CIS 6280 · FALL 2026

# WORLD MODELS

## LECTURE 16

Learning 3D and 4D World Models

**Jiatao Gu**

University of Pennsylvania · November 5, 2026

# Course announcements

| | |
|---|---|
| **Assignment 3** | Release date: November 2<br>Due: November 18 (unchanged) |
| **Final project** | Data released<br>Instructions and updates on Canvas |
| **Office hours** | Thursday, 4:30–5:30 PM<br>AGH 423 |

# Tracking the same surface point




An observed dynamic video and its reconstructed geometry and point motion.

### Choose one surface point

Follow that same point through the video.

### Its image location can change

The object may move, the camera may move, or both.

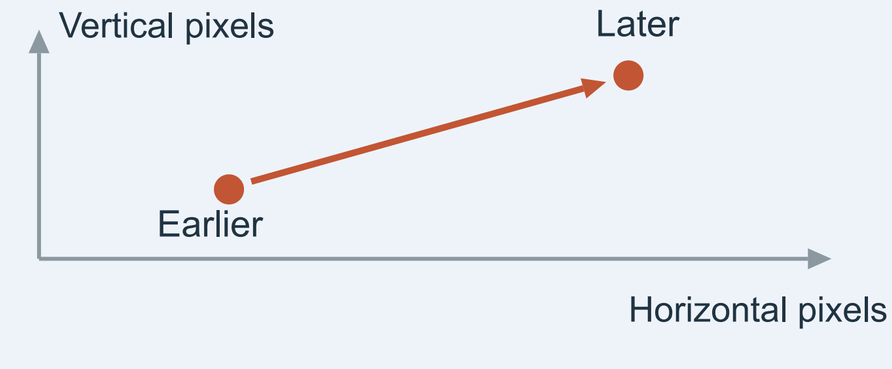
**Which surface point does one colored track follow?**

**Tracking preserves point identity while its location changes.**

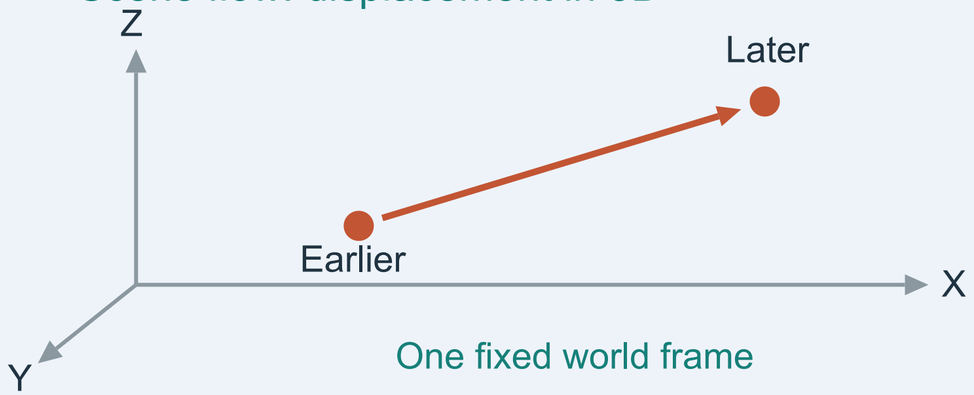
Feng et al., St4RTrack (2025)

