VIPER: Video-Informed PDE Extraction and Recovery Supplementary Material

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

This supplement addresses the three items requested by the program committee: (1) dataset generation details, video rendering procedure, and sample video frames (Sec. 1), (2) standard deviations for partial observability results (now included in Table 4 of the main paper), and (3) candidate library contents and size (Sec. 2).

1. Dataset Generation Details

All experiments use synthetically generated spatiotemporal fields from ground-truth PDE solutions. The pipeline consists of three stages: (i) numerical PDE simulation, (ii) video rendering via colormap mapping, and (iii) experiment-specific corruption (noise or spatial occlusion).

1.1. PDE Simulation

All five PDEs are solved on a 1D periodic spatial domain using spectral methods. Periodic boundary conditions are enforced implicitly through the discrete Fourier transform. Each simulation produces a scalar field $u \in \mathbb{R}^{n_t \times n_x}$ where $n_x = 128$ and $n_t = \lfloor T/\Delta t \rfloor + 1$.

Table 1 reports the complete specification for each PDE, including the governing equation, ground-truth coefficients, spatial domain length L, total simulation time T, time step Δt, initial condition, and numerical solver.

Solver details. We employ three numerical integration schemes matched to each PDE's mathematical structure.

ETDRK4 (Exponential Time Differencing, 4th-order Runge-Kutta) is used for KdV and KS. This scheme decomposes the PDE into linear and nonlinear parts; the linear part is integrated exactly in Fourier space via matrix exponentials while the nonlinear part uses a 4th-order Runge-Kutta scheme with contour integral coefficients ($M = 32$ quadrature points for KdV, $M = 16$ for KS).

Spectral Euler is used for Burgers. Spatial derivatives are computed spectrally via FFT, and the equation is advanced using explicit (forward) Euler time stepping. The milder stiffness of the Burgers equation (second-order diffusion only) permits this simpler scheme.

Split-step Fourier is used for NLS. It alternates between half-step nonlinear integration in physical space and full-step linear integration in Fourier space.

Spectral (exact) is used for the linear Schrödinger equation via an exact exponential propagator in Fourier space.

1.2. Experiment 1: Clean Video

Each of the 5 PDEs is solved with each of 5 random seeds $\{42, 123, 456, 789, 1011\}$, yielding 25 clean simulations. Seeds control initial condition perturbations (small random phase shifts) while preserving the qualitative dynamics. Both VIPER and PDE-FIND receive the identical $u(x, t)$ field with no noise or occlusion applied.

1.3. Experiment 2: Noisy Video

Additive Gaussian noise is applied to the clean spatiotemporal field before processing by either method:

$$\tilde{u}(x,t) = u(x,t) + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0,1), \tag{1}$$

where $\sigma = 0.05 \cdot \text{std}(u)$ and $\text{std}(u)$ is computed over the entire spatiotemporal field. This corresponds to a 5% noise level relative to the signal amplitude. Results in Table 3 of the main paper are averaged over 5 seeds $\{42, 123, 456, 789, 1011\}$.

We additionally tested salt-and-pepper noise (random pixels set to field $\min/\max$ at probability 0.01 and 0.05) and spatial blur (uniform 1D filter with kernel sizes 3, 5, and 7 in wrap mode). These results are consistent with the Gaussian noise findings and are omitted from the main paper for brevity.

1.4. Experiment 3: Partial Observability (Implicit Dynamics)

To simulate partial spatial coverage, we apply the following occlusion procedure:

1. Given a target coverage fraction $\gamma \in \{0.2, 0.4, 0.6, 0.8\}$, compute $n_{\text{keep}} = \lfloor 128 \cdot \gamma \rfloor$ observed spatial locations.

