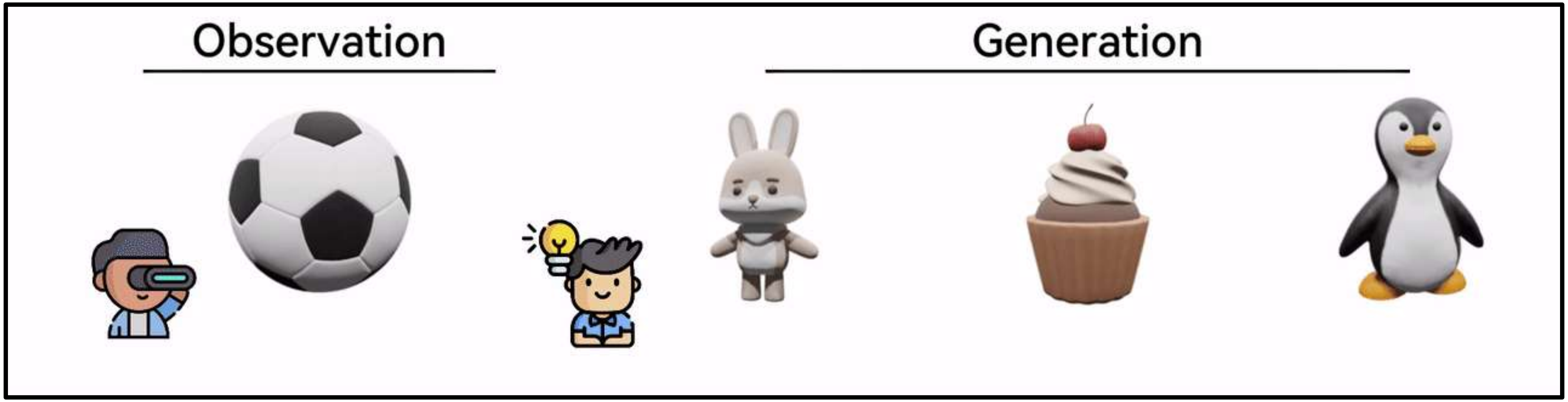
Github

NeuMA: Neural Material Adaptor for Visual Grounding of Intrinsic Dynamics

Junyi Cao[1], Shanyan Guan[2], Yanhao Ge[2], Wei Li[2], Xiaokang Yang[1], Chao Ma[1]

1: Shanghai Jiao Tong University 2: vivo Mobile Communication Co., Ltd.



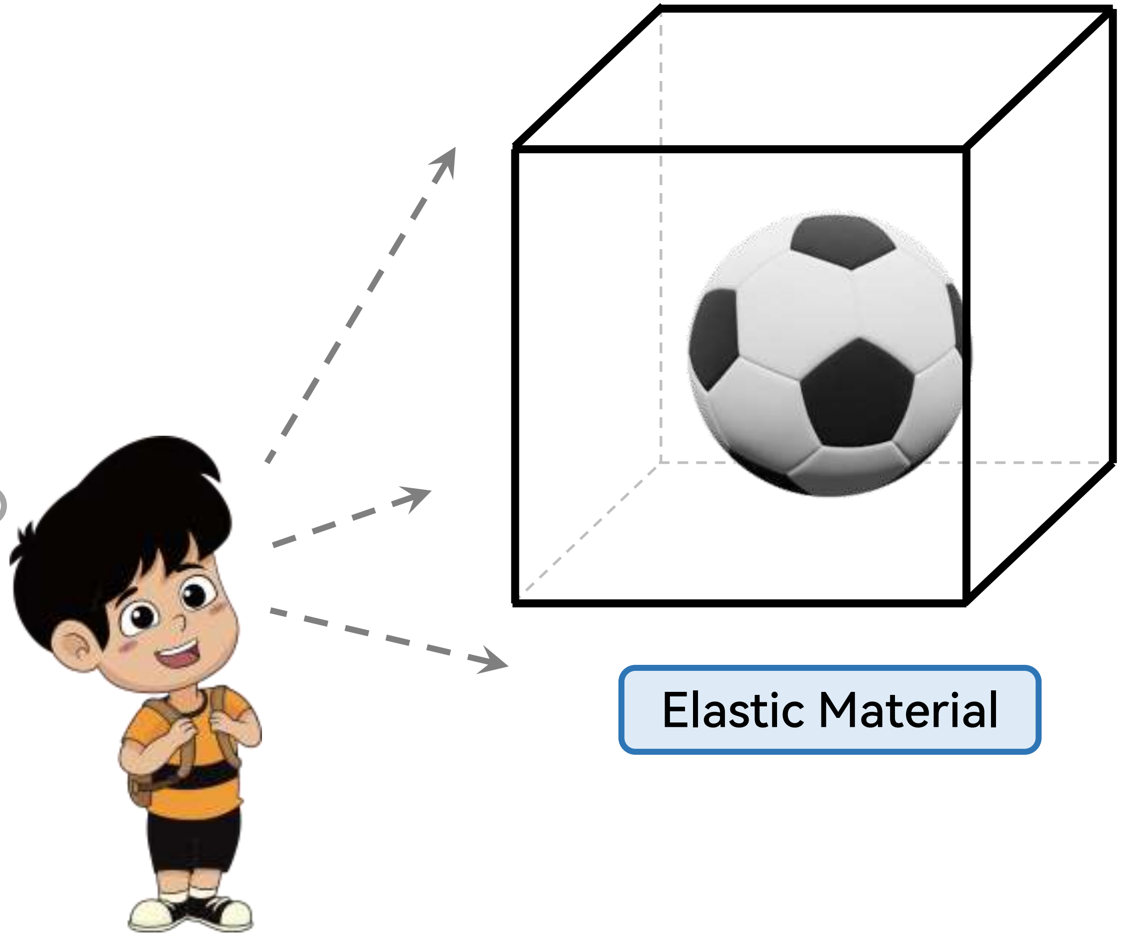
Visual Dynamics Grounding

Rough guess of

- Physical material
- Initial condition (gravity, velocity …)

Imagine similar dynamics for

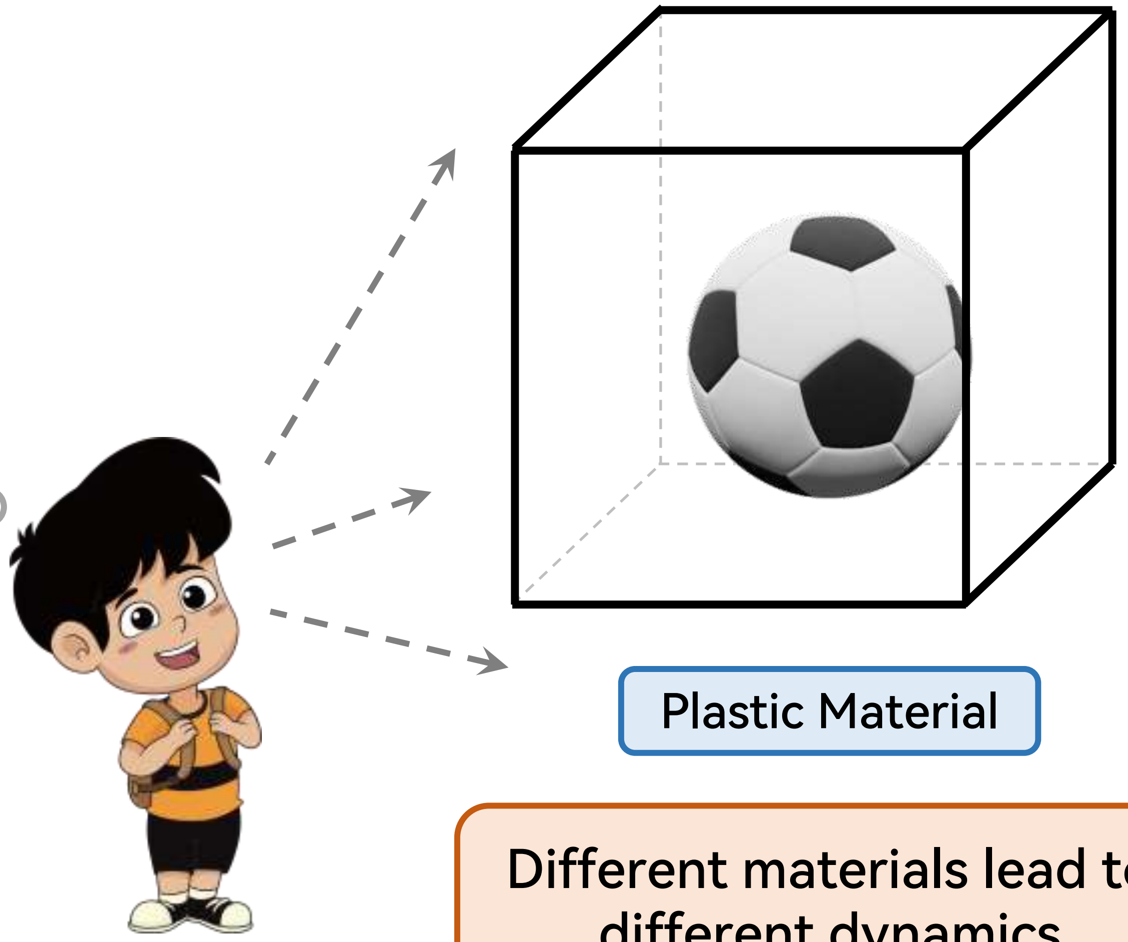
- Novel objects
- Unseen environments



Visual Dynamics Grounding

Rough guess of
- Physical material
- Initial condition (gravity, velocity ...)

