VIPER: Video-Informed PDE Extraction and Recovery

Farhat Shaikh Ayan Banerjee Sandeep Gupta
IMPACT Lab, School of Computing & Augmented Intelligence (SCAI)
Arizona State University, Tempe, AZ
{fshaik12, abanerj3, Sandeep.Gupta}@asu.edu

Abstract

Video captures physical phenomena everywhere, from smartphone recordings of ocean waves to thermal cameras in industrial processes, yet extracting the governing mathematical laws from such observations remains an open challenge at the intersection of computer vision and scientific discovery. Existing PDE discovery methods require direct access to spatiotemporal field measurements and rely on numerical differentiation, which amplifies noise at rate $\mathcal{O}((\sigma/\Delta x)^k)$ for k-th order derivatives and demands complete spatial coverage. We introduce VIPER (Video-Informed PDE Extraction and Recovery), the first framework to recover explicit sparse PDE coefficients directly from RGB video of spatiotemporal dynamics. VIPER extracts scalar fields via colormap inversion, encodes temporal dynamics through a Liquid Time-Constant (LTC) neural network, and validates coefficient estimates via a differentiable PDE solver that converts the noise-amplifying differentiation problem into a noise-attenuating integration problem. On five canonical PDEs (KdV, Burgers, Kuramoto-Sivashinsky, Schrödinger, NLS), VIPER achieves up to $101\times$ improvement on clean video and 4-$29\times$ improvement under 5% noise on four PDEs. Critically, VIPER recovers coefficients from as little as 20% spatial coverage, a regime where derivative-based methods require interpolation that degrades accuracy, enabling interpretable digital twin construction from partial visual observations. Code and data are available at: https://github.com/ImpactLabASU/VIPER-CVPR2026

1. Introduction

Video captures the evolution of physical phenomena with unprecedented accessibility. From smartphone recordings of ocean waves to thermal cameras monitoring industrial processes, visual observations of spatiotemporal dynamics are now ubiquitous. Yet extracting the underlying mathematical laws that govern these phenomena from video remains an open challenge, one that sits at the confluence of computer vision and scientific discovery.

This challenge is fundamentally a computer vision problem: transforming pixel observations into interpretable physical models. Unlike traditional inverse problems that assume access to calibrated sensor data, video-based discovery must contend with colormap encodings, compression artifacts, and partial observability. Success here would democratize physical model extraction, enabling researchers to extract governing equations from the scientific visualization videos that pervade research papers, simulation outputs, and online educational content, without requiring access to the original numerical data.

The ability to recover governing partial differential equations (PDEs) from visual data would transform how we build predictive models. Consider monitoring ocean surface waves from coastal cameras, where the dispersive dynamics follow the Korteweg-de Vries equation with coefficients tied to water depth and current velocity. Or thermal imaging of cooling metal plates, where the heat equation's diffusion coefficient determines how quickly temperature gradients dissipate. Recovering these coefficients from video alone would eliminate the need for specialized measurement equipment.

Recent progress in video-based physics learning has opened new possibilities. Delfys [4] demonstrated that physical parameters can be recovered from video without segmentation masks or differentiable rendering. GradSim [11] showed that differentiable simulation enables gradient-based optimization of physical parameters from visual observations. However, a fundamental gap remains: video-based methods have focused exclusively on ordinary differential equations (ODEs) with a handful of parameters. Physical systems governed by PDEs present a qualitatively different challenge: they involve sparse combinations of spatial derivative terms, and the coefficient structure itself must be discovered.

Traditional PDE discovery methods such as PDE-FIND [18] and SINDy [3] address coefficient recovery through sparse regression on candidate libraries. Yet these methods assume direct access to dense spatiotemporal mea-

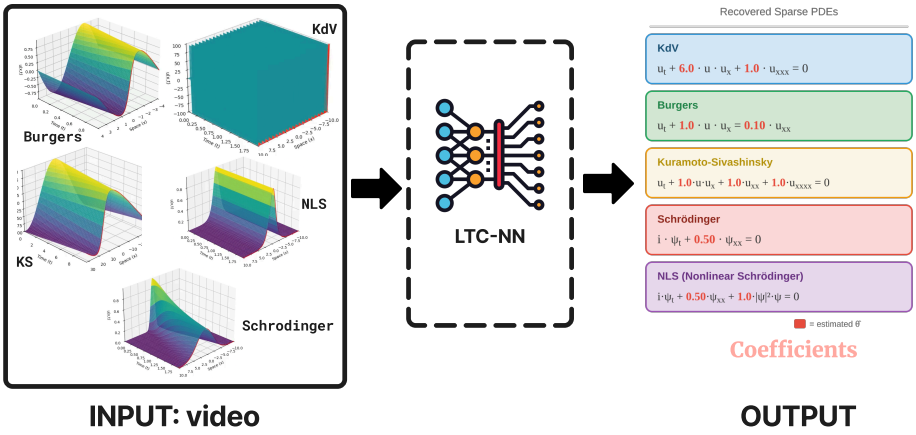

Figure 1. **High-level overview of VIPER.** Given video depicting spatiotemporal dynamics from diverse PDEs, VIPER processes the input through an LTC neural network to recover explicit sparse PDE coefficients.

surements $u(x,t)$, not video. More critically, they require computing spatial derivatives via finite differences, a process that amplifies noise dramatically: for a k-th order derivative, errors scale as $\mathcal{O}((\sigma/\Delta x)^k)$, rendering third and fourth-order terms nearly unrecoverable under realistic noise levels.