PDE	Equation	True θ^*	L	n_x	T	Δt	Initial Condition	Solver
KdV	$u_t + \alpha u u_x + \beta u_{xxx} = 0$	α=6, β=1	20	128	10	0.01	Sech soliton: $c\,\mathrm{sech}^2(\sqrt{c/2}\,x/2)$, c=0.5	ETDRK4
Burgers	$u_t + u u_x = \nu u_{xx}$	ν=0.1	8	128	2	0.005	$-\sin(\pi x/(L/2))$	Spectral Euler
KS	$u_t + u u_x + u_{xx} + \nu u_{xxxx} = 0$	ν=1.0	64	128	50	0.05	$\cos(2\pi x/L)\,(1 + 0.1\sin(2\pi x/L))$	ETDRK4
Schröd.	$iu_t + \alpha u_{xx} = 0$	α=0.5	20	128	5	0.01	Gaussian: $e^{-x^2}e^{i0.5x}$; field = $\lvert u\rvert$	Spectral (exact)
NLS	$iu_t + \alpha u_{xx} + \beta\lvert u\rvert^2 u = 0$	α=0.5, β=1	20	128	5	0.01	sech(x) (complex); field = $\lvert u\rvert$	Split-step

Table 1. Complete PDE simulation specifications. All PDEs use periodic boundary conditions on the spatial domain $[0, L]$ with $n_x = 128$ grid points.

2. Randomly select n_{keep} spatial columns via uniform sampling without replacement (seeded for reproducibility).
3. Set unselected columns to NaN.
4. Interpolate missing values along the spatial axis per timestep using linear interpolation (with extrapolation at boundaries).

Both VIPER and PDE-FIND receive the interpolation-filled field. PDE-FIND's results therefore represent an *optimistic upper bound* on its performance, since it receives a complete (though approximate) spatial field. For example, at 20% coverage, approximately 102 of 128 spatial columns are interpolated rather than measured.

Results are averaged over 5 random seeds $\{42, 123, 456, 789, 1011\}$ per PDE per coverage level, yielding 60 total simulations (5 PDEs $\times$ 4 coverage levels $\times$ 5 seeds).

1.5. Video Rendering

Each 1D spatial field $u(x, t)$ is rendered as a 2D heatmap image. The horizontal axis represents the spatial coordinate x (128 grid points), and field values are mapped to RGB colors using the `viridis` colormap from matplotlib. Specifically, each temporal snapshot $u(t_i, x)$ is a 1D array of 128 values that is mapped to RGB via the colormap, producing a 1×128 pixel row which is then replicated vertically to fill the full frame height. The horizontal axis thus encodes the spatial coordinate x while the vertical axis carries no additional information. Color limits are fixed to $[u_{\min}, u_{\max}]$ computed over the entire simulation, ensuring consistent color-to-value mapping across all frames. Each temporal snapshot produces one video frame, with the full spatiotemporal evolution encoded across the frame sequence.

Table 2 summarizes the rendering parameters.

1.6. Sample Video Frames

Figure 1 shows representative video frames for each PDE under the three experimental conditions. Each row corresponds to one PDE, and columns show (a) clean video, (b) 5% Gaussian noise, and (c) 20% spatial coverage with linear interpolation filling masked columns.

Parameter	Value
Colormap	`viridis` (perceptually uniform)
Frame resolution	256×128 px
Frame rate	30 fps
Output format	MP4 (H.264)
Color limits	Fixed per simulation
Spatial encoding	Horizontal axis

Table 2. Video rendering parameters.

PDE	Library Φ	$\lvert\Phi\rvert$	Active terms
KdV	$\{uu_x,\ u_{xxx}\}$	2	Both (α, β)
Burgers	$\{uu_x,\ u_{xx}\}$	2	u_{xx} (ν)
KS	$\{uu_x,\ u_{xx},\ u_{xxxx}\}$	3	All three
Schröd.	$\{u_{xx}\}$	1	u_{xx} (α)
NLS	$\{u_{xx},\ \lvert u\rvert^2 u\}$	2	Both (α, β)

Table 3. Candidate library specification per PDE. Both VIPER and PDE-FIND use identical PDE-specific libraries for fair comparison.