Imagine similar dynamics for
- Novel objects
- Unseen environments

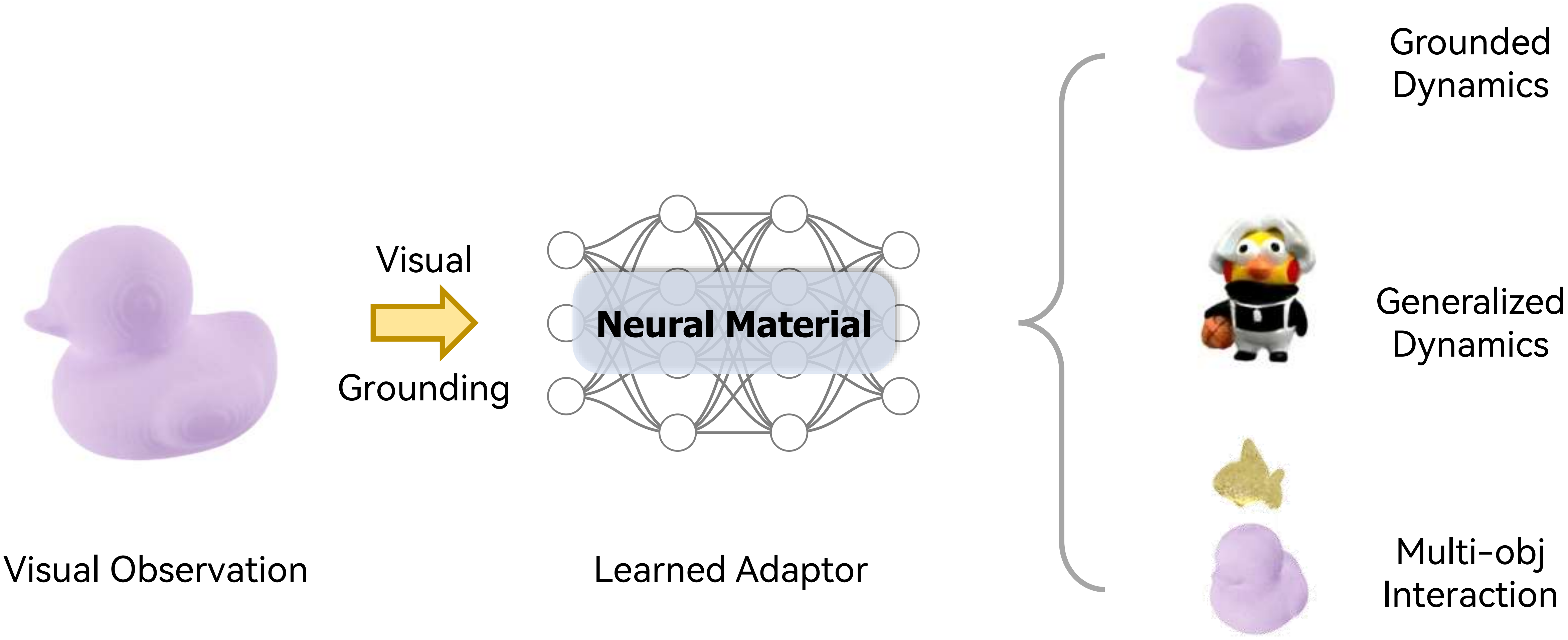


Plastic Material

Different materials lead to different dynamics

Overview

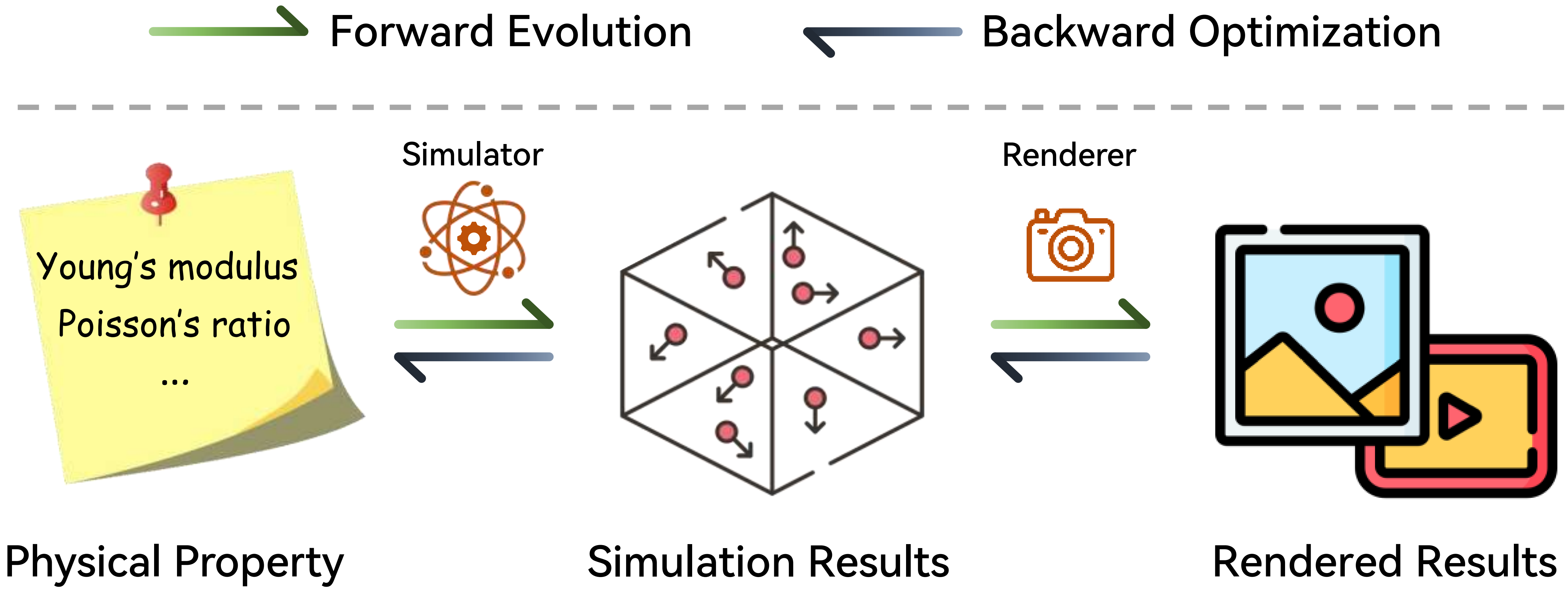
Overview


Visual
Grounding
Neural Material
Visual Observation
Learned Adaptor
Grounded Dynamics
Generalized Dynamics
Multi-obj Interaction

Sections

- Related Work
- Methodology
- Results

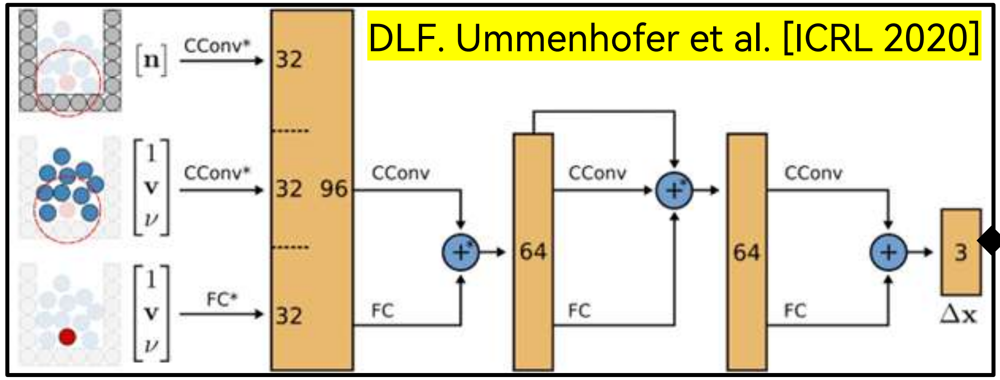
Common Pipeline



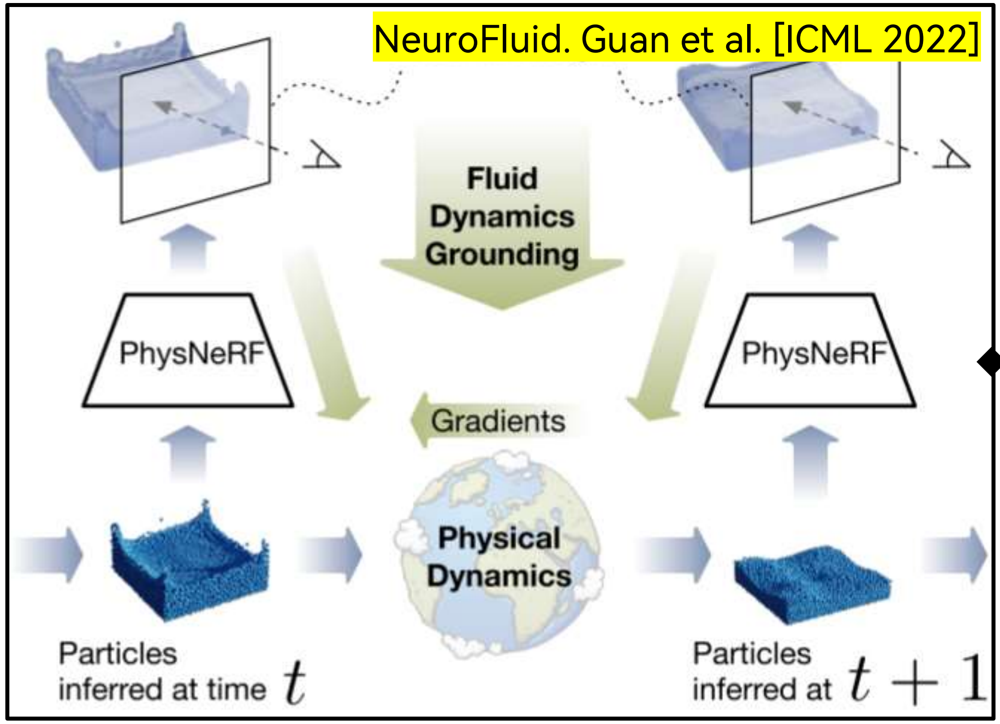
Type of

Black Box
(Implicit)

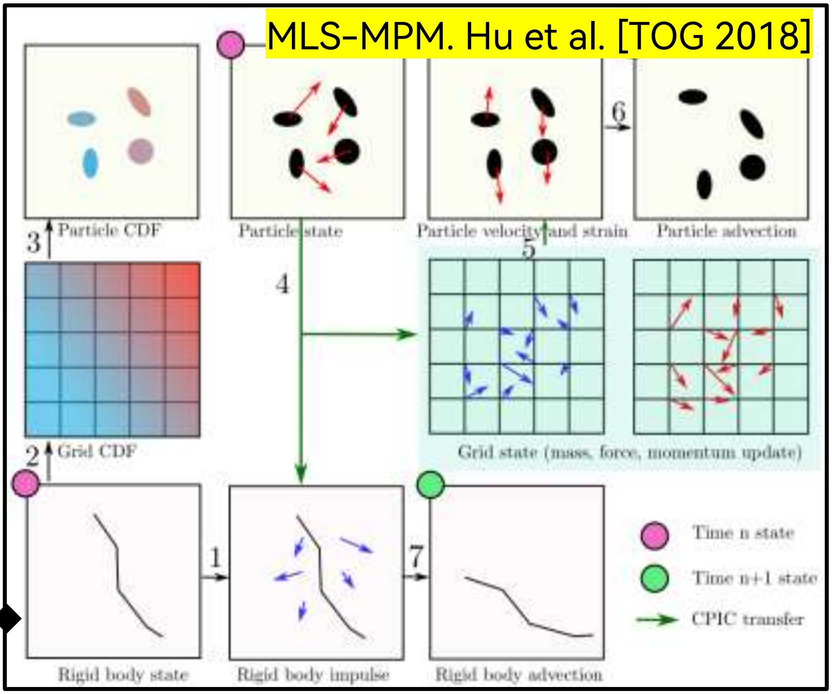
White Box
(Explicit)



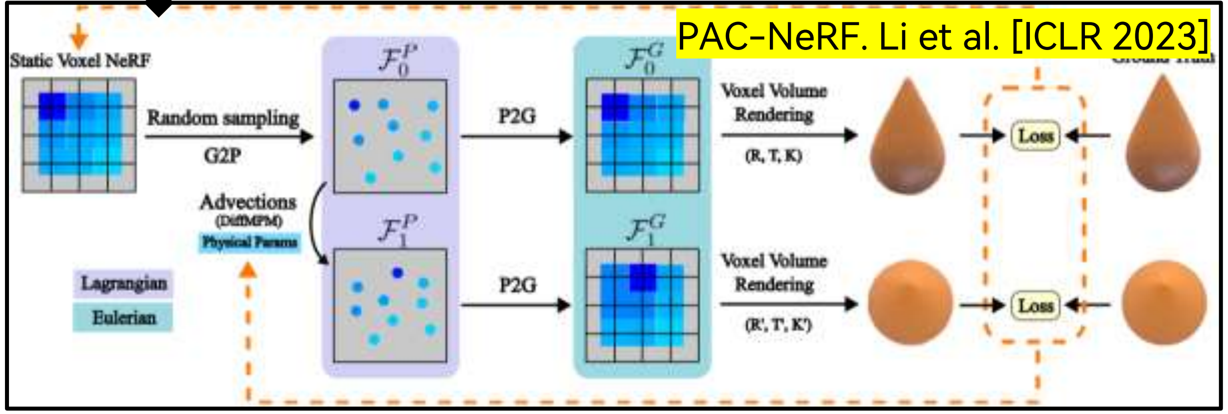
DLF. Ummenhofer et al. [ICLR 2020]