We introduce VIPER (Video-Informed PDE Extraction and Recovery), a framework that bridges these two worlds. As illustrated in Fig. 1, given video of spatiotemporal dynamics across diverse PDE types, VIPER extracts the underlying scalar field through colormap inversion, encodes temporal patterns using Liquid Time-Constant (LTC) neural networks, and validates candidate coefficients through a differentiable PDE solver. The key insight is that comparing integrated trajectories, rather than matching derivatives pointwise, converts the noise-amplifying differentiation problem into a noise-attenuating integration problem.

Contributions. We make four contributions:

- **First video-to-PDE framework**: We recover sparse governing equation coefficients directly from RGB video, achieving up to $101\times$ improvement on clean video and up to $29\times$ under noise compared to derivative-based methods.
- **Noise-robust integration-based learning**: By comparing integrated trajectories rather than derivatives, VIPER achieves 4-29$\times$ lower error than PDE-FIND under 5% measurement noise on four of five PDEs.
- **Implicit dynamics capability**: VIPER recovers accurate coefficients with only 20% spatial observability, a regime where derivative-based methods like PDE-FIND require interpolation that degrades accuracy.
- **Systematic evaluation**: We validate on five canonical PDEs spanning dispersive, diffusive, and chaotic dynamics, demonstrating strong performance on KS and Schrödinger while transparently analyzing limitations on KdV.

Method	Video	Sparse	No Diff.	Implicit
PDE-FIND [18]	✗	✓	✗	✗
WSINDy [15]	✗	✓	✓	✗
DeepMoD [2]	✗	✓	✓	✗
PINNs [17]†	✗	✓	✓	✗
Delfys [4]	✓	✗	✓	✗
GradSim [11]	✓	✗	✓	✗
VIPER (Ours)	✓	✓	✓	✓

Table 1. Comparison of PDE discovery and video-based physics methods. **Video**: accepts video input. **Sparse**: recovers explicit sparse coefficients. **No Diff**: avoids numerical differentiation. **Implicit**: handles unobserved spatial regions. VIPER satisfies all four criteria. †Inverse PINNs can recover explicit coefficients when PDE structure is specified; standard formulations assume dense collocation supervision.

Baseline selection. As Table 1 shows, video-based methods (Delfys, GradSim) recover ODE parameters but not sparse PDE coefficients. PDE discovery methods require field measurements, not video; we therefore extract fields via colormap inversion and compare against PDE-FIND, the canonical library-based approach, on identical data.

2. Related Work

Video-based parameter estimation. Recent work has explored extracting physical parameters directly from video observations. Delfys [4] recovers ODE parameters using unsupervised learning without requiring segmentation masks or differentiable rendering. NIRPI [9] and PAIG [10] employ neural implicit representations for physical parameter inference from video. GradSim [11] and ϕ-SfT [12] use differentiable rendering for parameter estimation but require known geometry or templates. Vid2Param [1] learns to predict physical parameters from video using simulation-trained models. While these methods demonstrate impres-

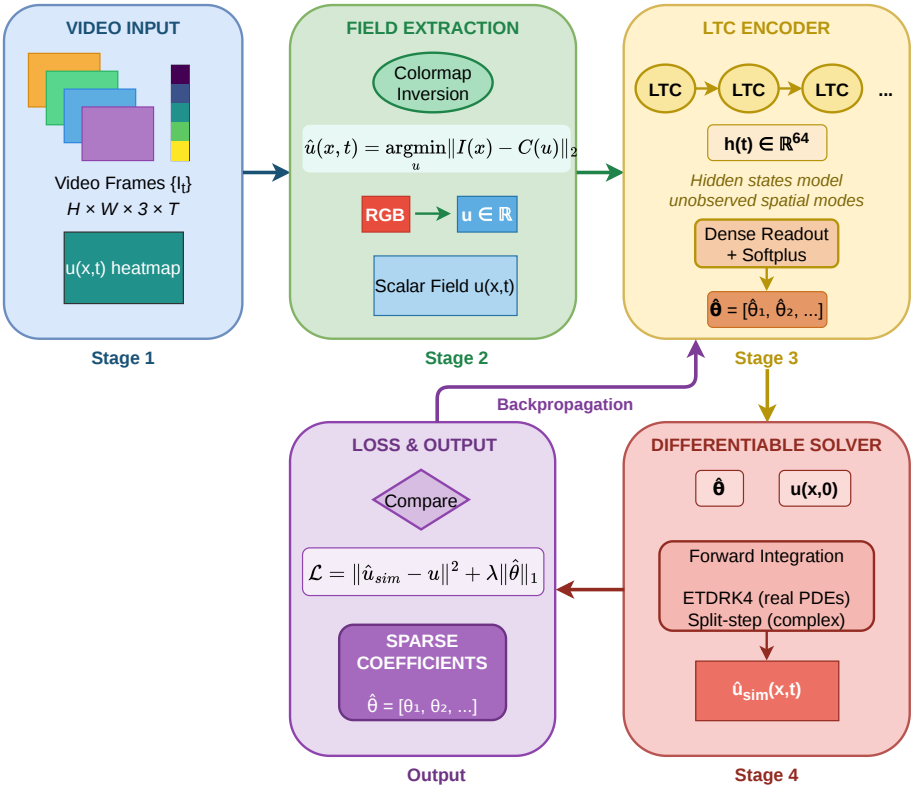

Figure 2. **VIPER pipeline. Stage 1**: Video frames are converted to scalar fields via colormap inversion. **Stage 2**: An LTC encoder processes the field sequence; hidden states $h(t) \in \mathbb{R}^{64}$ model unobserved spatial modes. **Stage 3**: A differentiable PDE solver validates coefficient estimates through forward integration. **Stage 4**: Trajectory loss with ℓ_1 sparsity drives end-to-end optimization via backpropagation.

sive results on ODE systems, they recover only a limited number of parameters and cannot identify the sparse coefficient structure of PDEs.