2. Candidate Library Contents and Size

2.1. Per-PDE Library Specification

In our experiments, both VIPER and PDE-FIND use PDE-specific candidate libraries containing 1 to 3 terms per equation. This ensures fair comparison: both methods receive the same structural knowledge about which derivative terms may appear.

Table 3 reports the library terms, size, and ground-truth active terms for each PDE.

2.2. How Libraries Are Used

The two methods employ the candidate library differently.

PDE-FIND constructs the library explicitly. Spatial derivatives $(u_x, u_{xx}, u_{xxx}, u_{xxxx})$ are computed via central finite differences on the observed data, and the temporal derivative u_t is computed via central differences in time. The resulting library matrix $\Theta \in \mathbb{R}^{(n_t \cdot n_x) \times |\Phi|}$ has one column per candidate term. Sequential thresholded least squares (STLS) with threshold $\lambda = 10^{-3}$ (maximum 10 iterations)

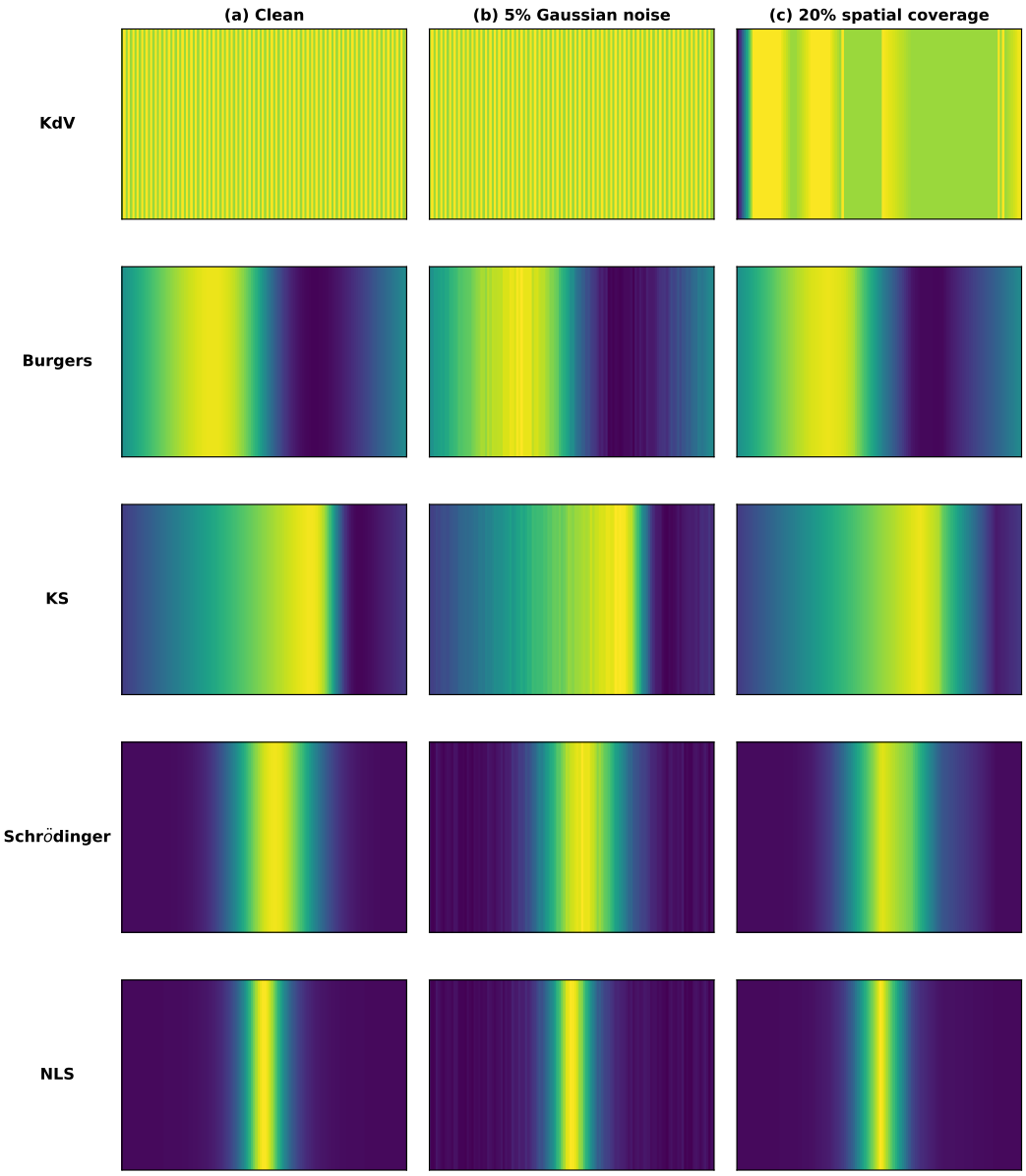

Figure 1. Sample video frames for each PDE under three experimental conditions. (a) Clean video with full spatial coverage. (b) 5% additive Gaussian noise. (c) 20% spatial coverage with masked columns filled via linear interpolation. The viridis colormap encodes field values; spatial coordinate x runs along the horizontal axis. KdV and KS show complex spatiotemporal structure; Schrödinger and NLS exhibit localized wavepacket dynamics; Burgers displays shock formation.

selects active terms and estimates coefficients.