NeuroFluid. Guan et al. [ICML 2022]



MLS-MPM. Hu et al. [TOG 2018]



PAC-NeRF. Li et al. [ICLR 2023]

Type of 

Methods	Aligned to Obs.?	Generalizable?	Efficient Training?	Efficient Inference?
Black Box (Implicit)	✓	✗	✗	✓
White Box (Explicit)	✗	✓	✓	✗

Type of

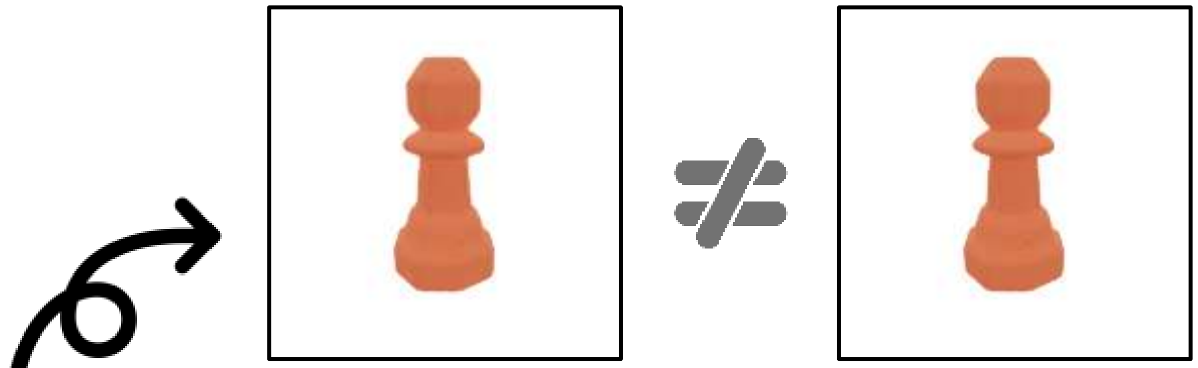
Methods	Aligned to Obs.?	Generalizable?	Efficient Training?	Efficient Inference?
Black Box (Implicit)	✓	✗	✗	✓
White Box (Explicit)	✗	✓	✓	✗

Problem: How to accurately infer the underlying intrinsic dynamics from the visual observations?

Sections

- Background
- Methodology
- Results

Core Idea



Visual
Observation

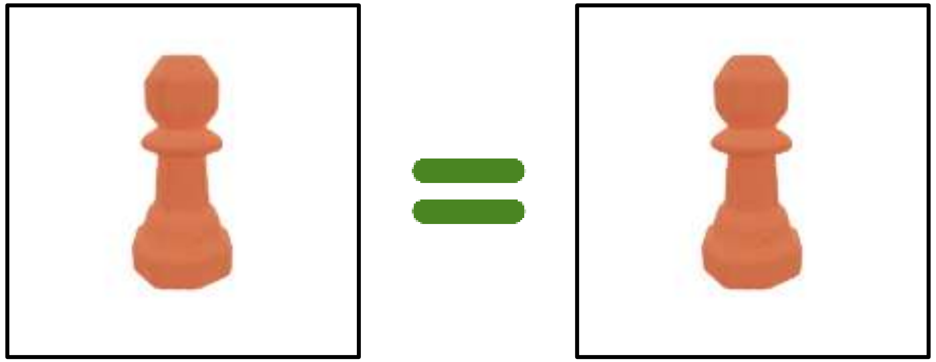
Learning of
Material Models

$$\mathcal{M} := \mathcal{M}_0$$

Physical prior

e.g., von Mises Plasticity

Core Idea



Visual Observation

supervise

Learning of Material Models

$$\mathcal{M}_\theta := \mathcal{M}_0 + \Delta\mathcal{M}_\theta$$

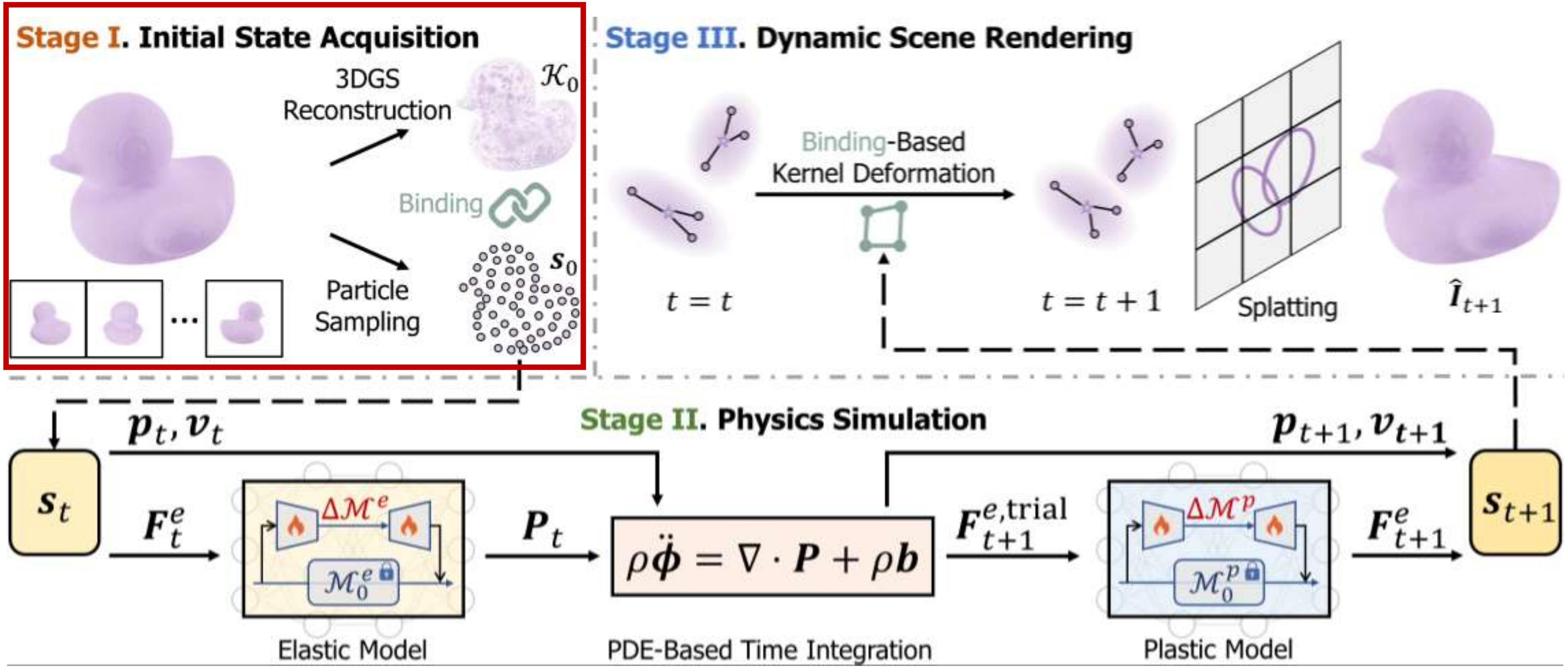
Residual correction

adapted to visual observations

Core Idea

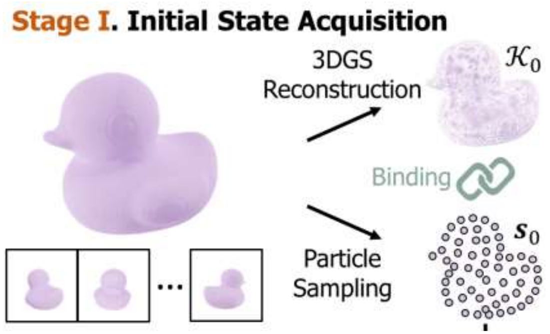
Methods	Aligned to Obs.?	Generalizable?	Efficient Training?	Efficient Inference?
Black Box (Implicit)	✓	✗	✗	✓
White Box (Explicit)	✗	✓	✓	✗
$\mathcal{M}_\theta := \mathcal{M}_0 + \Delta\mathcal{M}_\theta$	✓	✓	✓	✓

Framework



Stage I. Initial State Acquisition
3DGS Reconstruction
$\mathcal{K}_0$
Binding
Particle Sampling
s_0
Stage III. Dynamic Scene Rendering
Binding-Based Kernel Deformation
$t = t$
$t = t + 1$
Splatting
$\hat{I}_{t+1}$
Stage II. Physics Simulation
p_t, v_t
p_{t+1}, v_{t+1}
s_t
s_{t+1}
F_t^e
$\Delta\mathcal{M}^e$
$\mathcal{M}_0^e$
P_t
$\rho\ddot{\phi} = \nabla \cdot P + \rho b$
$F_{t+1}^{e,\text{trial}}$
$\Delta\mathcal{M}^p$
$\mathcal{M}_0^p$
F_{t+1}^e
Elastic Model
PDE-Based Time Integration
Plastic Model

Stage I

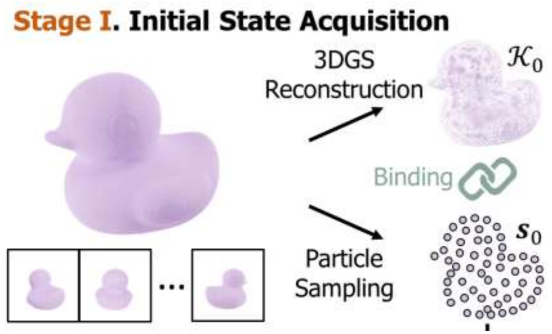


Interested in

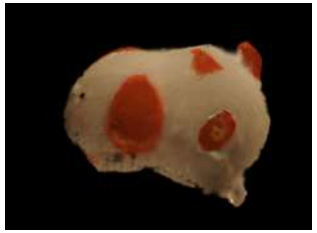
- 3D representation $\mathcal{K}_0$
  - Center $\boldsymbol{x}_0(i)$
  - Opacity $\alpha_0(i)$
  - Cov $\boldsymbol{A}_0(i)$
  - SH $\boldsymbol{c}_0(i)$
- Physical state $\boldsymbol{s}_0$
  - Position $\boldsymbol{p}_0(j)$
  - Velocity $\boldsymbol{v}_0(j)$
  - Def. grad. $\boldsymbol{F}_0^e(j)$

For detailed implementations, please refer to Section 3.1 of our paper.