Sparse PDE discovery. PDE-FIND [18] pioneered data-driven PDE discovery by constructing candidate libraries evaluated via finite differences and applying sequential thresholded least squares (STLS) for sparse regression. SINDy [3] introduced this library-regression paradigm for ODEs, with extensions addressing model-predictive control [13] and ensemble robustness [6]. WSINDy [15] replaces pointwise derivatives with weak-form integration against test functions, achieving orders-of-magnitude better noise robustness. However, the weak form requires evaluating integrals over the full spatial domain, making it incompatible with partial observations.

Neural network approaches. Physics-informed neural networks (PINNs) [17] embed PDE constraints as soft losses via automatic differentiation, enabling coefficient inference without numerical derivatives. DeepHPMs [16] learn nonlinear dynamics functions but recover black-box surrogates rather than explicit sparse coefficients. DeepMoD [2] combines neural surrogates with sparse regression on an auto-differentiated library, demonstrating robustness up to 75% noise on Burgers. PDE-Net [14] learns differential operators via constrained convolutional filters. While these methods bypass finite-difference noise amplification, they all assume access to the full spatial field and do not address implicit dynamics from unobserved regions.

Implicit dynamics in dynamical systems. In the ODE setting, several works have explored model recovery when not all state variables are measured. Liquid Time-Constant neural networks (LTC-NNs) [7] are particularly suited to this problem because their forward pass takes the algebraic form of a bilinear approximation of underlying dynamics [8], enabling hidden states to represent unmeasured variables. VIPER extends this principle to the PDE setting: method-of-lines discretization converts the PDE into coupled ODEs, and unobserved spatial grid points become implicit dynamics that the LTC hidden states can model.

3. Methodology

VIPER recovers sparse PDE coefficients from video through four stages: video-to-field extraction, method-of-lines discretization, LTC-based coefficient estimation, and differentiable solver validation. Figure 2 illustrates the complete pipeline. We now describe each stage, emphasizing the

key design choices that enable noise-robust recovery under partial observability.

3.1. Problem Formulation

Given video $\mathcal{V} = \{I_t\}_{t=1}^{T}$ depicting spatiotemporal dynamics, we assume the underlying field $u(x,t)$ satisfies a PDE of the form:

$$\frac{\partial u}{\partial t} = \mathcal{F}\left(u, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots; \boldsymbol{\theta}\right), \tag{1}$$

where $\mathcal{F}$ is a differential operator drawn from a known candidate library, and $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_p]$ are the unknown sparse coefficients we seek to recover.

The challenge is that video provides only indirect, potentially degraded access to $u(x,t)$. Pixel intensities encode field values through a colormap, observations may be corrupted by noise, and portions of the spatial domain may be occluded or outside the camera's field of view. We denote the observed spatial domain as $\Omega_{\text{obs}} \subseteq \Omega$, which may cover as little as 20% of the full domain. Our goal is to recover $\hat{\boldsymbol{\theta}} \approx \boldsymbol{\theta}^*$ despite these challenges.

3.2. Video-to-Field Extraction

Scientific visualizations encode scalar fields using standardized colormaps that map field values to RGB colors. Common examples include viridis, jet, plasma, and inferno, each providing a bijective mapping from scalar values to colors. We leverage this structure by inverting the colormap to recover the underlying field from video frames.

Given a known colormap $C : [u_{\min}, u_{\max}] \to \mathbb{R}^3$, we construct a lookup table by sampling C at $A = 256$ uniformly spaced values. For each pixel in frame I_t, we find the field value whose colormap entry minimizes Euclidean distance in RGB space:

$$\hat{u}(x_i, t_j) = \underset{u \in [u_{\min}, u_{\max}]}{\arg\min} \|I_j(x_i) - C(u)\|_2. \tag{2}$$

This nearest-neighbor matching against the A-entry lookup table is computationally efficient when vectorized across all pixels, and robust to minor color variations introduced by video compression artifacts. The output is an estimated spatiotemporal field $\hat{u} \in \mathbb{R}^{n_t \times n_x}$ suitable for downstream processing. Figure 3 illustrates the inversion process.

3.3. Method-of-Lines Discretization

To bridge continuous PDE dynamics with discrete sequence modeling, we apply method-of-lines discretization [19]. Discretizing the spatial domain into $n_x = 128$ grid points $\{x_1, \ldots, x_{n_x}\}$ converts Eq. 1 into a system of n_x coupled ordinary differential equations:

$$\frac{d\mathbf{u}(t)}{dt} = \mathbf{F}(\mathbf{u}(t); \boldsymbol{\theta}), \quad \mathbf{u}(t) = [u(x_1, t), \ldots, u(x_{n_x}, t)]^\top. \tag{3}$$

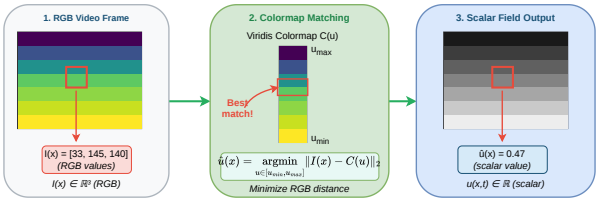

Figure 3. **Colormap inversion.** RGB pixel values from video frames are mapped back to scalar field values by finding the closest match in the colormap lookup table. This enables extraction of $u(x,t)$ from standard scientific visualizations without requiring access to the original simulation data.