VIPER uses the library implicitly through the differentiable PDE solver. Each PDE's forward integration uses the parameterized form (e.g., $u_t = -\hat{\alpha}\, uu_x - \hat{\beta}\, u_{xxx}$ for KdV), where the coefficients $\hat{\boldsymbol{\theta}}$ are learnable parameters optimized end-to-end via backpropagation through the solver. The ℓ_1 sparsity penalty ($\lambda = 10^{-3}$) encourages zero coefficients for inactive terms. Critically, spatial derivatives are computed spectrally (via FFT) *inside the solver* on the integrated trajectory, never on noisy observed data.

2.3. General (Agnostic) Library

For completeness, a general-purpose library applicable to unknown PDE identification would include the following 9 terms: $\{u,\ u_x,\ u_{xx},\ uu_x,\ u_{xxx},\ u_{xxxx},\ u^2,\ u^3,\ u^2u_x\}$. In the experiments reported in the main paper, we use PDE-specific libraries (1-3 terms) since both methods receive identical structural knowledge for fair comparison.

	PDE-FIND (finite diff. on data)	VIPER (spectral, in solver)
u_x	$\frac{u_{j+1}-u_{j-1}}{2\Delta x}$	$\mathcal{F}^{-1}[i\kappa\,\hat{u}]$
u_{xx}	$\frac{u_{j+1}-2u_j+u_{j-1}}{\Delta x^2}$	$\mathcal{F}^{-1}[-\kappa^2\,\hat{u}]$
u_{xxx}	5-point stencil	$\mathcal{F}^{-1}[-i\kappa^3\,\hat{u}]$
u_{xxxx}	5-point stencil	$\mathcal{F}^{-1}[\kappa^4\,\hat{u}]$

Table 4. Derivative computation comparison. PDE-FIND differentiates noisy data; VIPER differentiates the smooth integrated trajectory inside the solver.

2.4. Derivative Computation Comparison

Table 4 highlights the key methodological difference between VIPER and PDE-FIND in how spatial derivatives are computed. PDE-FIND applies finite-difference stencils directly to the (potentially noisy) observed data, amplifying noise at rate $\mathcal{O}((\sigma/\Delta x)^k)$ for k-th order derivatives. VIPER computes derivatives spectrally inside the differentiable solver on the forward-integrated (smooth) trajectory, avoiding noise amplification entirely.