Stage I



Visual observation



Gaussian Kernels

One-to-one mapping

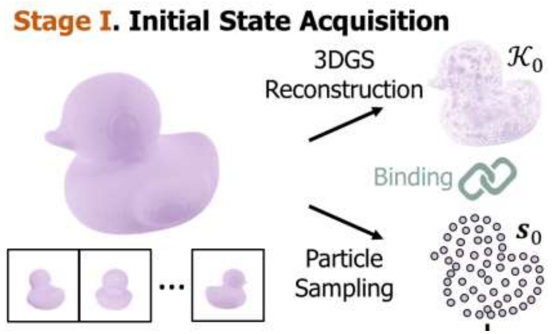
Physical Particles

Better for rendering

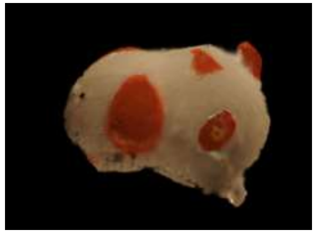
Better for simulation

Property Gap

Stage I



Visual observation



Gaussian Kernels

Particle Binding

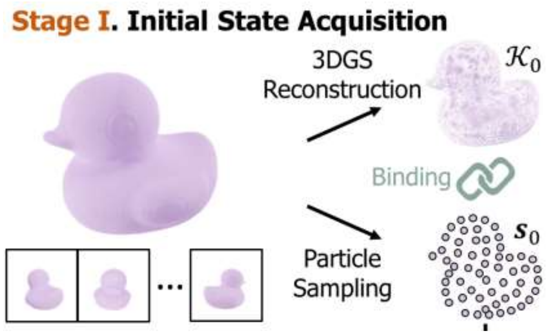
Physical Particles

Better for rendering

Better for simulation

Property Gap

Stage I –



Algorithm 1: Particle Binding

Input: Gaussian centers $\{\boldsymbol{x}(i)\}_{i=1}^{N_K}$, Gaussian covariance $\{\boldsymbol{A}(i)\}_{i=1}^{N_K}$, particle positions $\{\boldsymbol{p}_0(j)\}_{j=1}^{N_P}$, confidence threshold τ

Output: Binding matrix $\mathcal{B}$

$\mathcal{B} = \texttt{zeros}(N_K, N_P)$;

for $i \leftarrow 1$ **to** N_K **do**

  for $j \leftarrow 1$ **to** N_P **do**

    `// Mahalanobis distance`

    $d_{\mathrm{m}} = \left(\boldsymbol{p}_0(j) - \boldsymbol{x}(i)\right)^\top \boldsymbol{A}(i)^{-1} \left(\boldsymbol{p}_0(j) - \boldsymbol{x}(i)\right)$;

    `// Check the threshold`

    if $d_{\mathrm{m}} \leq \texttt{chi2}(\tau)$ **then**

      $\mathcal{B}(i, j) = 1$;

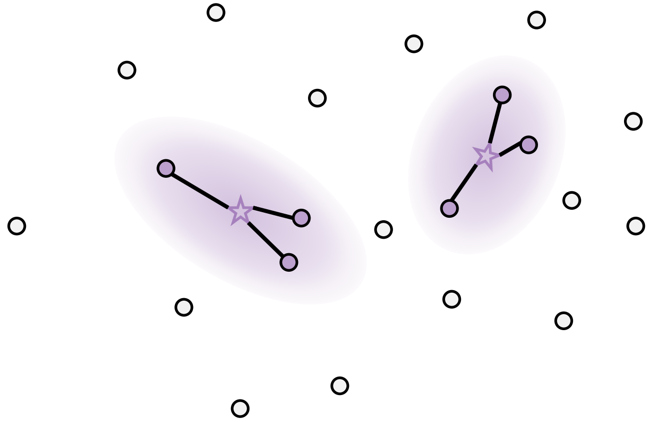
    end

  end

  `// Normalize for each row`

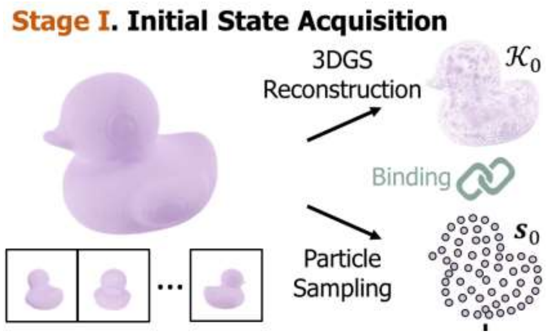
  $\mathcal{B}(i, :) = \mathcal{B}(i, :)/(\texttt{sum}(\mathcal{B}(I, :)))$;

end



$\mathcal{B}$ ~ Binding matrix

Stage I –



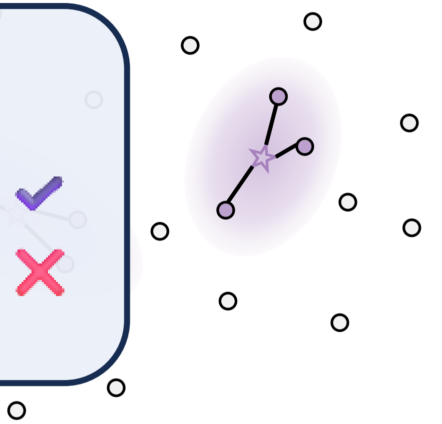
Algorithm 1: Particle Binding

Input: Gaussian centers $\{x(i)\}_{i=1}^{N_K}$, Gaussian covariance $\{A(i)\}_{i=1}^{N_K}$, particle positions $\{p_0(j)\}_{j=1}^{N_P}$, confid

Output: Binding matrix $\mathcal{B}$

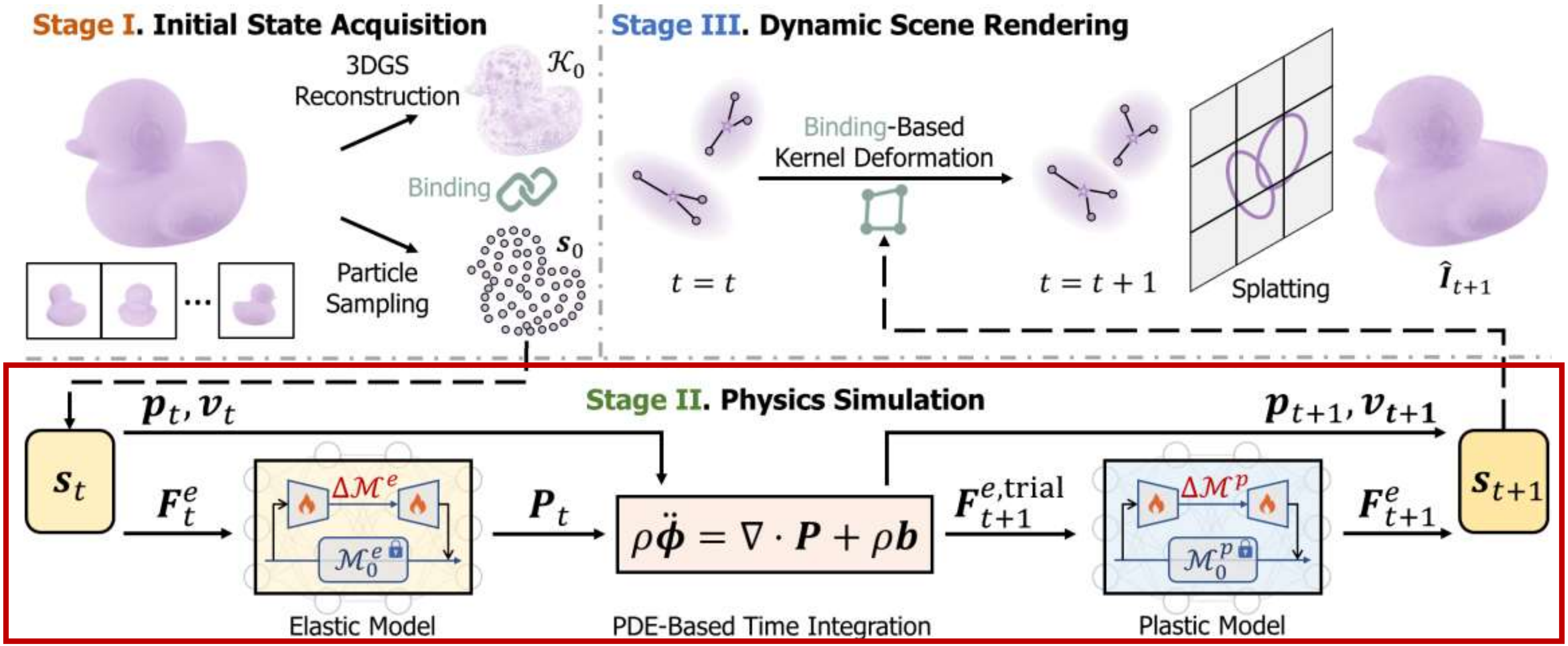
```
B = zeros(N_K, N_P);
for i ← 1 to N_K do
    for j ← 1 to N_P do
        // Mahalanobis dista
        d_m = (p_0(j) − x(i))^T A(
        // Check the thresho
        if d_m ≤ chi2(τ) then
            B(i, j) = 1;
        end
    end
    // Normalize for each row
    B(i, :) = B(i, :)/(sum(B(I, :)));
end
```