Spatial derivatives within $\mathbf{F}$ are computed via Fourier spectral differentiation rather than finite differences:

$$\frac{\partial^k u}{\partial x^k} = \mathcal{F}^{-1}\left[(i\kappa)^k \hat{u}(\kappa)\right], \tag{4}$$

where κ is the wavenumber vector and $\mathcal{F}$ denotes the discrete Fourier transform. Spectral differentiation is exact for band-limited signals and, crucially, avoids the $\mathcal{O}((\sigma/\Delta x)^k)$ noise amplification inherent in finite-difference stencils used by PDE-FIND [18].

Handling implicit dynamics. When only a subset $\Omega_{\text{obs}} \subset \Omega$ is observed, the unobserved grid points become latent variables in the coupled ODE system. This is precisely the setting of *implicit dynamics*: the measured nodes correspond to observed state variables, while missing nodes act as unmeasured states that still influence the observed dynamics through spatial coupling in Eq. 3. For instance, at 20% coverage, approximately 102 of 128 spatial points are unobserved, yet their values affect the evolution of the 26 observed points through derivative terms. This reframing allows us to apply implicit-dynamics recovery techniques originally developed for ODEs [7] to the PDE setting.

3.4. LTC Encoder for Coefficient Estimation

We employ a Liquid Time-Constant (LTC) network [7] to process the extracted time series and estimate PDE coefficients. LTC networks are particularly suited to this task for two reasons.

First, their forward pass has the algebraic form of a bilinear approximation of dynamical systems [8], enabling hidden states to naturally represent unmeasured variables. Second, the input-dependent time constant allows the network to modulate its temporal response based on the input signal, capturing the diverse timescales present across different PDEs.

The LTC hidden state $h(t) \in \mathbb{R}^V$ with $V = 64$ units evolves according to:

$$\frac{dh_i}{dt} = -\frac{h_i}{\tau_i(1 + \tau_i f_{NN})} + f_{NN}(h, I, t; \omega) \cdot A, \tag{5}$$

where $\tau_i > 0$ is the input-dependent time constant, ω and A are learnable parameters, and f_{NN} is a neural network mapping.

Overdetermined recovery. When the number of hidden units V exceeds the number of unobserved spatial points ($V = 64 \gg n_x - |\Omega_{\text{obs}}|$ for high coverage), the hidden states form an overdetermined representation of the latent dynamics. This redundancy provides robustness: the network can represent the implicit dynamics through multiple consistent hidden-state configurations, reducing sensitivity to initialization and noise.

Sparse readout layer. A dense layer maps the mean-pooled hidden state to coefficient estimates:

$$\hat{\boldsymbol{\theta}} = \text{Softplus}\left(W_2\sigma(W_1\bar{h} + b_1) + b_2\right), \tag{6}$$

where $\bar{h} = \frac{1}{T}\sum_t h(t)$ is the temporal average of hidden states, σ is the tanh activation, and Softplus ensures positive coefficient estimates. The ℓ_1 penalty on $\hat{\boldsymbol{\theta}}$ encourages the parsimonious coefficient structure expected of physical laws [3].

3.5. Differentiable PDE Solver

Following the solver-in-the-loop paradigm [20], we embed a differentiable PDE solver in the training loop to enable noise-robust recovery. This design choice fundamentally changes how noise affects coefficient estimation.

Integration vs. differentiation. Traditional sparse regression methods like PDE-FIND construct the equation $\partial u/\partial t = \Theta\Phi(u)$ and solve for coefficients Θ by matching derivatives pointwise. This requires computing $\partial u/\partial t$ and spatial derivatives from noisy data via finite differences, amplifying noise at rate $\mathcal{O}((\sigma/\Delta x)^k)$ for k-th order derivatives. For the fourth-order term in KS or the third-order term in KdV, this amplification is catastrophic under realistic noise levels.

VIPER inverts this approach: given estimated coefficients $\hat{\boldsymbol{\theta}}$ and initial condition $u(x, 0)$, the solver produces predicted trajectories by forward integration:

$$\hat{\mathbf{Y}} = \text{SOLVE}(u(x, 0), \hat{\boldsymbol{\theta}}). \tag{7}$$

Comparing integrated trajectories rather than derivatives converts the noise-amplifying differentiation problem into a noise-attenuating integration problem [2].

PDE-specific integration schemes. Numerical stability requires matching the integrator to the PDE structure:

- **Real-valued PDEs** (KdV, KS): Exponential time-differencing fourth-order Runge-Kutta (ETDRK4) [5] schemes that handle stiff linear terms via matrix exponentials and nonlinear terms explicitly. For Burgers, we use spectral Euler, where spatial derivatives are computed via FFT and time is advanced with explicit Euler stepping.
- **Complex-valued PDEs** (Schrödinger, NLS): Split-step Fourier methods [21] that alternate between linear dispersion half-steps and nonlinear phase-rotation half-steps.