# Optical flow and scene flow

Optical flow: displacement in the image




Scene flow: displacement in 3D



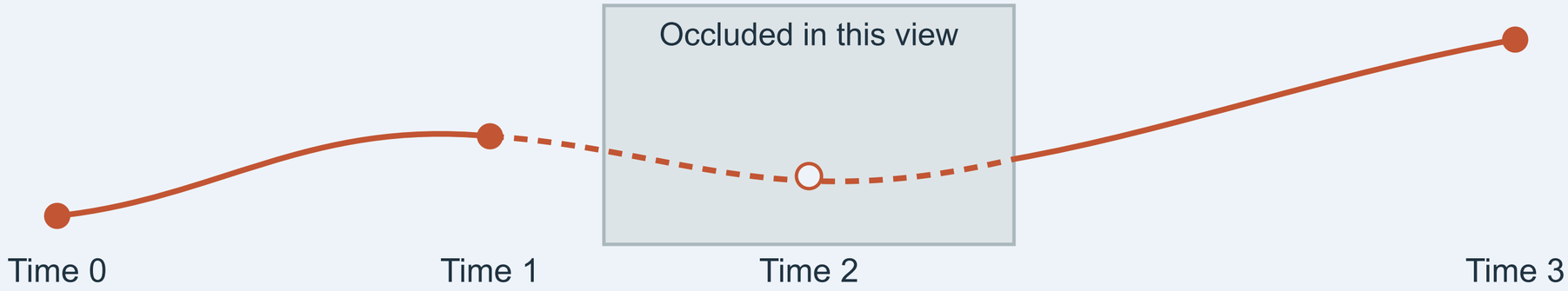

**2D:** where the point appears in the two images.

**3D:** where the same point is in the world at the two times.

**A moving camera can create image flow while a stationary world point has zero scene flow.**

# Point trajectories and visibility

One selected surface point, in one fixed reference frame




## Trajectory positions

$$X \in \mathbb{R}^{N \times T \times 3}$$

N points × T times × 3 coordinates.

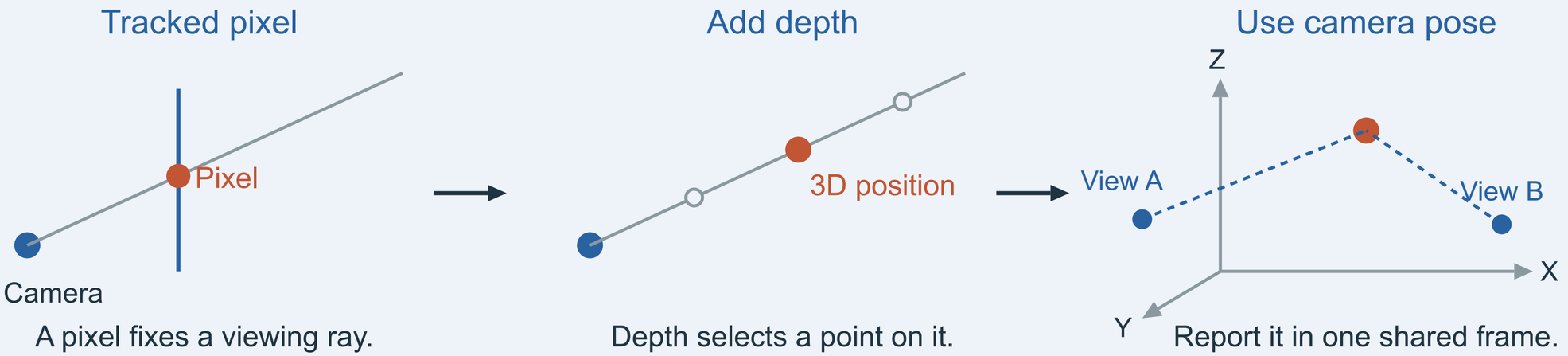
## Visibility is a separate output

Can this camera see the point at that time?

Dashed path: estimated during occlusion.