	Average C.D.
NeuMA **(Ours)**	1.31×10^{-4}
NeuMA *w/o* Bind	6.60×10^{-4}



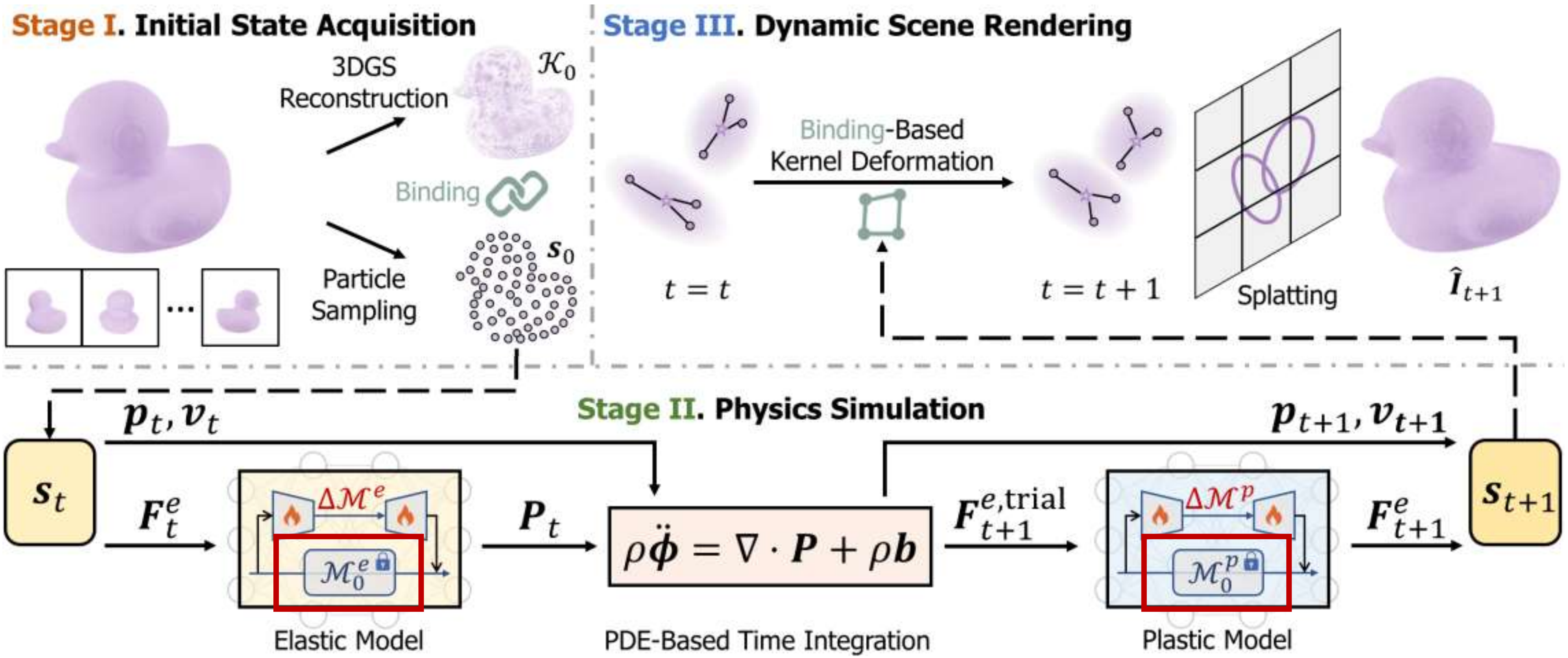
$\mathcal{B}$ ~ Binding matrix

Stage II



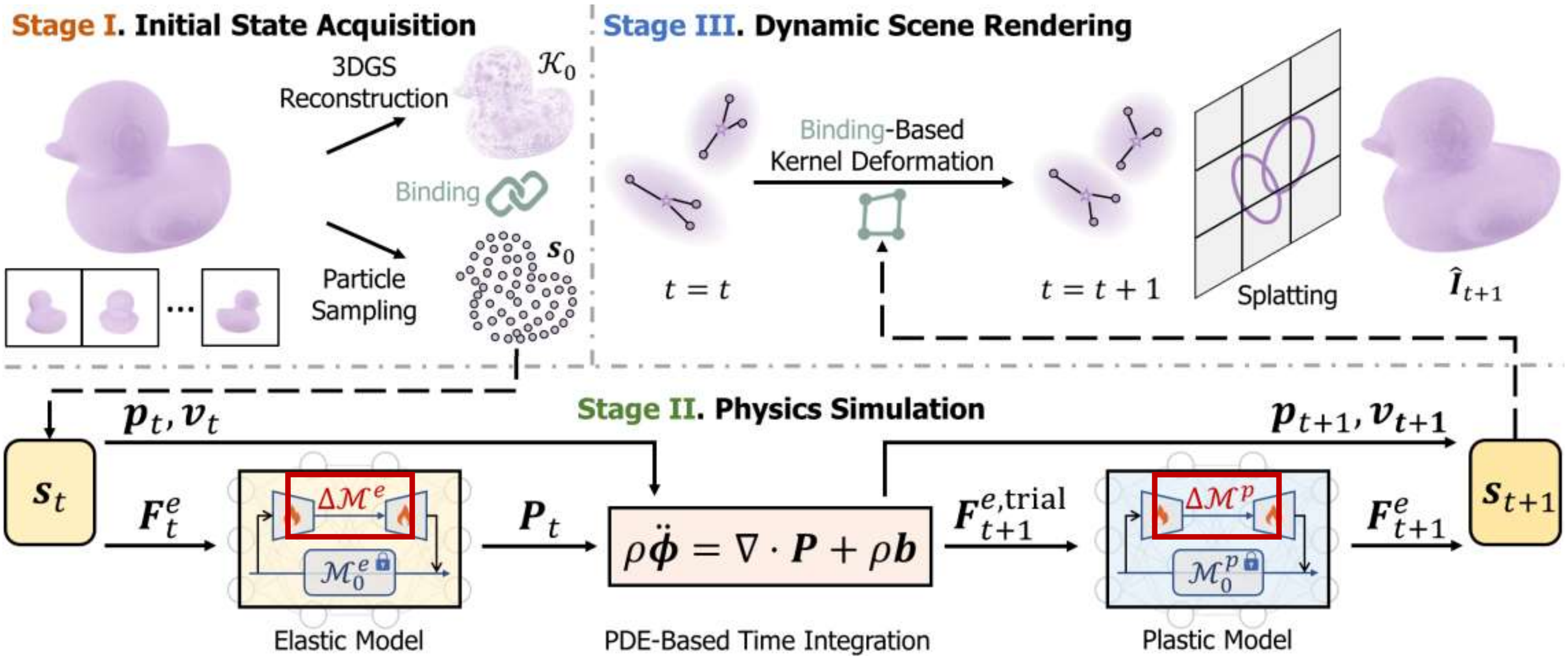
Stage I. Initial State Acquisition
3DGS Reconstruction
$\mathcal{K}_0$
Binding
Particle Sampling
s_0
Stage III. Dynamic Scene Rendering
Binding-Based Kernel Deformation
$t = t$
$t = t + 1$
Splatting
$\hat{I}_{t+1}$
Stage II. Physics Simulation
p_t, v_t
p_{t+1}, v_{t+1}
s_t
F_t^e
$\Delta\mathcal{M}^e$
$\mathcal{M}_0^e$
Elastic Model
P_t
$\rho\ddot{\phi} = \nabla \cdot P + \rho b$
PDE-Based Time Integration
$F_{t+1}^{e,\mathrm{trial}}$
$\Delta\mathcal{M}^p$
$\mathcal{M}_0^p$
Plastic Model
F_{t+1}^e
s_{t+1}

Stage II



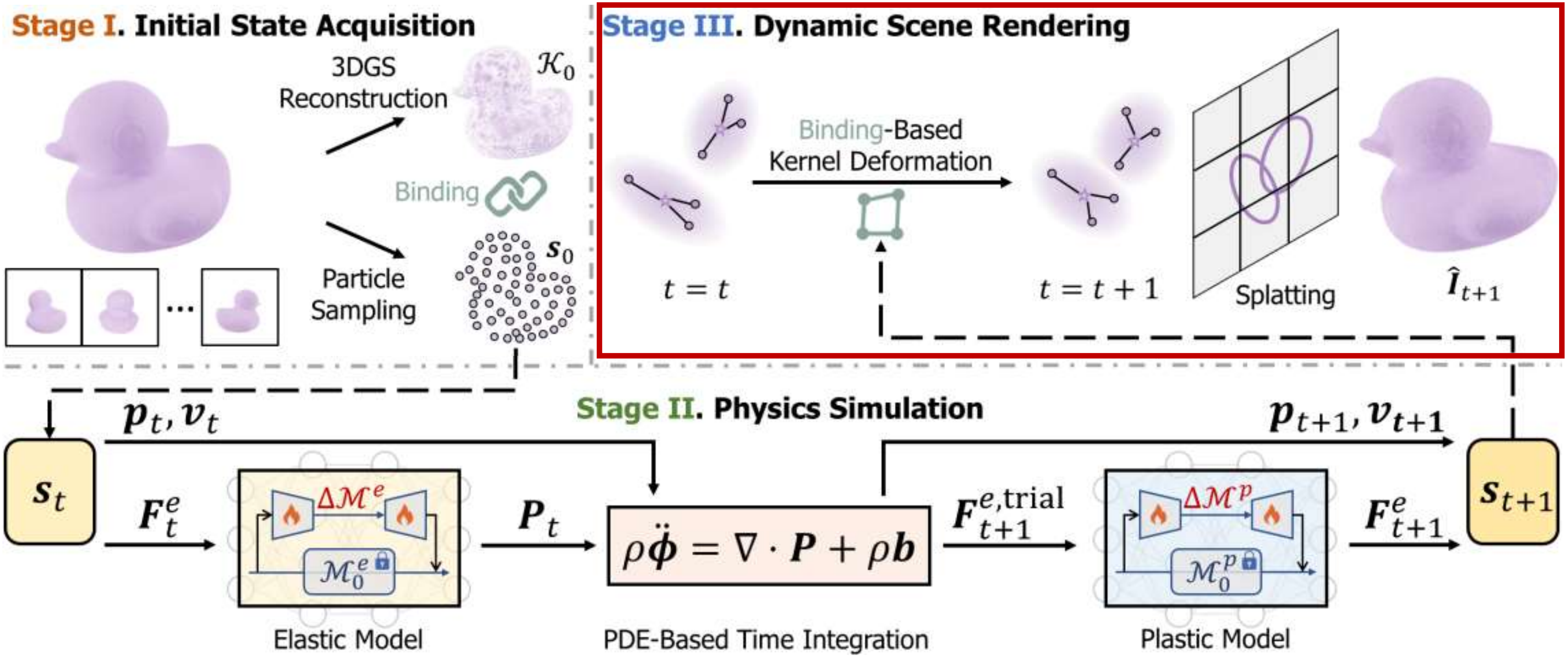
Neural Constitutive Laws. Ma et al. [ICML 2023]

Stage II



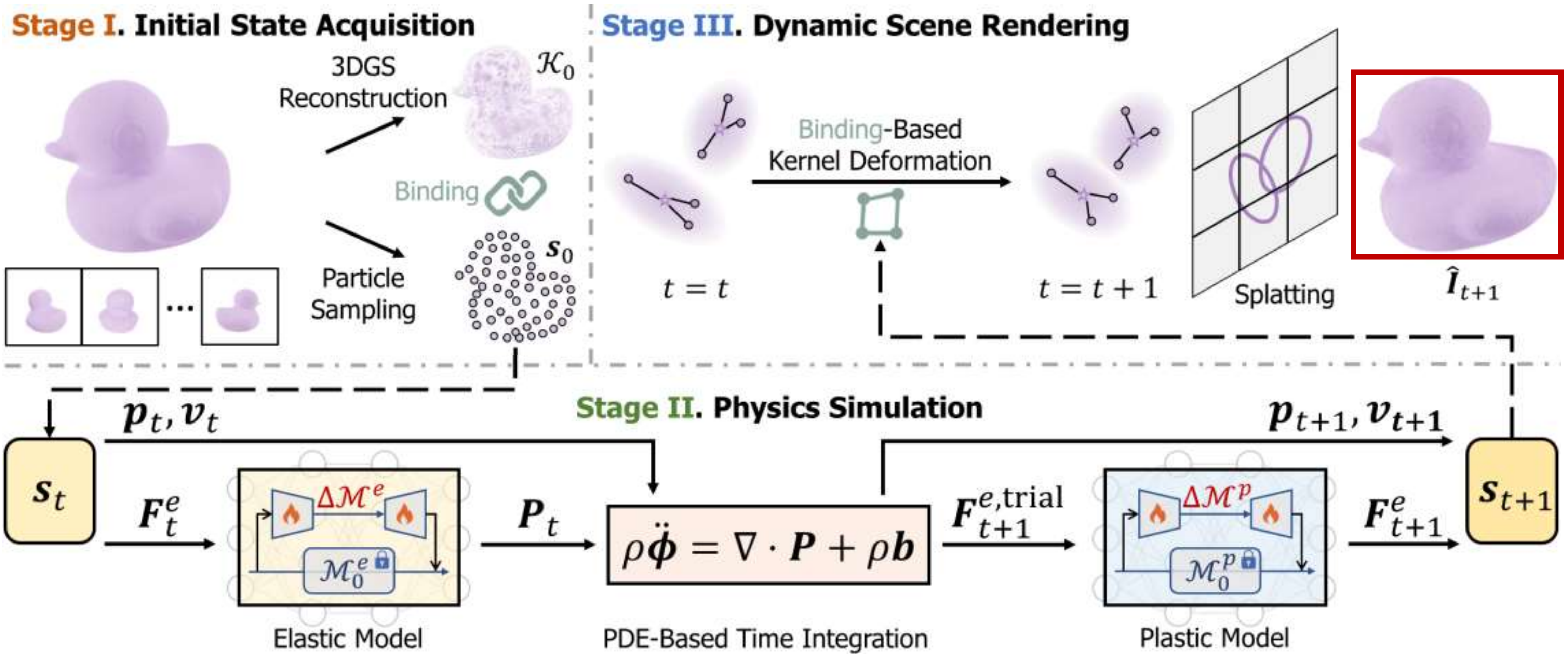
Low-rank Adaptation. Hu et al. [ICLR 2021]