Stability is further ensured through coefficient clamping ($\hat{\theta}_i \in [0, 20]$) applied only within the solver to prevent numerical overflow, solution clamping ($|u| \leq 100$), and gradient clipping ($\|\nabla\|_{\max} = 1.0$). The ℓ_1 sparsity penalty operates on unclamped parameters, allowing coefficients to approach zero during optimization.

3.6. Loss Function and Training

The total loss combines trajectory matching with sparsity regularization:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{traj}} + \lambda\|\hat{\boldsymbol{\theta}}\|_1, \tag{8}$$

where $\lambda = 10^{-3}$ controls sparsity strength.

The trajectory loss measures mean squared error between simulated and observed fields over the observed domain:

$$\mathcal{L}_{\text{traj}} = \frac{1}{N}\sum_{x_j \in \Omega_{\text{obs}}}\sum_{k=1}^{n_t}\left\|\hat{u}(x_j, t_k) - \tilde{u}(x_j, t_k)\right\|^2. \tag{9}$$

This formulation naturally handles partial observability: unobserved regions contribute to the dynamics through the coupled ODE system but do not directly enter the loss. The solver propagates information from observed to unobserved regions through spatial derivatives, while the LTC hidden states model the unobserved dynamics implicitly.

Optimization details. We use Adam with learning rate 5×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$. Training runs for up to 500 epochs with early stopping (patience 100) based on coefficient error when ground truth is available for validation. A ReduceLROnPlateau scheduler (factor 0.5, patience 50) adapts the learning rate during training.

4. Experiments

We evaluate VIPER under three experimental conditions: clean video (Sec. 4.2), noisy video (Sec. 4.3), and partial spatial observability (Sec. 4.4). Our experiments answer three questions: (1) Can video-based discovery match or exceed methods operating on ground-truth field data? (2) How robust is the approach to realistic video noise? (3) Can we recover coefficients when large portions of the spatial domain are occluded?

4.1. Experimental Setup

PDEs. We evaluate on five canonical equations spanning dispersive waves, viscous dynamics, spatiotemporal chaos, and quantum mechanics:

- **KdV**: $u_t + 6uu_x + u_{xxx} = 0$ (dispersive solitons)
- **Burgers**: $u_t + uu_x - 0.1u_{xx} = 0$ (viscous shocks)
- **KS**: $u_t + uu_x + u_{xx} + u_{xxxx} = 0$ (spatiotemporal chaos)

- **Schrödinger**: $iu_t + 0.5u_{xx} = 0$ (linear dispersion)
- **NLS**: $iu_t + 0.5u_{xx} + |u|^2 u = 0$ (nonlinear optics)

These PDEs test different aspects of the method: KdV and KS have high-order derivatives (third and fourth order) that challenge finite-difference methods; Schrödinger and NLS involve complex-valued fields requiring specialized solvers; Burgers provides a simpler second-order baseline with shock formation.

Video generation. Spatiotemporal solutions are computed using spectral methods with $n_x = 128$ spatial grid points and PDE-specific time steps ensuring numerical stability. Solutions are rendered to 256×128 RGB video frames using the viridis colormap, producing videos that resemble standard scientific visualizations common in computational physics publications. This setup reflects the real-world scenario where a researcher encounters a simulation video and wishes to extract the underlying model. Results are averaged over 5 random seeds $\{42, 123, 456, 789, 1011\}$ with different initial conditions.

Baseline: PDE-FIND. We compare against PDE-FIND [18], the canonical sparse PDE discovery method, which constructs a library of candidate terms evaluated via finite differences and applies sequential thresholded least squares (STLS) with $\lambda = 10^{-3}$. To ensure fair comparison, PDE-FIND receives the same extracted field data as VIPER after colormap inversion, giving both methods access to equivalent information.

We do not compare against video-only baselines (Delfys [4], gradSim [11], NIRPI [9]) because they solve a fundamentally different problem: estimating a small number of known ODE parameters rather than discovering sparse PDE coefficient structure from a candidate library. As shown in Table 1, these methods do not produce sparse coefficients and thus cannot be meaningfully evaluated on our task. We note that WSINDy [15] is applicable to Experiments 1 and 2 (full spatial coverage); we leave this comparison to the extended version as our pipeline does not currently implement weak-form integral evaluation. WSINDy is fundamentally incompatible with Experiment 3 (partial coverage), which is VIPER's primary contribution.

Metrics. We report coefficient error (%): $\text{Err} = \|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*\| / \|\boldsymbol{\theta}^*\| \times 100$, where lower is better. An error of 100% corresponds to $\|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}^*\| = \|\boldsymbol{\theta}^*\|$; values may exceed 100% when estimated coefficients diverge significantly from ground truth (e.g., PDE-FIND on KdV yields 534.70%).

4.2. Experiment 1: Clean Video

Table 2 compares VIPER against PDE-FIND on noise-free video. VIPER achieves lower error on all five PDEs, with improvements ranging from $1.0\times$ (KS) to $101\times$ (KdV).

The dramatic improvement on KdV (5.27% vs. 534.70%) demonstrates the advantage of integration-based learning for dispersive dynamics with third-order spatial derivatives.

