, we formulate the problem as learning kinematic formulas of an object moving with only the force of gravity. The inputs are multiple view videos of an object moving with different initial speed v. We set the blending functions as the monomial basis and introduce the sparsity penalty on the linear weights, which follows the law of parsimony (Occam's razor). Then we demonstrate that the motion network can successfully discover the kinematic formula, i.e., $vt - \frac{1}{2}gt^2$. Next, we investigate if the motion network can leverage physical priors by studying the kinematics of large amplitude pendulum. The differential equation, $\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0$, which represents the motion of a simple pendulum is used as a penalty term, to ensure that the learned kinematic follows the physical laws. Finally, we validate the practical value of the learned kinematic formulas by considering an animation of explosion. To sum up, our contributions are as follows:

- We propose a novel framework for learning kinematic formulas from multiple view videos, for which we only assume the knowledge of camera intrinsic and extrinsic parameters.
- We design several experiments to validate the effectiveness and extensibility of our framework. We demonstrate that our framework can be easily combined with various prior knowledge and readily applicable to animation tasks.

1.1 Related Works

1.1.1 Physics-informed deep learning. Data-driven machine learning schemes can be equipped with prior knowledge from physics [7], which makes the model more robust and explainable. In [17–19], Raissi et al. consider two tasks: data-driven solution and data-driven discovery of partial differential equations (PDEs). The authors successfully solve nonlinear PDEs including continuous time models like the Burgers' equation, Shrödinger equation, and discrete time models like the Allen-Cahn equation. Moreover, in [17] the authors demonstrate that their framework is able to discover nonlinear PDEs like Navier-Stokes equations and Korteweg–de Vries equation. In [6], the authors use neural networks to exploits conservation laws to make predictions and gain conceptual insights such as Copernicus' conclusion that the solar system is heliocentric. All of the above existing works directly use physical status like locations as inputs to the network, while we emphasize the exploitation of vision data. More related work is [20], in which Raissi et al. use fluid visualization images as a reference rather than real captured images considered in our paper.

1.1.2 Physics from vision. In [23], Stewart et al. use physics priors to train a deep network for extracting representations from sensor inputs like images. Methods like partial image velocimetry and partial tracking velocimetry are proposed to measure the velocity field from imaging data [1, 12]. For example, Li et al. use a compact lenslet-based light field camera to reconstruct the 3D fluid flow by tracking dense particles floating in the fluid. Most of the existing works focus on inferring complex physical values like velocity and pressure fields of fluid from carefully controlled environments. As a comparison, we are interested in learning simple kinematics from common multiple video inputs.

1.1.3 Scene reconstruction with neural radiance fields. A scene can be effectively reconstructed with neural rendering techniques [24]. In this paper, we adopt the renderer neural radiance fields (NeRF) proposed by Mildenhall et al. [13]. In NeRF, differentiable volume rendering is employed to connect fields with images. Since NeRF represents the scene implicitly and renders a novel view is time-consuming, Neural Sparse Voxel Fields is proposed by Liu et al. [11] to speed up querying the fields. NeRF has also been successfully extended to modeling dynamic scenes. In [14, 16], deformations are considered for reconstructing a non-rigidly deforming scene. Also, in [2], Gafni et al. propose dynamic NeRF for facial avatar reconstruction. Further, space-time constraints are considered in [10, 27], therefore generating high-quality videos for novel views. Our method can be viewed as an explicit version of the above dynamic NeRF since we model the motion explicitly with blending functions.

2 PRELIMINARIES

We employ an implicit scene representation, Neural Radiance Fields (NeRF), as the differentiable renderer in our framework. NeRF is a volume rendering framework proposed in [13]. In NeRF, the scene is represented by a neural field F_Θ, which is a multi-layer perceptron that takes 3D point locations $\mathbf{p} = (x, y, z)$ and viewing direction $\mathbf{d} = (d_x, d_y, d_z)$ as input. For the field, we have $(\mathbf{c}, \sigma) = F_\Theta(\mathbf{p}, \mathbf{d})$, where $\mathbf{c} = (r, g, b)$ is the color and σ is density. Final image colors are rendered from the field F_Θ through differentiable volume rendering: Denote a camera ray emitted from the center $\mathbf{o}$ with direction $\mathbf{d}$ as $\mathbf{p} = \mathbf{r}(u) = \mathbf{o} + u\mathbf{d}$, then the expected color $C(\mathbf{r})$ with near and far

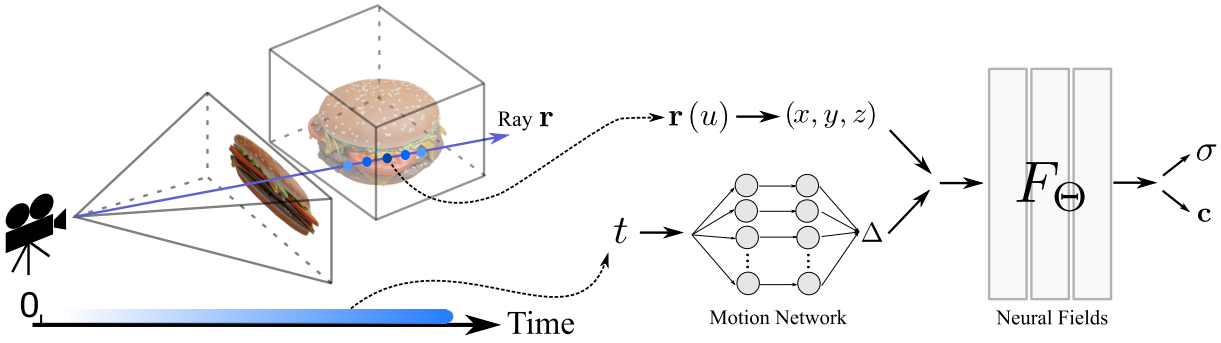

Figure 2: Illustration of computing the color of a ray r. During training, a batch of samples (3D points) are randomly selected on each ray and motion for the points are computed with the motion network.