Stage III



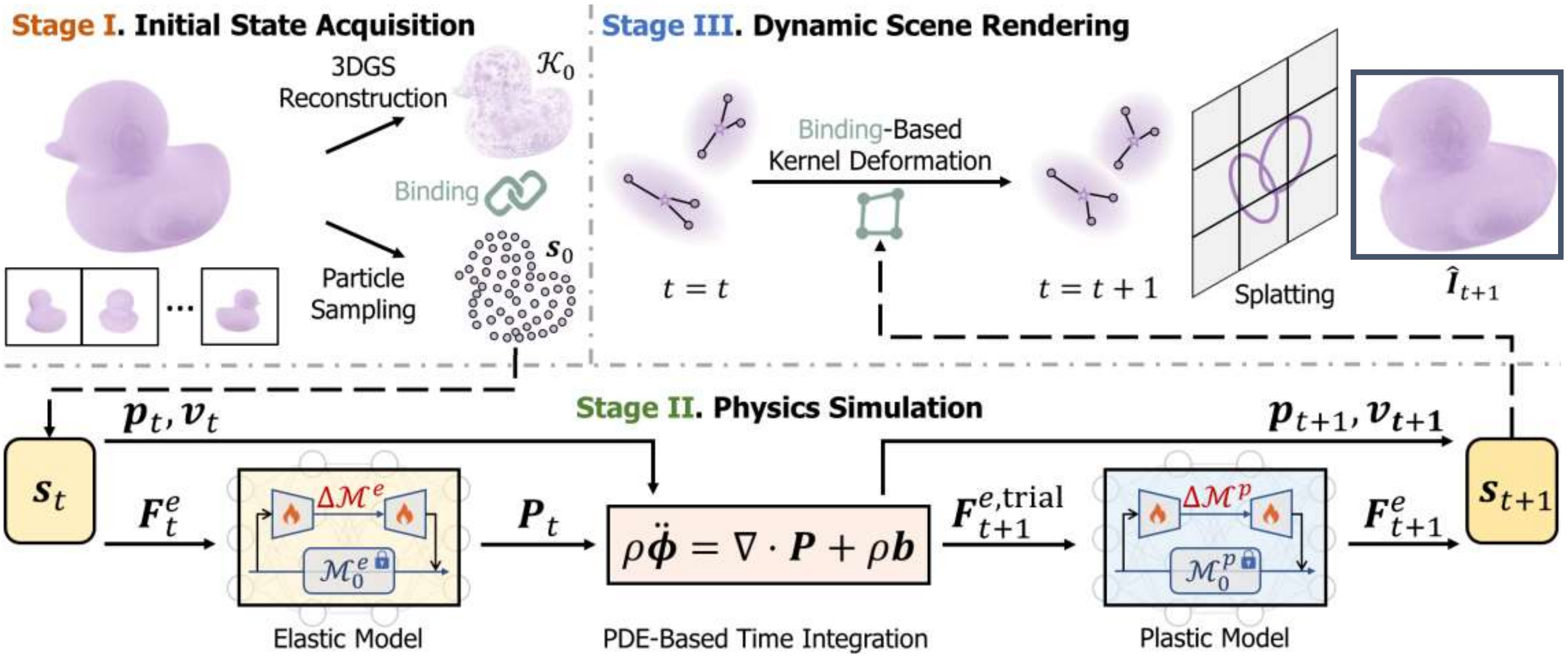
For detailed implementations, please refer to Section 3.3 of our paper.

Stage III



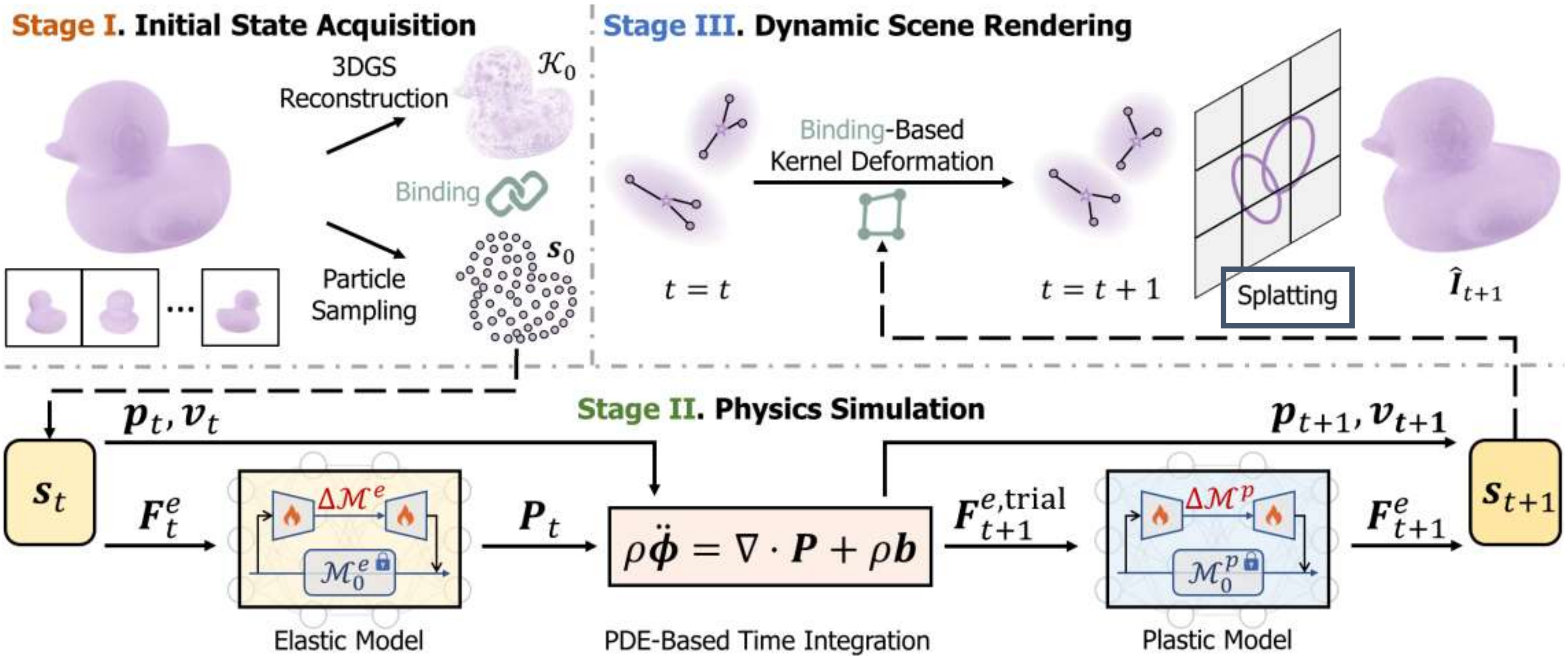
For detailed implementations, please refer to Section 3.3 of our paper.

Backpropagation

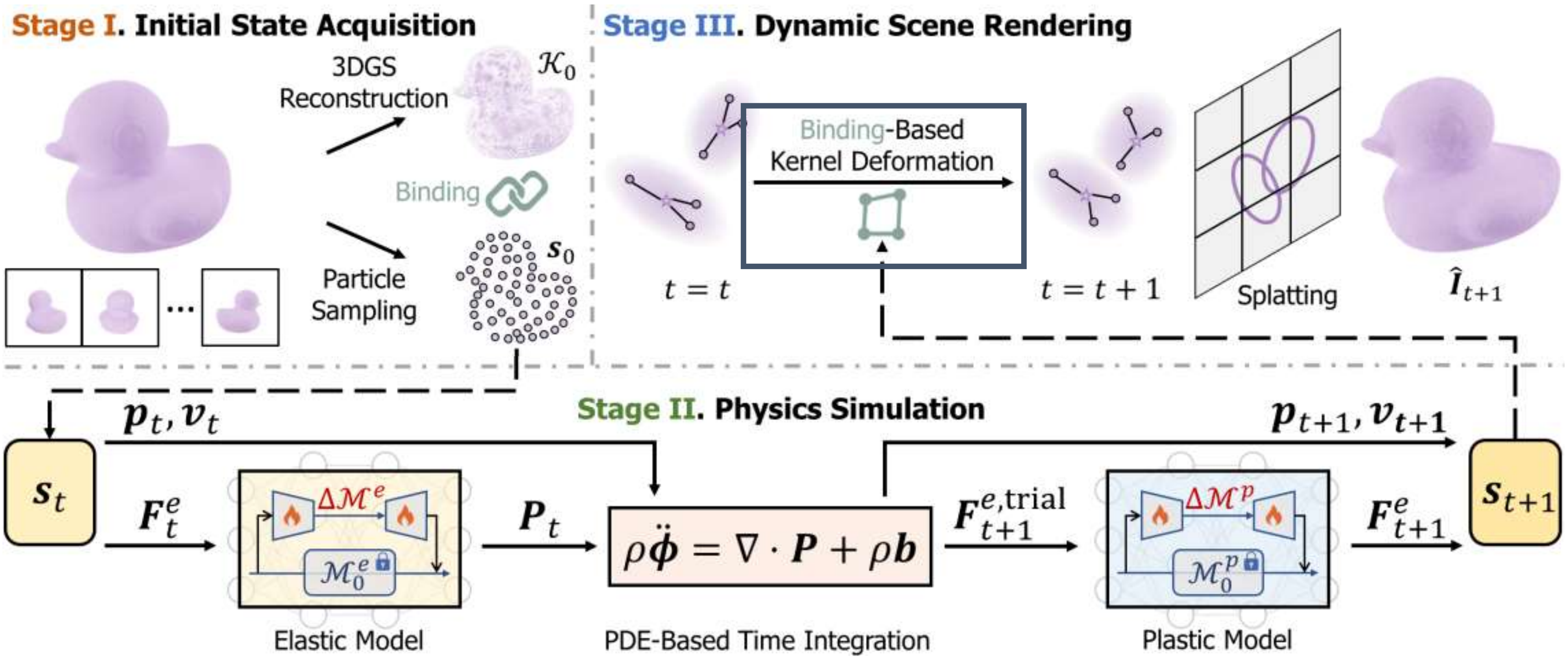


For detailed implementations, please refer to Section 3.3 of our paper.