PDE	VIPER (%)	PDE-FIND (%)	Improv.
KdV	**5.27** ±1.57	534.70	**101**×
Burgers	**0.67** ±0.42	1.95	2.9×
KS	**2.30** ±2.42	2.31	1.0×
Schröd.	**6.22** ±11.06	92.95	**14.9**×
NLS	**28.56** ±2.05	100.00	3.5×

Table 2. **Clean video results.** Coefficient error (%) on noise-free video. VIPER outperforms PDE-FIND on all five PDEs, with $101\times$ improvement on KdV where finite-difference errors compound for third-order derivatives. Lower coefficient error (%) is better; **green** marks best per PDE.

Even on noise-free data, PDE-FIND's finite-difference computation of u_{xxx} accumulates discretization errors that compound during sparse regression, causing coefficient estimates to diverge.

KS shows near-parity (2.30% vs. 2.31%), indicating that both methods handle fourth-order derivatives adequately when data is clean and fully observed. This result validates our experimental setup: when conditions favor PDE-FIND, VIPER matches its performance.

NLS remains challenging for VIPER (28.56%) due to the complex-valued nonlinear interaction $|u|^2 u$, which creates a more complex loss landscape. Nevertheless, VIPER still outperforms PDE-FIND's complete failure (100%) by $3.5\times$.

4.3. Experiment 2: Noisy Video

Real-world videos contain compression artifacts, sensor noise, and quantization errors. Table 3 evaluates robustness under 5% additive Gaussian noise applied to the extracted field: $\tilde{u} = u + 0.05 \cdot \text{std}(u) \cdot \epsilon$, $\epsilon \sim \mathcal{N}(0, 1)$.

PDE-FIND degrades to near-total failure ($\geq 99\%$ error) on all five PDEs. This collapse is expected: numerical differentiation amplifies noise at rate $\mathcal{O}((\sigma/\Delta x)^k)$ for k-th order derivatives, and even moderate 5% noise renders higher-order derivative estimates unusable for sparse regression.

VIPER maintains robust performance on four of five PDEs, achieving improvements of $4.3\times$ to $28.8\times$. The integration-based loss acts as a natural low-pass filter: comparing integrated trajectories attenuates rather than amplifies measurement noise, a key advantage for video-based discovery where some noise is unavoidable.

KdV is a notable exception, with only $1.15\times$ improvement (87.30% vs. 100%). The dispersive u_{xxx} term creates oscillatory soliton solutions where small coefficient errors lead to large phase mismatches in the predicted trajectory, making the loss landscape difficult to optimize. We analyze this limitation in detail in Sec. 5.

4.4. Experiment 3: Implicit Dynamics

A key advantage of VIPER is its ability to operate under partial spatial observability, a setting motivated by real-world

PDE	VIPER (%)	PDE-FIND (%)	Improv.
KdV	87.30 ±2.47	100.00	1.15×
Burgers	**10.97** ±0.67	100.00	**9.1**×
KS	**3.47** ±1.45	99.95	**28.8**×
Schröd.	**12.51** ±9.25	99.02	**7.9**×
NLS	**23.36** ±6.13	99.82	**4.3**×

Table 3. **Noisy video results (5% Gaussian noise).** PDE-FIND degrades to complete failure on all PDEs due to noise amplification in finite differences. VIPER maintains robust performance on 4/5 PDEs through its integration-based loss, achieving up to 28.8× improvement. Lower coefficient error (%) is better; **green** marks best per PDE.

scenarios where sensors have limited coverage or video frames show only a portion of the physical domain.

Table 4 evaluates VIPER when only a fraction of the spatial domain is observed. Random spatial columns are masked and filled via linear interpolation before processing, simulating occluded or cropped video regions.

This setting is *fundamentally incompatible* with PDE-FIND and all derivative-based methods: computing u_x or u_{xx} requires data on a complete spatial grid to evaluate finite differences or weak-form integrals. For comparison, we show PDE-FIND results on the interpolation-filled data, representing an optimistic upper bound on its performance.

Strong performers. KS shows remarkable robustness across all coverage levels, with error below 7% even at 20% coverage and achieving sub-1% at 60-80% coverage (up to 135× improvement).

Coverage-dependent recovery. Burgers shows graceful degradation: error decreases from 15.40% at 20% coverage to 1.08% at 80% coverage, suggesting that shock-forming dynamics benefit from observing the shock location directly. Schrödinger shows high variance across seeds (std ≈ 15-19%) due to bimodal optimization behavior, though it consistently outperforms PDE-FIND by 4.5-6.5×.

Challenging cases. KdV fails across all coverage levels (> 90% error), consistent with its poor performance under noise. NLS shows moderate error (26-28%) but remains substantially better than PDE-FIND.

4.5. Results Summary

Figure 4 summarizes our findings across all three experimental conditions.

Clean video. VIPER matches or exceeds PDE-FIND on all PDEs, with dramatic gains on KdV (101×) and Schrödinger (14.9×) where finite-difference errors compound.

Noisy video. Integration-based learning provides 4-29× improvement on four PDEs (KS, Schrödinger, Burgers, NLS). KdV remains challenging due to dispersive phase sensitivity.