bounds u_n, u_f is

$$C(\mathbf{r}) = \int_{u_n}^{u_f} e^{-\int_{u_n}^{u} \sigma(\mathbf{r}(v))dv} \sigma(\mathbf{r}(u))\mathbf{c}(\mathbf{r}(u), \mathbf{d})du. \tag{1}$$

In our framework, since we only care about the motion, input view direction $\mathbf{d}$ is not used as an input. The integrations in the above equation are numerically estimated. First, the interval $[u_n, u_f]$ is partitioned into K bins and then one sample is randomly drawn from each bin for summation. Specifically, in each bin a sample is uniformly drawn and for ith bin we have

$$u_i \sim \mathcal{U}\left[u_n + \frac{i-1}{N}(u_f - u_n), u_n + \frac{i}{N}(u_f - u_n)\right]. \tag{2}$$

Once we have the samples $\{u_k\}_{k=1}^{K}$, the integration is computed as

$$\hat{C}(\mathbf{r}) = \sum_{k=1}^{K} e^{-\sum_{k'=1}^{k-1} \sigma_{k'}\delta_{k'}} (1 - e^{-\sigma_k \delta_k})\mathbf{c}_k, \tag{3}$$

where $\sigma_k = \sigma(\mathbf{r}(u_k))$, $\mathbf{c}_k = \mathbf{c}(\mathbf{r}(u_k))$ and $\delta_k = u_{k+1} - u_k$ is the distance between the two samples. Positional encoding is employed to map the coordinate inputs to a higher dimension. The encoding function is defined as

$$\gamma(p) = (\sin(2^0\pi p), \cos(2^0\pi p), \cdots, \sin(2^{L-1}\pi p), \cos(2^{L-1}\pi p)). \tag{4}$$

The function is applied to each input coordinates (i.e., x, y, z) and L is set to 10. Finally, the field is trained with the reconstruction loss between the rendered and true pixel colors:

$$L_{\text{rec}} = \sum_{\mathbf{r}} \|\hat{C}(\mathbf{r}) - C_{\text{gt}}(\mathbf{r})\|_2^2, \tag{5}$$

where C_{gt} is the ground truth RGB colors for ray $\mathbf{r}$.

3 PROPOSED FRAMEWORK

Eqn. (5) is the learning of static scenes using NeRF ??. However in our problem, the modeling of dynamic scene is required. The key idea in our framework is to represent the motion with a set of weighted blending functions of time. After the training process is finished, the whole motion network can be explicitly written as a kinematic formula, composed by the weighted summation of blending functions such as $\{t, t^2, ..., t^n\}$.

3.1 Motion Network

As introduced before, for each point $\mathbf{r}(u)$ input on the ray $\mathbf{r}$, the neural field outputs a color $\mathbf{c}$ and an occupancy score σ. For each frame, the motion of a point is computed from a set of blending functions. We denote the motion network as $M(t) : \mathbb{R} \rightarrow \mathbb{R}^3$, where t means the time. Then the new point becomes $\mathbf{r}(u) + M(t)$ and the σ and $\mathbf{c}$ in Eqn. (3) becomes

$$\sigma_k = \sigma(\mathbf{r}(u_k) + M(t)), \ \mathbf{c}_k = \mathbf{c}(\mathbf{r}(u_k) + M(t)). \tag{6}$$

After finishing the forward process of all the samples on the ray $\mathbf{r}$, the generation of the final color of ray $\mathbf{r}$ is computed with Eqn. (3). The whole forward path is demonstrated in Fig. 2.

In addition to the color reconstruction, we add a constraint for the accumulated occupancy along a ray based on silhouette image. The reason for the extra constraint is that the geometry is not unique with only color reconstruction: For the background area, space can be either empty or filled with points of the background color. Thus the constraint is to avoid such unnecessary points which make the motion network hard to train. Specifically, the accumulated occupancy along a ray is computed by

$$\hat{O}(\mathbf{r}) = \sum_{k=1}^{K} e^{-\sum_{k'=1}^{k-1} \sigma_{k'}\delta_{k'}} (1 - e^{-\sigma_k \delta_k}). \tag{7}$$

From the equation, we have $\hat{O}(\mathbf{r}) \in [0, 1]$, where $\hat{O}(\mathbf{r}) = 0$ means the ray $\mathbf{r}$ hits nothing. Therefore, for empty space, not only the color of $\mathbf{r}$ should be consist with the background, but also the accumulated occupancy $\hat{O}(\mathbf{r})$ should be zero. Since binary silhouette images are easy to acquire, we improve the reconstruction loss L_{rec} in Eqn. (5) to make sure the background space is empty:

$$L^*_{\text{rec}} = \sum_{\mathbf{r},t} \|\hat{C}(\mathbf{r}, t) - C_{\text{gt}}(\mathbf{r}, t)\|_2^2 + \alpha \sum_{\mathbf{r},t} \|\hat{O}(\mathbf{r}, t) - O_{\text{gt}}(\mathbf{r}, t)\|_2^2, \tag{8}$$

where α is set to 0.1 for all experiments and $O_{\text{gt}}(\mathbf{r})$ is the binary value on the silhouette image for the ray.