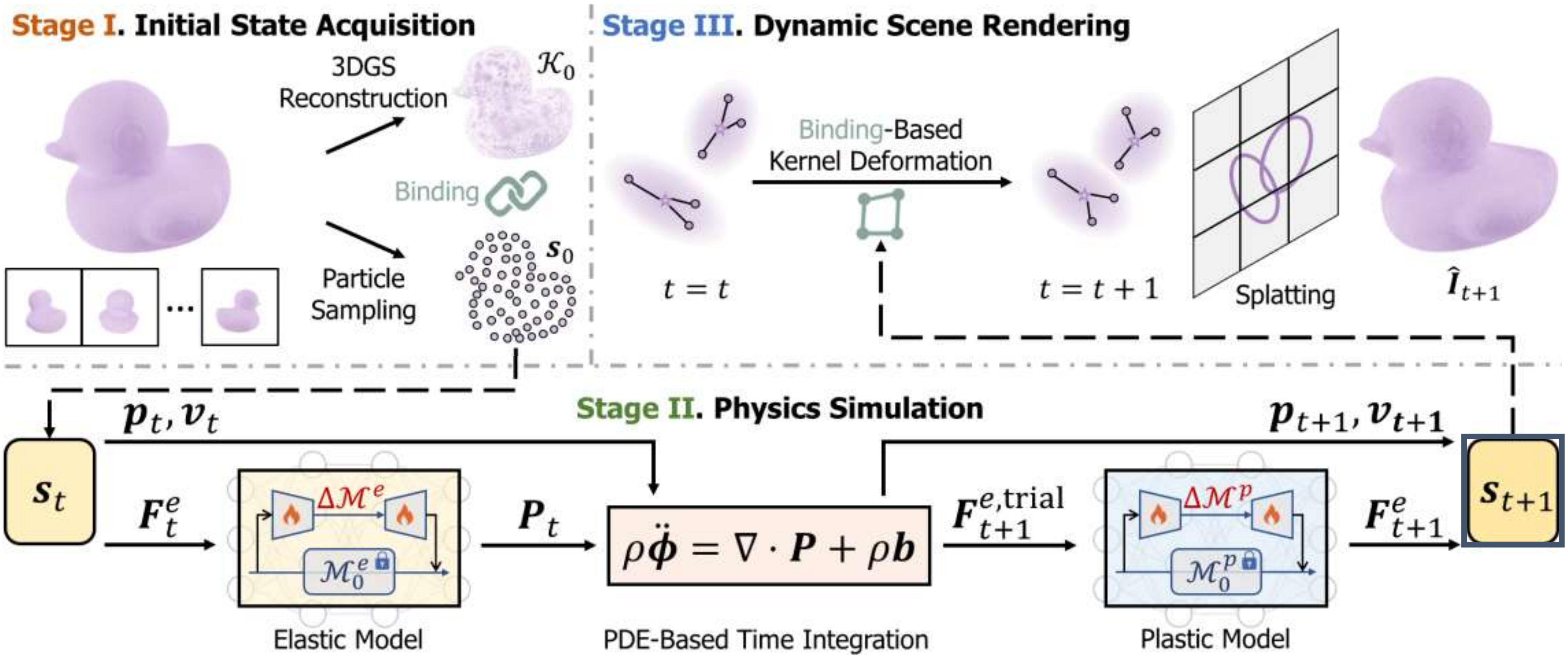
Backpropagation



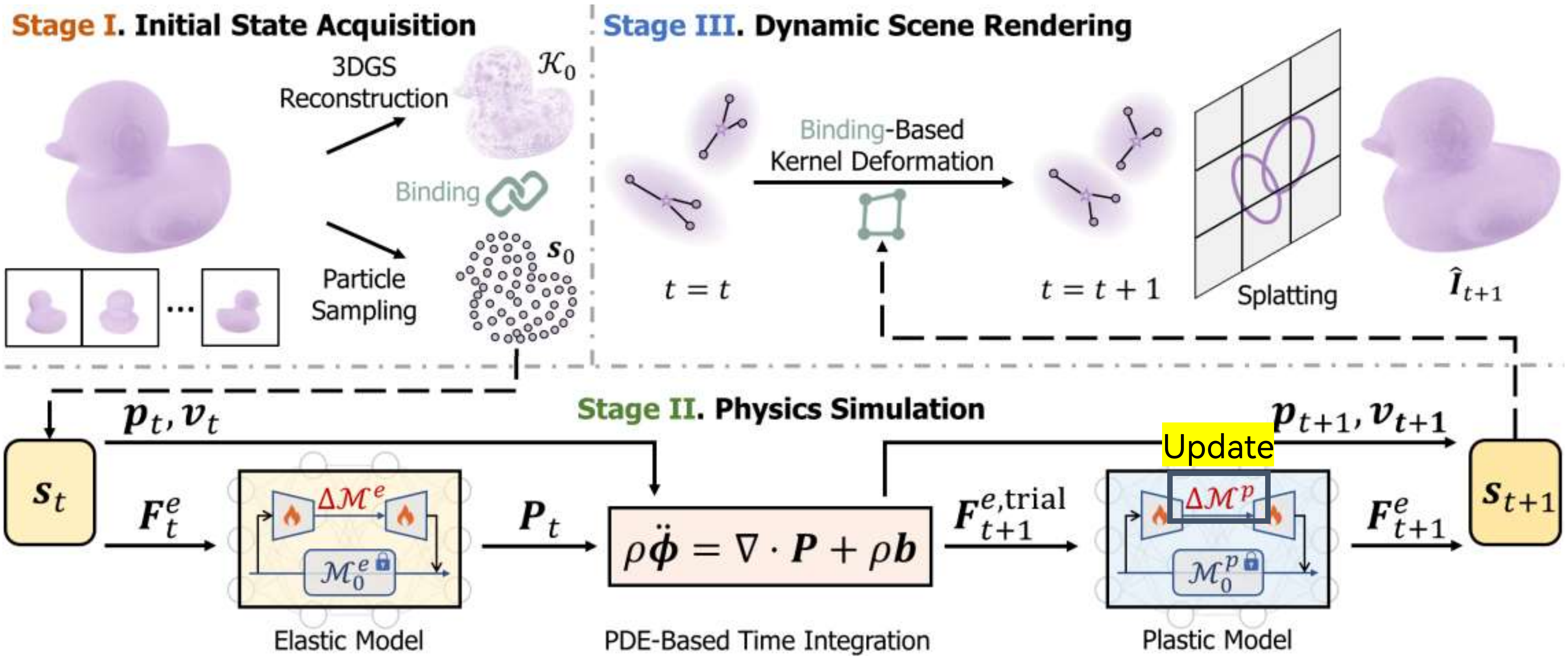
Backpropagation



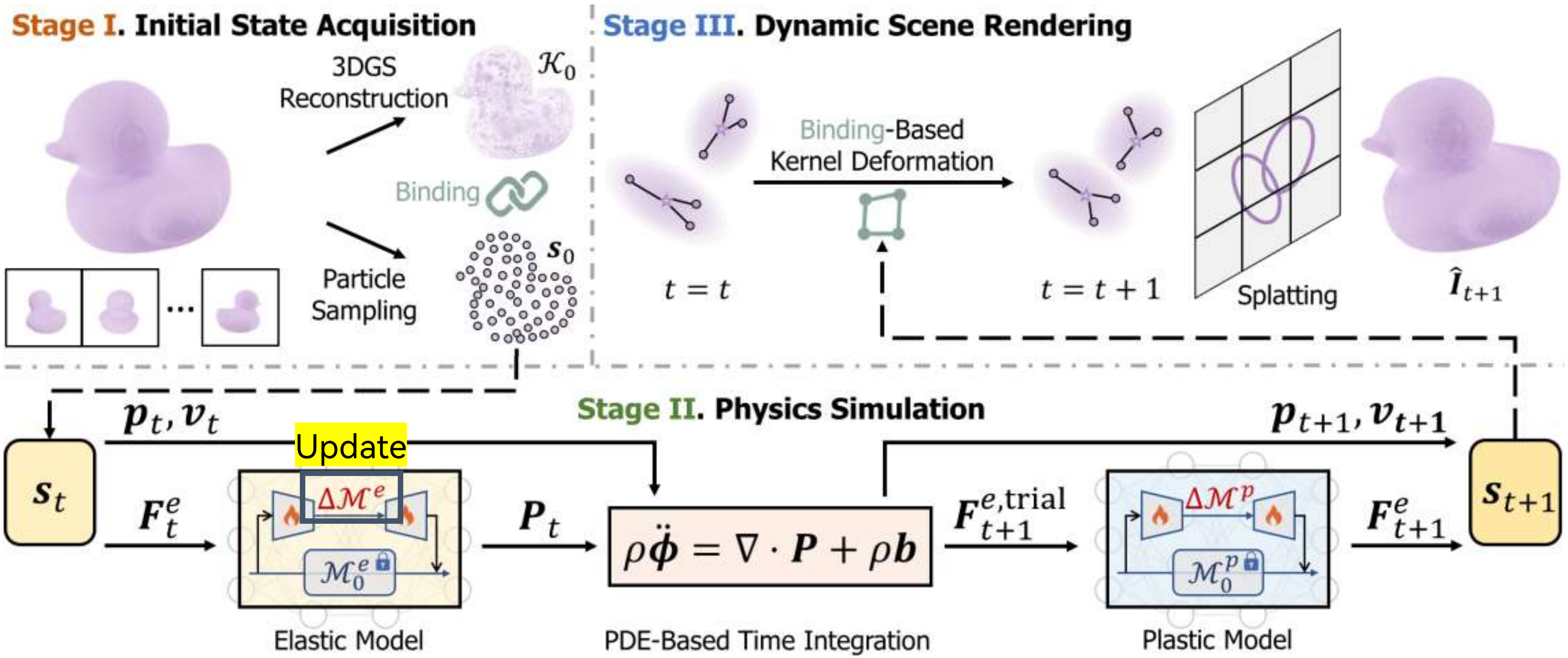
Backpropagation



Backpropagation



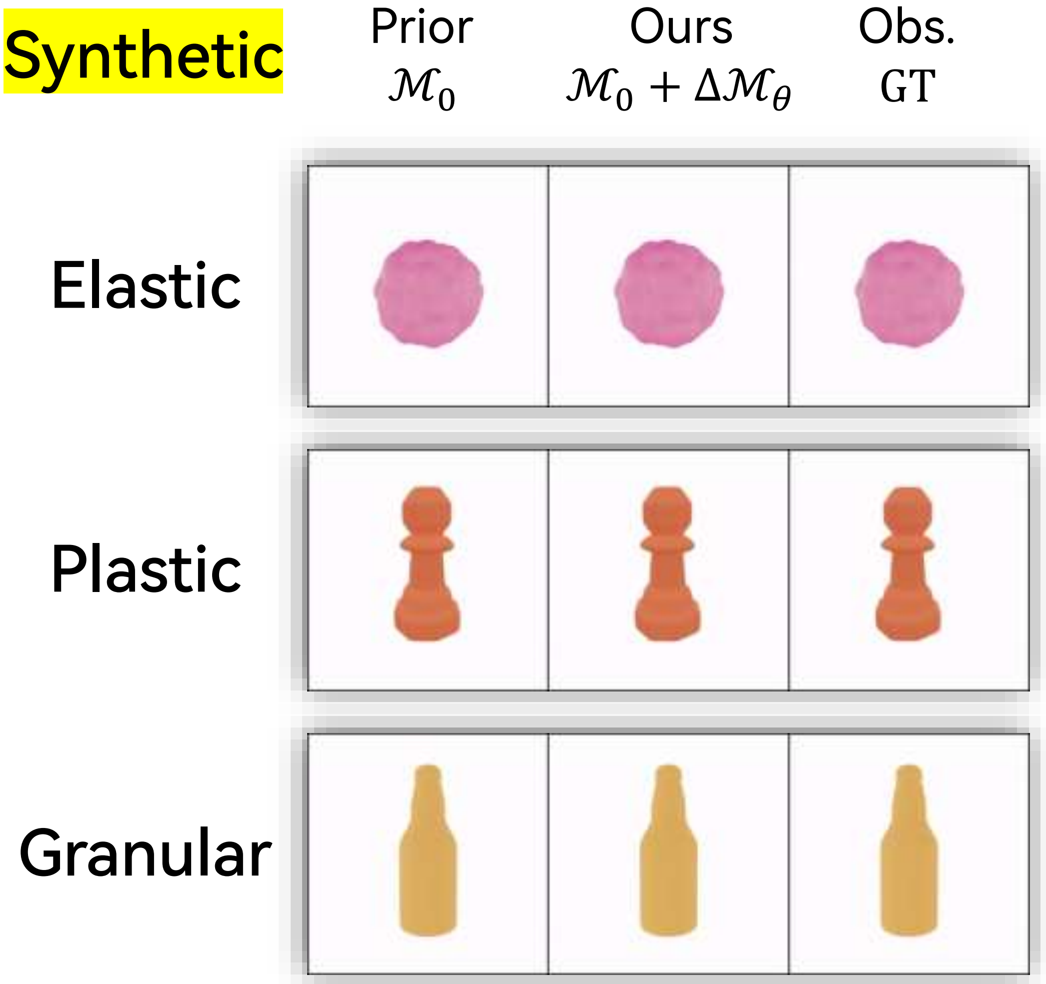
Backpropagation

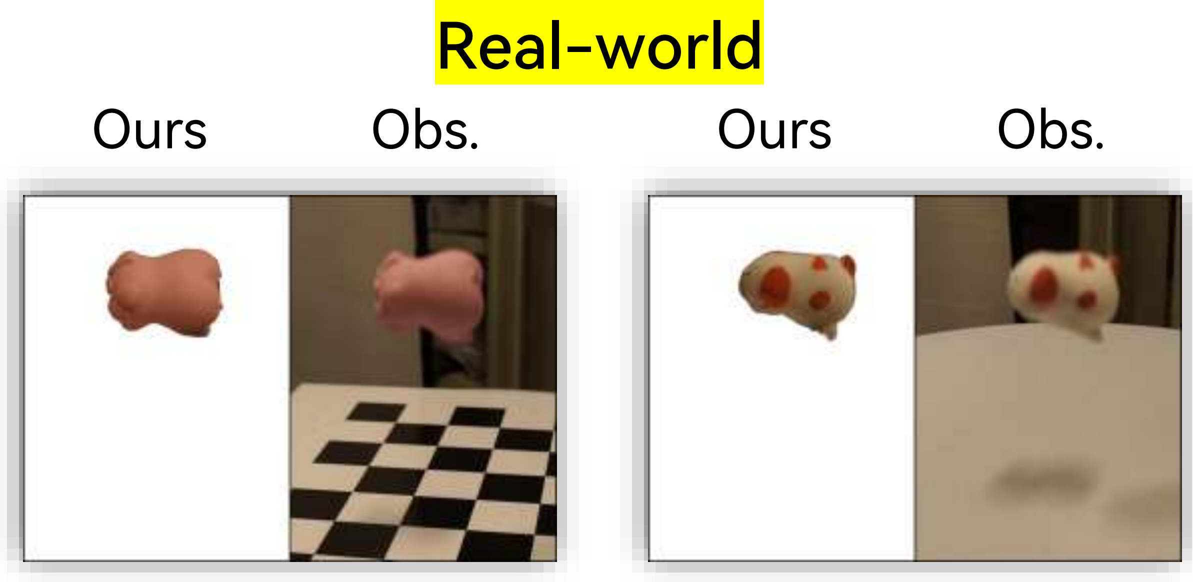