**The point index keeps its identity. A hidden position is inferred, with uncertainty.**

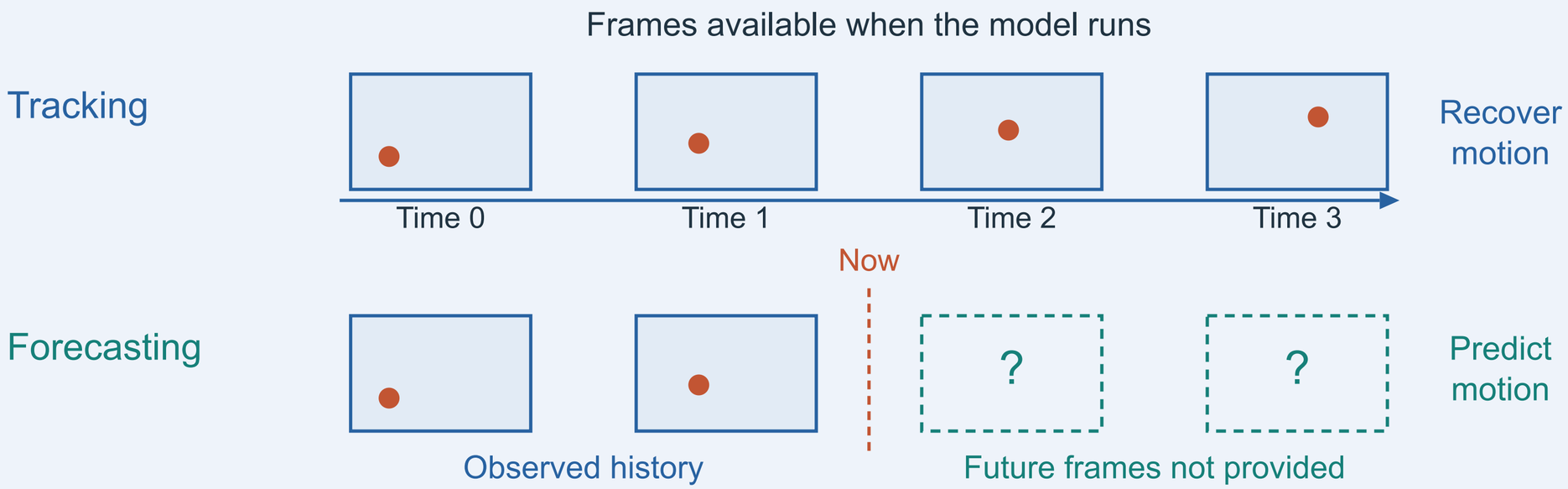
# Lifting a 2D track into 3D




Repeat for the same surface point at each time.

**Learning a 3D track requires geometry, correspondence, and camera information.**

# Tracking observed motion and forecasting

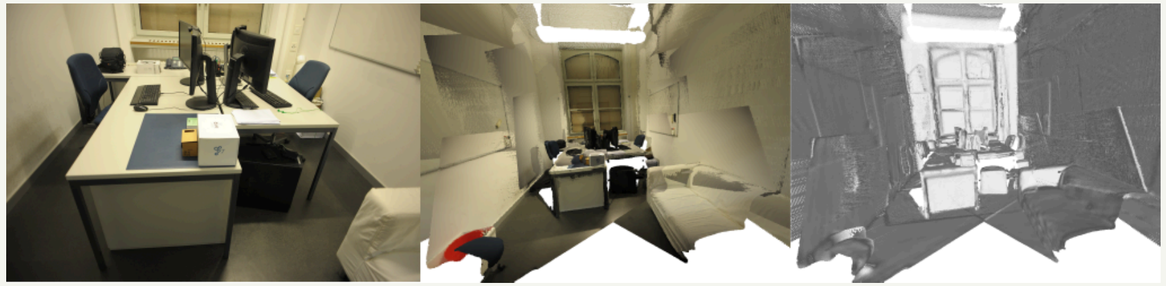



Both can return point trajectories. Their available observations differ.

**We will learn geometry and correspondence, then predict unobserved future motion.**

Zhang et al., D4RT (2026) · Huang et al., PointWorld (2026) · Jeon et al., Video-Point Model (October 2026)