3.2 Regularization for motion network

Motion network consists of a linear combination of blending functions. Compared to deep networks with nonlinear activations like ReLU, the proposed motion network is easier to be interpreted since we can turn the network into an analytical function, i.e., kinematic formulas. A major benefit brought by the interpretability is

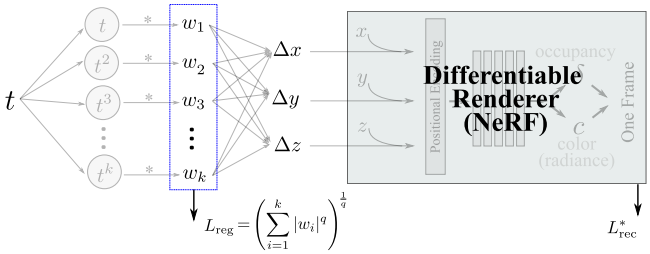

Figure 3: Adopting sparsity regularization in our framework. With a proper ℓ_p norm ($0 \leq p \leq 1$), the framework learns a concise kinematic formula from multiple view inputs.

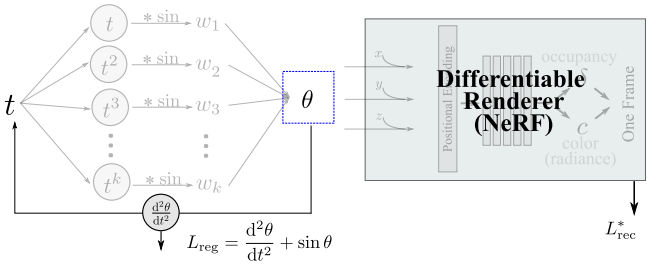

Figure 4: Physical priors, which are in the form of differential equations, can be employed in the framework as a regularization term. The equation of motion $\frac{d^2\theta}{dt^2} + \sin\theta = 0$ for the simple pendulum is used as a regularization.

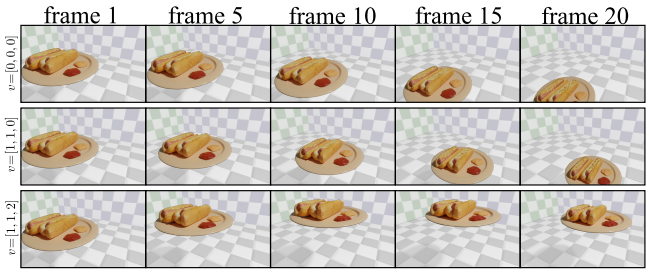

Figure 5: Example training frames for learning the kinematics of free fall. To demonstrate the generality of our method, the hotdog object with complex geometry and textures is selected. Note that the planes with checkerboard pattern is not used in experiments and added only for visualization.

that we can infuse prior knowledge into the network by simply adding a regularization term $L_{\text{reg}}(M)$. Therefore, the final loss is the combination of the reconstruction loss and the regularization term, i.e.,

$$L = L^*_{\text{rec}} + \beta L_{\text{reg}}(M), \tag{9}$$

where β is a balancing parameter.

In the following parts of the section, we introduce several examples of extending our framework with different regularization terms, therefore learning different kinematic formulas of interest.

3.2.1 Case 1: Sparsity Regularization. The principle of parsimony is the principle that "entities should not be multiplied without necessity". For representing the motion of an object, we usually have many solutions for fitting the observed moving frames. That is, in our framework, we can have different sets of blending weights ($\{w_i\}$) when minimizing L_{rec}. Therefore, from all possible solutions, we follow the principle of parsimony and aim to pick out the simplest kinematic formula.

Since the blending weights $\{w_i\}$ directly control the combination of the functions, enforcing sparsity on the weight vector $\mathbf{w} = [w_1, \cdots, w_k]$ will leads to simple kinematic formulas. An illustration is demonstrated in Fig. 3. The sparsity constraint is added to the system by adopting ℓ_p norm ($0 \leq p \leq 1$) to the blending weights. ℓ_p norm for vector $\mathbf{w}$ is defined as

$$\|\mathbf{w}\|_p = \left(\sum_{i=1}^{k} |w_i|^q\right)^{\frac{1}{q}}. \tag{10}$$

The regularization with ℓ_p norm will introduce sparsity to the final $\{w_i\}$ solution [26].

3.2.2 Case 2: Regularization from Differential Equations. Compared to the motion caused solely by gravity, it is more common that there does not exist a closed-form expression for the physical dynamics, such as thermodynamics controlled by heat equation, molecular dynamics controlled by Poisson-Boltzmann equation. To simulate the dynamics, numerical methods and approximation functions are usually used. A successful example is the usage of Fourier series in spectral methods [22].

In our framework, we can simply set the blending functions to the basis used in spectral methods. A benefit brought by our framework is that known physical priors, which are in the form of differential equations, can be easily employed as a regularization term in the loss function.