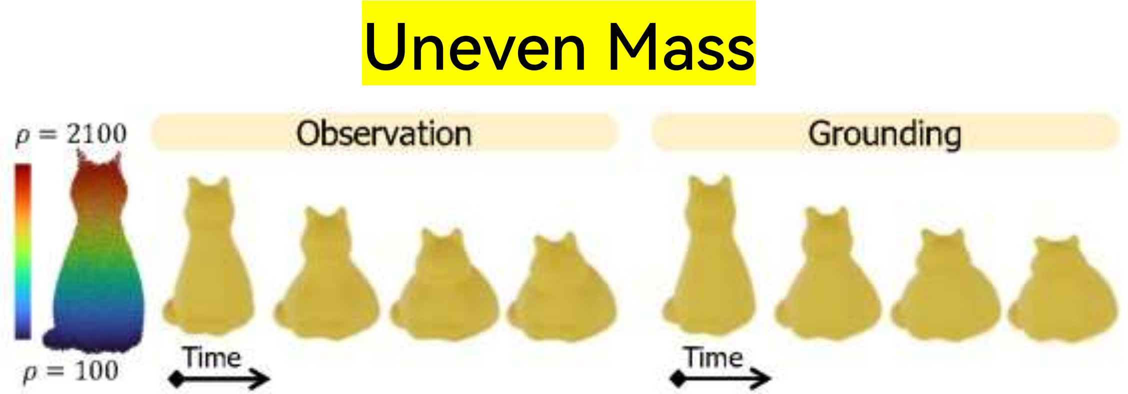
Sections

- Background
- Methodology
- Results

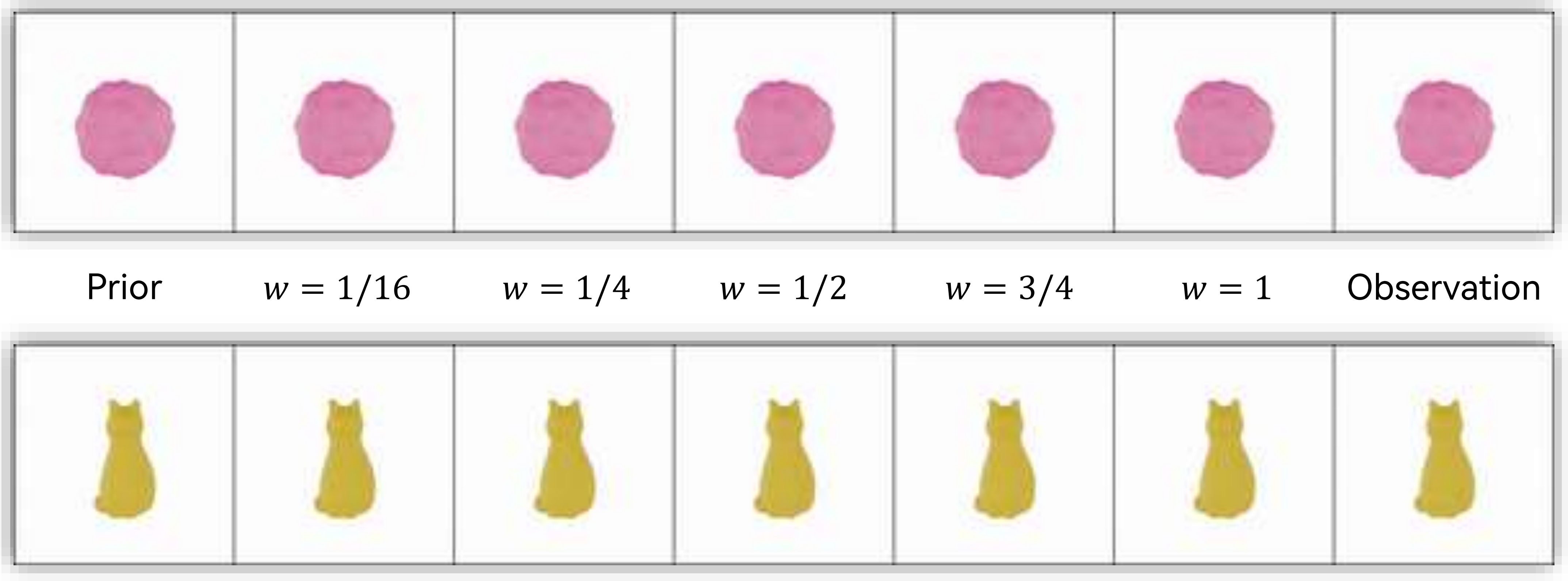
Grounding Results





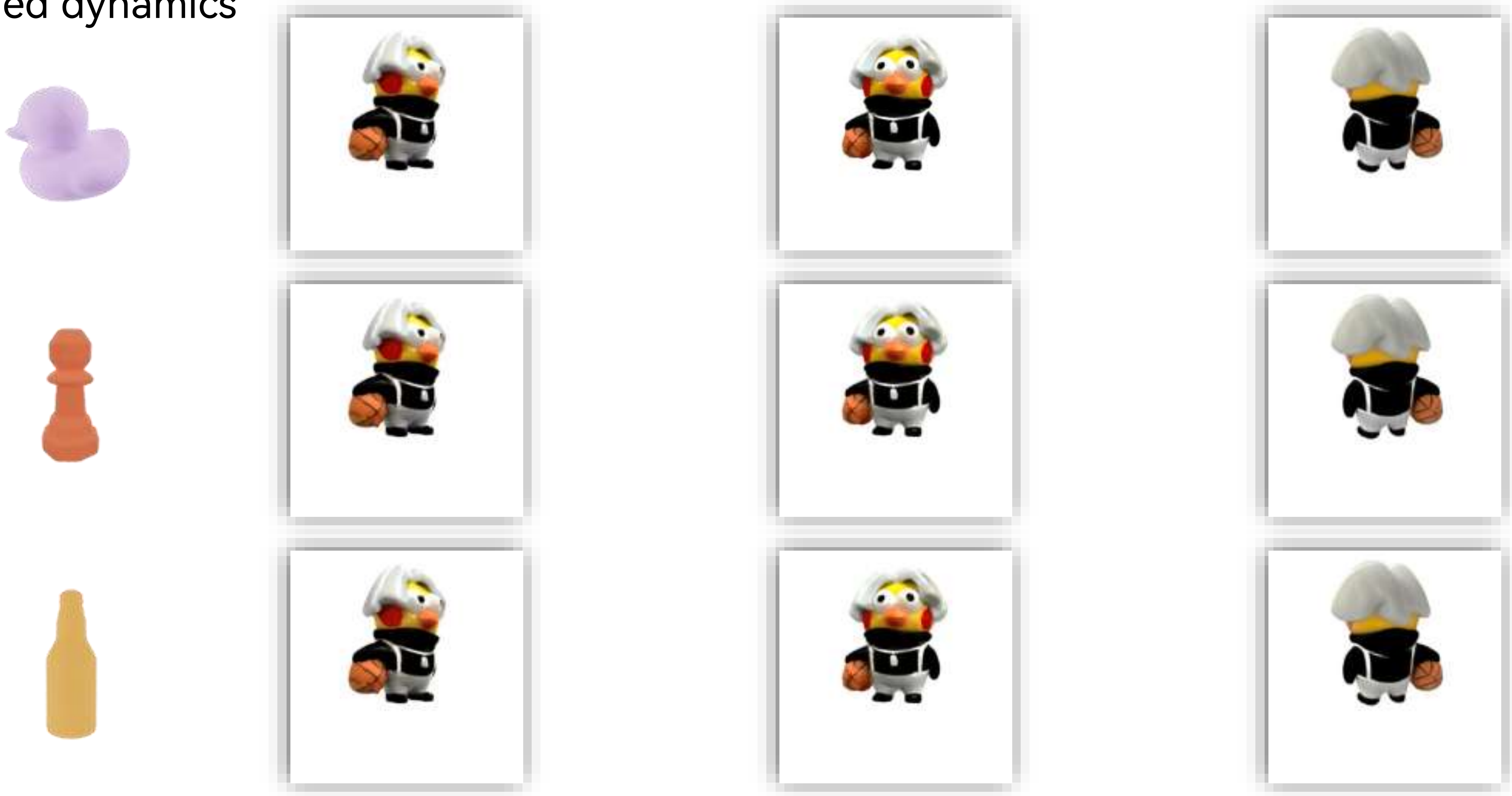


Interpolation Results



Generation Results

Applied dynamics



Thanks for watching!

Github

We have released our code and data :)