PDE	Cov.	VIPER (%)	PDE-FIND (%)	Improv.
KS	20%	6.48 ± 5.10	97.48 ± 0.57	**15**×
	40%	3.56 ± 3.25	95.22 ± 0.85	**27**×
	60%	**0.69 ± 0.38**	92.74 ± 1.64	**135**×
	80%	0.72 ± 0.34	91.59 ± 0.73	**126**×
Schröd.	20%	**14.28** ± 19.09	92.91 ± 3.70	6.5×
	40%	20.76 ± 15.12	93.15 ± 1.00	4.5×
	60%	**14.28** ± 19.09	93.47 ± 0.39	6.5×
	80%	14.90 ± 18.56	93.03 ± 0.49	6.2×
Burgers	20%	15.40 ± 15.00	90.19 ± 1.98	5.9×
	40%	7.18 ± 8.20	80.26 ± 4.44	11.2×
	60%	2.63 ± 3.65	66.13 ± 8.94	**25**×
	80%	**1.08 ± 0.73**	40.97 ± 12.09	**38**×
NLS	20%	27.81 ± 7.17	100.00 ± 0.00	3.6×
	40%	27.07 ± 6.67	100.00 ± 0.00	3.7×
	60%	**26.25 ± 7.10**	100.00 ± 0.00	3.8×
	80%	26.26 ± 7.68	100.00 ± 0.00	3.8×
KdV	20%	**90.04** ± 4.63	100.00 ± 0.00	1.1×
	40%	96.08 ± 11.19	100.00 ± 0.00	1.0×
	60%	92.63 ± 9.41	100.00 ± 0.00	1.1×
	80%	92.39 ± 4.68	100.00 ± 0.00	1.1×

Table 4. **Implicit dynamics results.** Coefficient error (%) (mean ± std over 5 seeds) at varying spatial coverage. PDE-FIND operates on interpolation-filled data (optimistic baseline). VIPER achieves sub-1% error on KS at 60-80% coverage. Both methods struggle on KdV due to dispersive phase sensitivity. Lower coefficient error (%) is better; **green** marks best per PDE.

Implicit dynamics. VIPER operates in a regime where derivative-based methods fundamentally cannot, achieving sub-1% error on KS at 60-80% coverage. This capability is particularly relevant for computer vision applications where video frames may show only partial views of the physical domain.

Figure 5 shows training dynamics across PDEs. KS and Burgers converge quickly (< 100 epochs) with low final error, while KdV exhibits high variance and slow convergence, consistent with its challenging dispersive dynamics.

4.6. Implementation Details

Table 5 summarizes training hyperparameters. All experiments were conducted on CPU. Training time ranges from 5 minutes (KS with early stopping) to 45 minutes (NLS requiring more epochs). The LTC network uses 64 hidden units, following standard configurations for continuous-time dynamical modeling [8].

5. Discussion and Conclusion

When does VIPER excel? VIPER achieves strong performance on KS (chaotic, fourth-order) across all experimental conditions. The integration-based loss is particularly advantageous for high-order derivatives where finite differences

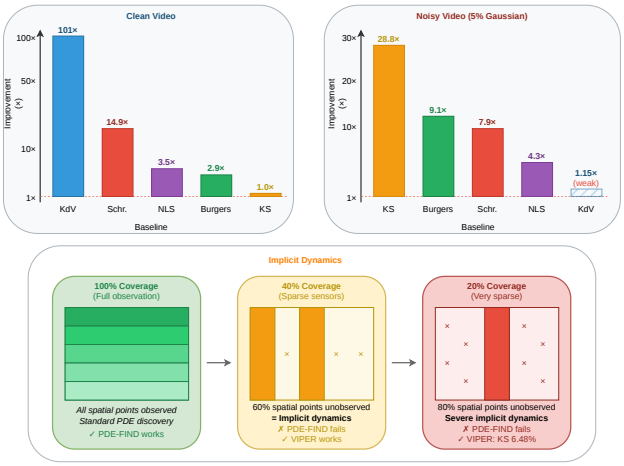

Figure 4. **Results summary. Top**: Improvement factor over PDE-FIND on clean and noisy video (log scale). **Bottom**: Implicit dynamics performance at 20-80% spatial coverage. VIPER excels where derivative-based methods struggle or fail entirely.

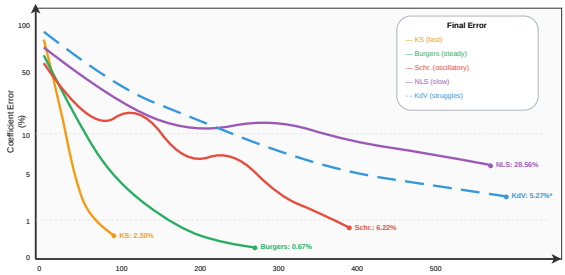

Figure 5. **Training dynamics.** Coefficient error (%) vs. training epoch for each PDE. KS and Burgers converge quickly to low error; Schrödinger shows oscillatory behavior before settling; KdV struggles with high variance throughout training.

Hyperparameter	Value
LTC hidden units	64
Learning rate	5×10^{-4}
Optimizer	Adam (β_1=0.9, β_2=0.999)
Max epochs	500
Early stopping patience	100
LR scheduler	ReduceLROnPlateau (patience 50)
Sparsity λ	10^{-3}
Spatial grid n_x	128
Video resolution	256×128
Random seeds	$\{42, 123, 456, 789, 1011\}$

Table 5. Training hyperparameters. All settings are fixed across PDEs except for PDE-specific solver configurations.

are most unstable. On KS, VIPER achieves sub-1% error at 60-80% spatial coverage, demonstrating robust coefficient recovery in regimes where derivative-based methods struggle.