An illustration of employing physical priors for a pendulum is presented in Fig. 4. It is known that the motion equation of a simple pendulum is

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0, \tag{11}$$

where g is the gravitational acceleration, l is the length of the pendulum and θ is the angle from the vertical to the pendulum. Without loss of generality, we set $\frac{g}{l} = 1$ in the following text. For small oscillations, when θ is small, the $\sin\theta$ term can be approximated by θ and then there is a closed form solution for Eqn. (11). However, if θ gets large, e.g., $\theta = \frac{\pi}{2}$, there is no analytical solution for $\theta(t)$.

During training, for each frame t, θ is generated from the motion network. We then calculate the gradient with respect to t and new computation graphs are created for minimizing the regularization term. In this way, the motion network is regularized to follow the physical prior, i.e., differential equation.

4 EXPERIMENTS

We conduct three experiments with synthetic data for validating our proposed framework. The first experiment is to validate the sparsity regularization proposed in Section 3.2. We set three different initial speed for the model of a plate of hotdog, which has complex geometry and textures. We calculate the location per frame based on the well-known kinematics of free fall: $vt - \frac{1}{2}gt^2$. The

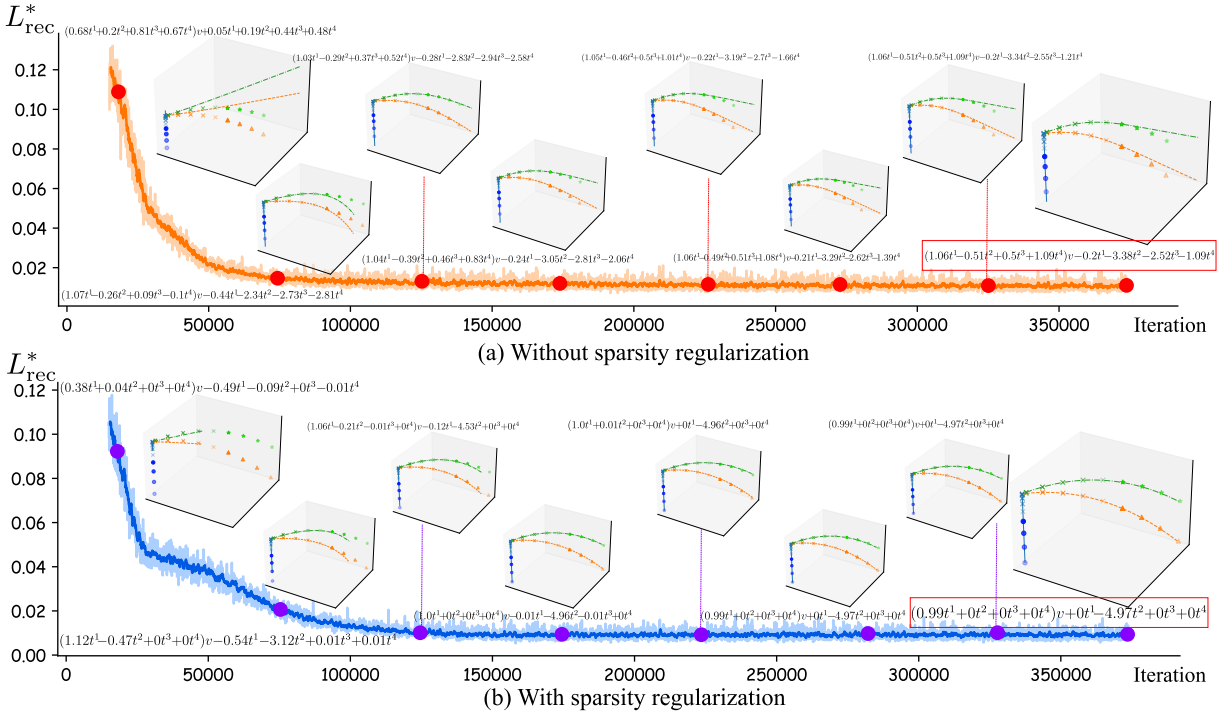

Figure 6: The learned kinematic formula for free fall with and without sparsity regularization. Along the curve of reconstruction loss, we demonstrate the learned kinematic formulas every 50000 iterations. Also, we present the trajectory of the object with the learned formulas. In the trajectory visualization, the three different colors represent three different initial speeds. Points marked by $\times$ are the observed points for training, while all other points are ground-truth future locations and plotted for evaluating the extrapolation of the learned kinematic formulas. The formulas in the red box are the final output formulas.

second experiment corresponds to the regularization from the differential equation. The synthetic data is a pendulum with the object being a burger. The initial angle is set to $\frac{\pi}{2}$ so theoretically there is no closed form kinematic formula. We calculate the θ at each frame numerically with ODEPACK [4]. Finally, we run experiments with the explode effect in Blender, which can be viewed as a black box. For the animation, 100 frames are generated and we uniformly extract 14 frames, which are as the supervision signal. The motion is associated with both translation and rotation, thus more challenging than the previous two experiments. We study the ability of interpolation and extrapolation of our motion network. Moreover, we demonstrate that the motion network generates a smooth trajectory with simple dynamics prior.

For all the experiments, we set 24 cameras uniformly distributed on the upper hemisphere of the scene. Adam optimizer [8] is used for training the network and the initial learning rate is set to 4e-5 for all experiments. For each training iteration, we select 5×2^{10} camera rays (pixels) from one image and the random ray sampling strategy makes sure 90% rays will hit the objects (i.e., pixels inside the silhouette mask). Moreover, we first train the whole network for 10000 iterations only on the first frame. Then we freeze the neural field F_Θ and train the motion network on all frames. The above strategy is based on the idea that we should first reconstruct the geometry and then focus on the motion of the object.

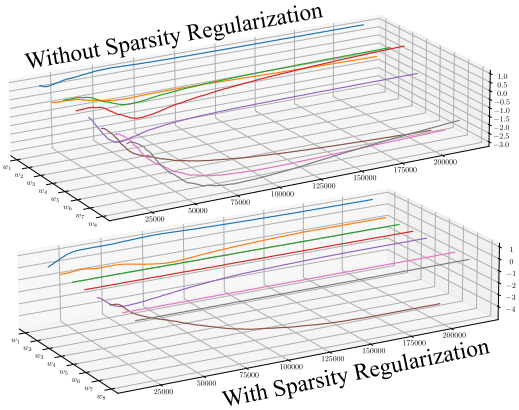

Figure 7: The change of w_i during training for with and without sparsity regularization. Adding extra regularization terms has little impact on the convergence speed.

4.1 Kinematics of Free Fall

We set the initial speed v to three values $\{[0, 0, 0], [1, 1, 0], [1, 1, 2]\}$ and then generate three video sequence with different speed. We define gravity as the negative z direction. Example frames are demonstrated in Fig. 5. To better visualize the motion, we add planes with

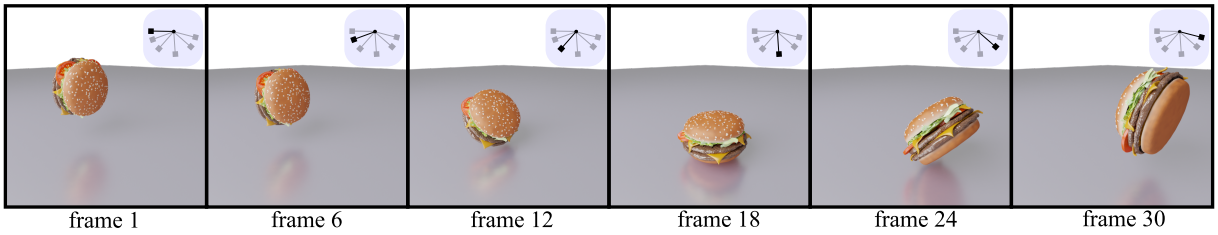

Figure 8: Example frames of the large angle pendulum experiment.

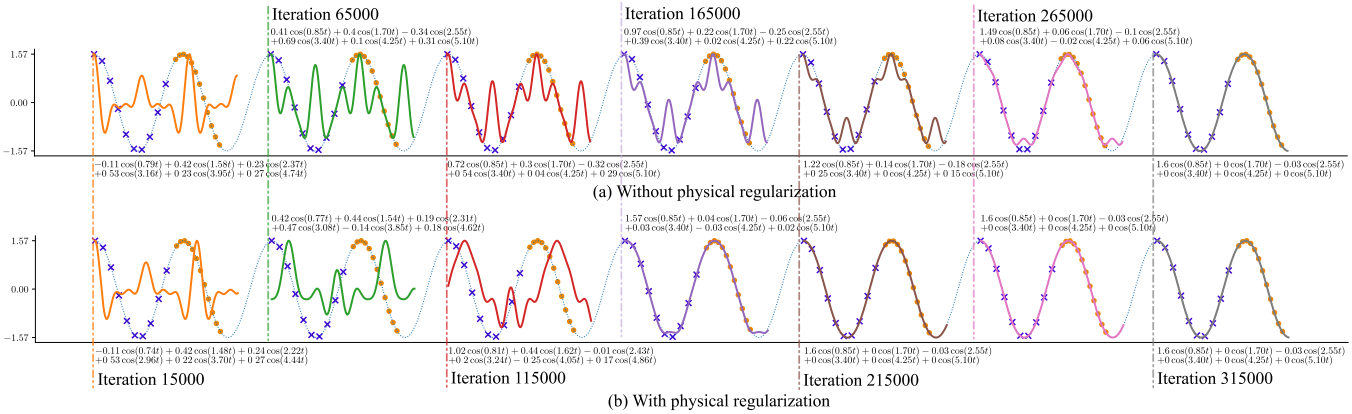

Figure 9: The learned kinematic formula for large angle pendulum with and without physical regularization. Dotted background curves are the numerical results generated with ODEPACK, which can be viewed as ground truth. Points marked by $\times$ are the points for supervision and solid dot points are used to validate the extrapolation loss of the learned kinematics.

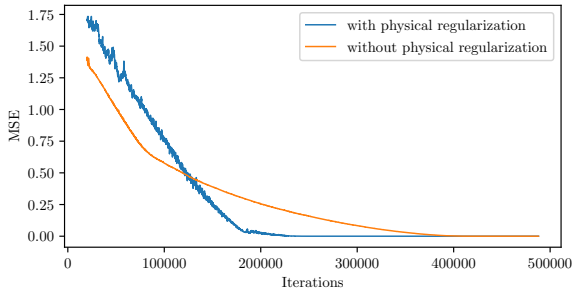

Figure 10: MSE between the future θ values predicted with ODEPACK and predicted with the learned formulas. The error is computed on the solid points demonstrated in Fig. 9.

a checkerboard pattern as the background and they are not used for real training.