Understanding KdV limitations. KdV consistently underperforms, achieving only marginal improvement under noise ($1.15\times$) and failing under implicit dynamics. This difficulty stems from the dispersive nature of the u_{xxx} term, which creates oscillatory soliton solutions where small coefficient perturbations cause significant phase shifts. Unlike diffusive PDEs (Burgers, KS) where errors decay over time, dispersive errors accumulate: a 5% error in the dispersion coefficient produces trajectories that drift increasingly out of phase with the target, creating a trajectory loss landscape with multiple local minima separated by phase barriers. This analysis suggests that dispersive PDEs may require specialized phase-invariant loss functions or multi-scale optimization strategies, an important direction for future work.

NLS challenges. NLS remains challenging (26-28% error) due to the complex-valued nonlinear interaction $|u|^2 u$, which couples amplitude and phase dynamics. However, VIPER still substantially outperforms PDE-FIND across all conditions, indicating that the integration-based approach provides value even when absolute accuracy is limited.

Practical applicability. Scientific visualization videos are ubiquitous in research and industry: simulation outputs, thermal imaging, flow visualization, and medical imaging all encode scalar fields via standardized colormaps. Our colormap inversion approach is immediately applicable to any video using viridis, jet, plasma, or similar mappings. Extension to real sensors requires only colormap calibration or direct pixel-to-value mapping, which is simpler than the tracking and segmentation required by existing video physics methods [4, 11]. Current validation uses synthetic videos; real-world video presents additional challenges including camera motion, lighting variations, and unknown colormaps that we leave for future investigation.

Conclusion. We presented VIPER, the first framework to recover sparse PDE coefficients directly from video. By combining colormap-based field extraction, LTC temporal encoding, and differentiable PDE simulation, VIPER bridges computer vision with sparse equation discovery. Experiments demonstrate up to $101\times$ improvement on clean video and 4-29$\times$ improvement under noise on KS, Schrödinger, Burgers, and NLS. Critically, VIPER enables coefficient recovery from as little as 20% spatial coverage, a regime where derivative-based methods have not been demonstrated. Future work will investigate phase-invariant losses for dispersive PDEs, extension to two-dimensional spatial domains, and validation on real experimental videos captured from physical systems.

Acknowledgments

This project is partially funded by DARPA AMP-N6600120C4020, DARPA FIRE-P000050426, NSF FDT-Biotech grant (2436801), NIH R21 grant (1R21HL175632).

References

[1] Martin Asenov, Michael Burke, Daniel Angelov, Todor Davchev, Kartic Subr, and Subramanian Ramamoorthy. Vid2param: Modeling of dynamics parameters from video. *IEEE Robotics and Automation Letters*, 5(2):414–421, 2020. 2

[2] Gert-Jan Both, Subham Choudhury, Pierre Sens, and Remy Kusters. Deepmod: Deep learning for model discovery in noisy data. *Journal of Computational Physics*, 428:109985, 2021. 2, 3, 5

[3] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016. 1, 3, 5

[4] Alejandro Castaneda Garcia, Jan Warchocki, Jan van Gemert, Daan Brinks, and Nergis Tomen. Learning physics from video: Unsupervised physical parameter estimation for continuous dynamical systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 2, 6, 8

[5] Steven M Cox and Paul C Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002. 5

[6] Urban Fasel, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Ensemble-sindy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control. *Proceedings of the Royal Society A*, 478(2260): 20210904, 2022. 3

[7] Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7657–7666, 2021. 3, 4

[8] Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 4(11):992–1003, 2022. 3, 4, 7

[9] Florian Hofherr, Lukas Koestler, Florian Bernard, and Daniel Cremers. Neural implicit representations for physical parameter inference from a single video. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2093–2103, 2023. 2, 6

[10] Miguel Jaques, Michael Burke, and Timothy Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *International Conference on Learning Representations (ICLR)*, 2020. 2

[11] Krishna Murthy Jatavallabhula, Miles Macklin, Florian Golemo, Vikram Voleti, Linda Petrini, Martin Weiss, Breandan Considine, Jerome Parent-Levesque, Kevin Xie, Kenny Erleben, Liam Paull, Florian Shkurti, Derek Nowrouzezahrai, and Sanja Fidler. gradsim: Differentiable simulation for system identification and visuomotor control. In *International Conference on Learning Representations (ICLR)*, 2021. 1, 2, 6, 8

[12] Navami Kairanda, Edith Tretschk, Mohamed Elgharib, Christian Theobalt, and Vladislav Golyanik. ε-sft: Shape-from-template with a physics-based deformation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3948–3958, 2022. 2

[13] Eurika Kaiser, J Nathan Kutz, and Steven L Brunton. Sparse identification of nonlinear dynamics for model predictive control in the low-data limit. *Proceedings of the Royal Society A*, 474(2219):20180335, 2018. 3

[14] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. Pde-net: Learning pdes from data. In *International Conference on Machine Learning (ICML)*, pages 3208–3216. PMLR, 2018. 3

[15] Daniel A Messenger and David M Bortz. Weak sindy for partial differential equations. *Journal of Computational Physics*, 443:110525, 2021. 2, 3, 6

[16] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19(25):1–24, 2018. 3

[17] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. 2, 3

[18] Samuel H Rudy, Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4):e1602614, 2017. 1, 2, 3, 4, 6

[19] William E Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations*. Academic Press, 1991. 4

[20] Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. In *Advances in Neural Information Processing Systems*, pages 6111–6122, 2020. 5

[21] J A C Weideman and B M Herbst. Split-step methods for the solution of the nonlinear schrödinger equation. *SIAM Journal on Scientific and Statistical Computing*, 7(1):49–76, 1986. 5