To train the motion network, we set the blending functions to be $\{t, t^2, t^3, t^4\}$. In this experiment, there are eight blending weights and the motion network can be written as

$$M(v,t) = v \cdot \sum_{i=1}^{4} w_i t^i + \sum_{i=1}^{4} w_{i+4} t^i. \tag{12}$$

It is obvious that the correct solution should be $w_1 = 1, w_6 = -5 (g = 10)$ and all other $w_i = 0$. However, as shown in Fig. 5, the time range is short in the experiments, there can be many solutions if no regularization is used. In this experiment, we set $p = \frac{1}{10}$ for regularizing a sparse solution and β is set to 1e-3.

A comparison of the learned formulas between the two schemes is presented in Fig. 6. The final formula learned with sparsity regularization is

$$0.99vt - 4.97t^2$$

and learned without sparsity regularization is

$$(1.06t - 0.51t^2 + 0.5t^3 + 1.19t^4)v - 0.2t - 3.38t^2 - 2.52t^3 - 1.09t^4.$$

From the results, we can easily observe that our framework can be well equipped with the sparsity regularization and generate the optimal solution in this experiment. A noteworthy point in Fig. 6(a) is that the output formula without sparsity regularization fits in the given points quickly, but failed to learn the compact form. This observation validates our claim that there are many possible solutions given only a short clip of object motion and our framework is able to converge to the global optima.

Moreover, we study the convergence speed of w_i in Fig. 7. The change of w_i values is smooth and not trapped by potential local minimal, as the sparsity regularization with $\ell_{\frac{1}{10}}$ norm is non-convex. The results demonstrate that our framework is well compatible with regularization terms like sparsity norm and convergence speed is hardly affected by the introduced regularization terms.

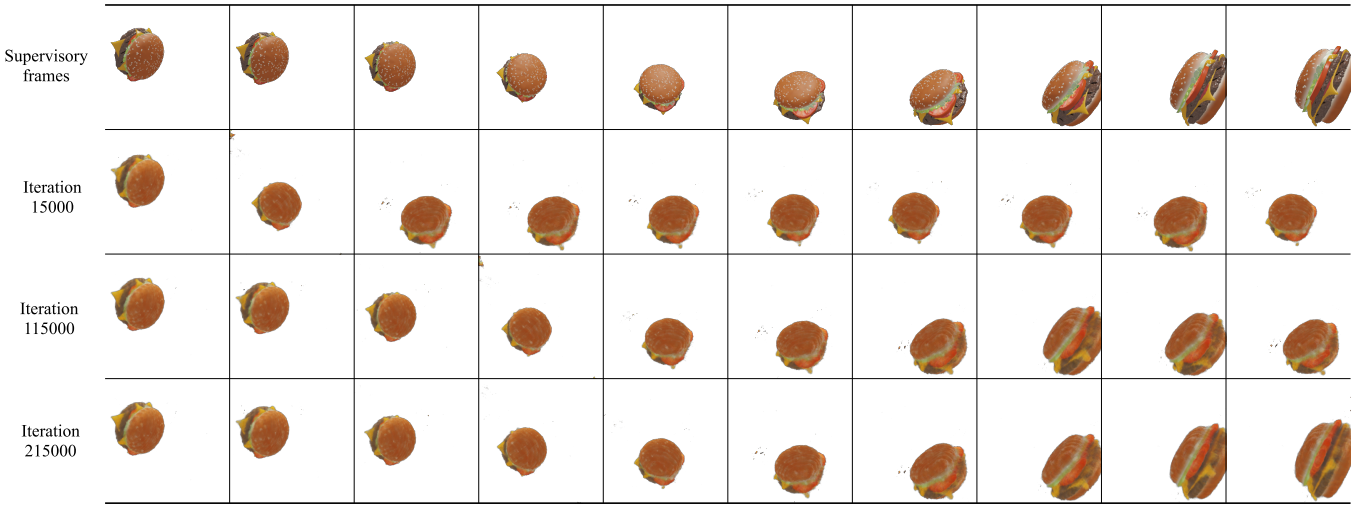

Figure 11: Visualization of rendered frames for the large angle pendulum experiment. Motion network trained with physical regularization is used for demonstration.

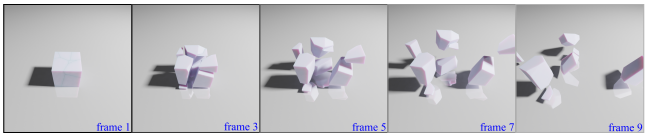

Figure 12: An animation of the explosion effect. Functions controlling the animation is unknown, so the motion of objects is generated by a black box.

Figure 13: Interpolating frames of the explosion animation with our proposed framework.

4.2 Kinematics of Large Angle Pendulum

As introduced in Section 3.2, there is no analytical kinematic equation for large angle pendulum, and finding approximate solutions has been a long standing challenge [5]. To synthesize training data, we set the initial angle as $\theta_0 = \frac{\pi}{2}$ and θ is numerically calculated. In total 101 frames are used for supervising the kinematic formula in this experiment. Some example frames are demonstrated in Fig. 8. The second derivative in the regularization term is computed with PyTorch autograd package [15].

In this experiment, the motion network is constructed with cosine basis functions:

$$\{w_1 \cos(\omega t), w_2 \cos(2\omega t), \cdots, w_6 \cos(6\omega t)\},$$

where $\{w_1, \cdots, w_6\}$ and ω are trainable parameters. The balancing parameter β is set to 1e-3, which is the same as the previous free fall experiment.

Results of the leaned kinematics are presented in Fig. 9. We can observe that both of the two learned kinematics fit the data well and finally converge to the same formula:

$$\theta(t) = 1.598 \cos(0.847t) - 0.026 \cos\left(2.541t^3\right),$$

which is consistent with the approximation theoretically derived in [5]. However, a remarkable phenomenon is that the convergence speed is much faster when physical regularization is adopted. To further study the convergence speed, in Fig. 10 we compute the Mean Squared Error (MSE) on the points not observed in the future time, which are the solid dots in Fig. 9. The reason for converging to the same good formula may be that the supervision signal (101 frames) is enough for the cosine basis functions. Besides, the faster convergence speed brought by physical regularization may due to the reason that the regularization of the differential equation produces a good optimization trajectory. The experiments demonstrate that our framework is highly extensible.

Furthermore, we visualize the rendered images from different training iterations in Fig. 11. Recall that our goal is not to generate photorealistic images, but to learn the motion of an object through the render-then-compare pipeline. The visualization validates the effectiveness of our framework since we can clearly observe the tendency of approaching supervisory frames for the rendered images.

4.3 Kinematics of Explosion

The third experiment is to study the kinematics of the explosion effect generated by Blender, which can be viewed as a black box since we have no knowledge about the internal animation functions. In Fig. 12, we demonstrate several frames that are used for supervision. As can be observed from the frames, we set a large time range between frames to make the task challenging. In total 14

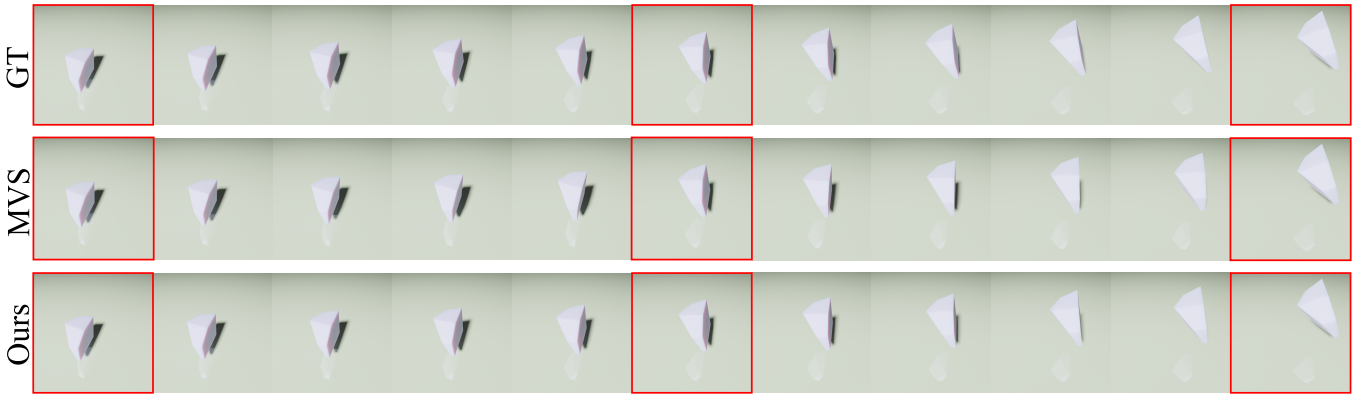

Figure 14: Comparison of the frame interpolation results. For MVS, we use the point cloud computed from COLMAP [21] for determine its position. Images with red bounding boxes are the supervisory frames.

α	0	0.001	0.01	0.1	1
FreeFall	0.91	0.04	0.01	0.00	NaN
Pendulum	1.34	0.16	0.00	0.00	NaN

(a)

β	0	0.0001	0.001	0.01	0.1
FreeFall	0.37	0.08	0.00	0.00	0.00
Pendulum	0.26	0.07	0.00	0.00	1.15

(b)

Table 1: Analysis of hyper-parameters. MSE on validation points are reported. NaN means the training does not converge.

frames are used for supervision. For modeling the motion, we set the blending functions to be sine functions with different periods and there are 32 nodes for linearly combining the functions.

We first demonstrate the results of interpolating frames of the explosion animation in Fig. 13. We interpolate the object locations and rotations from the motion network and then render a new image with the inferred values. From the figure, we can observe that the change of object positions is smooth, which validates the effectiveness of our proposed framework.

Moreover, we design a baseline method as a comparison. Since there are multiple views for each frame, a straightforward solution is to reconstruct the object frame by frame and then calculate the motion on the reconstructed objects. Specifically, we reconstruct the object per frame with multi-view stereo (MVS) implemented in COLMAP [21] and calculate the positions from the reconstructed surface. A comparison of the straightforward solution and the proposed framework is demonstrated in Fig. 14. By comparing the interpolation results from the first frame to the second frame, the simple MVS based solution performs worse than our framework in this case. The main difference is that the MVS baseline cannot well handle the vertical (z axis) rotation. We think the reason is that the vertical rotation is small between the two frames, leading to a small loss of motion on this axis. In our framework, the motion is trained with reconstruction loss, so the motion network is able to capture the difference brought by the small rotation.

4.4 Analysis of Hyper-parameters

There are two hyper-parameters in the loss function: α for balancing empty space reconstruction and β for balancing the regularization on the motion network. Though we set $\alpha = 0.1$ and $\beta = 0.001$ for all experiments, the proposed framework's sensitivity to the two hyper-parameters is still unclear. Therefore, in Tab. 1 we quantitatively study the impact of the two hyper-parameters. Several interesting points can be observed from the table: First, the training process of the network does not converge if α is 1, which indicates that a large α makes fitting the geometry hard. Second, overall the framework is not very sensitive to α and β, but the impact of β varies among the experiment. The results in the table demonstrate the reliability of the proposed framework.

5 CONCLUSION

In this paper, we propose a novel framework for learning kinematic formulas from a set of multiple view videos. The framework consists of two parts: a motion network and a differentiable renderer. The idea behind the motion network is to set up a series of blending functions and trainable weights for combining the functions. Thus after training, the kinematic formulas can be directly exported from the motion network. We employ NeRF as the differential renderer because the geometry is represented by querying the coordinates in the space. We design three experiments to validate our framework: The experiment of free fall show that our framework can converge to the correct formula with sparsity regularization; The experiment of large angle pendulum show that our framework converges faster when equipped with physical priors; The experiment of explosion animation validates the practical value of our framework. Finally, as all the experiments are carried with synthetic data, which means all input data are clean, conducting experiments with real-world data is important for future work.

ACKNOWLEDGMENTS

This work is supported in part by a gift grant from OPPO Research. The authors would like to thank Xuan Gong, Jiemin Fang, Yuzhu Sun, Helong Zhou and all the reviewers for their comments and supports.

REFERENCES

[1] Lara Adrian, Ronald J Adrian, and Jerry Westerweel. 2011. *Particle image velocimetry*. Number 30. Cambridge university press.

[2] Guy Gafni, Justus Thies, Michael Zollhofer, and Matthias Nießner. 2021. Dynamic neural radiance fields for monocular 4d facial avatar reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8649–8658.

[3] James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. 2014. From capture to simulation: connecting forward and inverse problems in fluids. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 1–11.

[4] Alan C Hindmarsh. 1983. ODEPACK, a systematized collection of ODE solvers. *Scientific computing* (1983), 55–64.

[5] Peter F Hinrichsen. 2020. Fourier analysis of the non-linear pendulum. *American Journal of Physics* 88, 12 (2020), 1068–1074.

[6] Raban Iten, Tony Metger, Henrik Wilming, Lídia Del Rio, and Renato Renner. 2020. Discovering physical concepts with neural networks. *Physical review letters* 124, 1 (2020), 010508.

[7] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. Physics-informed machine learning. *Nature Reviews Physics* (2021), 1–19.

[8] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, Yoshua Bengio and Yann LeCun (Eds.).

[9] Zhong Li, Yu Ji, Jingyi Yu, and Jinwei Ye. 2020. 3D Fluid Flow Reconstruction Using Compact Light Field PIV. In *European Conference on Computer Vision*. Springer, 120–136.

[10] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. 2021. Neural Scene Flow Fields for Space-Time View Synthesis of Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

[11] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. 2020. Neural Sparse Voxel Fields. In *Advances in Neural Information Processing Systems*.

[12] HG Maas, A Gruen, and D Papantoniou. 1993. Particle tracking velocimetry in three-dimensional flows. *Experiments in fluids* 15, 2 (1993), 133–146.

[13] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2020. Nerf: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision*. Springer, 405–421.

[14] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo-Martin Brualla. 2020. Deformable Neural Radiance Fields. *arXiv preprint arXiv:2011.12948* (2020).

[15] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (Eds.). 8024–8035.

[16] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. 2021. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10318–10327.

[17] Maziar Raissi and George Em Karniadakis. 2018. Hidden physics models: Machine learning of nonlinear partial differential equations. *J. Comput. Phys.* 357 (2018), 125–141.

[18] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. 2018. Numerical Gaussian processes for time-dependent and nonlinear partial differential equations. *SIAM Journal on Scientific Computing* 40, 1 (2018), A172–A198.

[19] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 2019. 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.

[20] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. 2020. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science* 367, 6481 (2020), 1026–1030.

[21] Johannes L Schonberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4104–4113.

[22] Jie Shen, Tao Tang, and Li-Lian Wang. 2011. *Spectral methods: algorithms, analysis and applications*. Vol. 41. Springer Science & Business Media.

[23] Russell Stewart and Stefano Ermon. 2017. Label-free supervision of neural networks with physics and domain knowledge. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[24] Ayush Tewari, Ohad Fried, Justus Thies, Vincent Sitzmann, Stephen Lombardi, Kalyan Sunkavalli, Ricardo Martin-Brualla, Tomas Simon, Jason M. Saragih, Matthias Nießner, Rohit Pandey, Sean Ryan Fanello, Gordon Wetzstein, Jun-Yan Zhu, Christian Theobalt, Maneesh Agrawala, Eli Shechtman, Dan B. Goldman, and Michael Zollhöfer. 2020. State of the Art on Neural Rendering. *Comput. Graph. Forum* 39, 2 (2020), 701–727.

[25] Yuanhao Wang, Ramzi Idoughi, and Wolfgang Heidrich. 2020. Stereo Event-based Particle Tracking Velocimetry for 3D Fluid Flow Reconstruction. In *European Conference on Computer Vision*. Springer, 36–53.

[26] John Wright, Yi Ma, Julien Mairal, Guillermo Sapiro, Thomas S Huang, and Shuicheng Yan. 2010. Sparse representation for computer vision and pattern recognition. *Proc. IEEE* 98, 6 (2010), 1031–1044.

[27] Wenqi Xian, Jia-Bin Huang, Johannes Kopf, and Changil Kim. 2021. Space-time Neural Irradiance Fields for Free-Viewpoint Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9421–9431.