# Learning 3D geometry



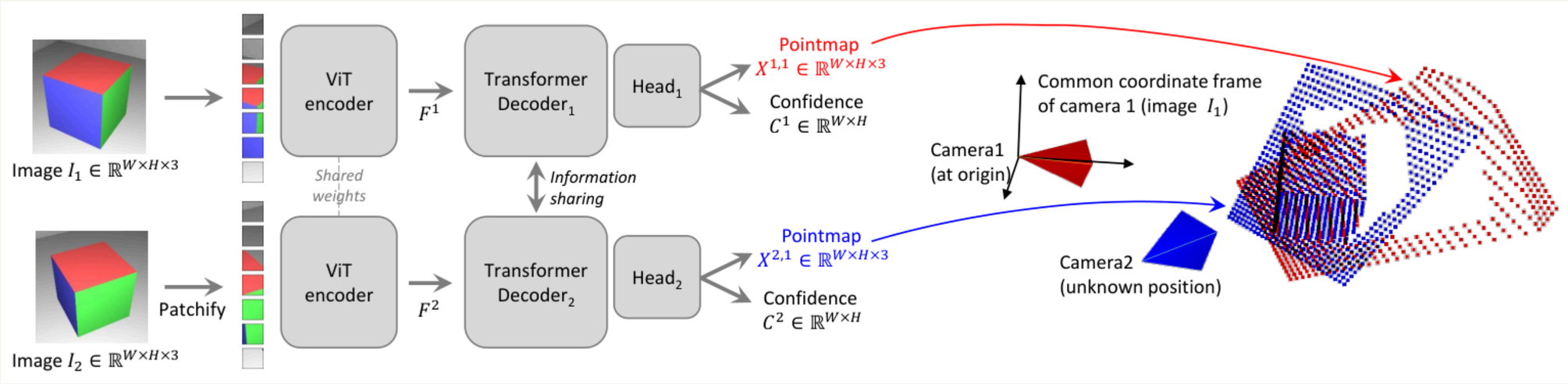
Different images constrain one underlying scene.

## What supervision turns images into a spatial representation?

First learn from reference 3D geometry. Then fit scenes using image supervision.

Mildenhall et al., NeRF (2020) · Wang et al., DUSt3R (2024)

# DUSt3R

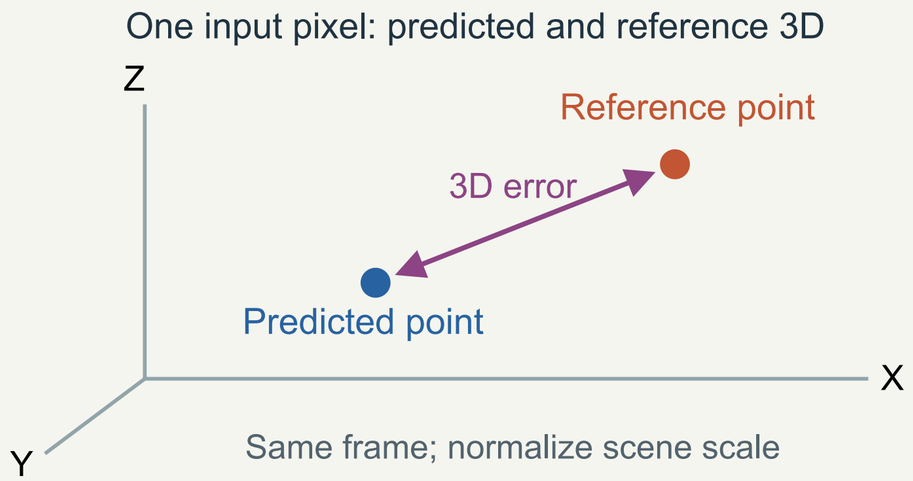



Two images are encoded and decoded into two pointmaps in the first camera's coordinate frame.

**A pointmap stores one 3D position for each image pixel; both outputs use the first camera's axes.**

Wang et al., DUSt3R (2024)

# Pointmap supervision




## During training

Image pairs + reference 3D pointmaps.

## For a new scene

Images in → predicted pointmaps out.

Reference geometry can come from depth sensors, reconstruction, or simulation.

$$\mathcal{L}_{\text{geometry}} = \sum_{\text{valid pixels}} \left\| \underbrace{\widehat{\mathbf{X}}}_{\text{predicted 3D}} - \underbrace{\mathbf{X}^*}_{\text{reference 3D}} \right\|$$

**Compare 3D positions during training; no reference pointmap is needed for a new scene.**

Wang et al., DUSt3R (2024)

# Aligning multiple views



Multiple views must be reconciled into one reconstruction.

### Pairwise predictions

Each pair has a local geometric frame.

### Alignment variables

Estimate compatible scale and pose.

### Shared scene

Bring the predicted pointmaps into agreement.

**A strong learned predictor can still use geometric optimization at inference.**

Wang et al., DUSt3R (2024)

# What geometry learning recovers



Learned geometry priors help under large viewpoint changes and incomplete observations.

### Visible evidence

Images constrain surfaces that were seen.

### Learned prior

Training supplies plausible geometric structure.

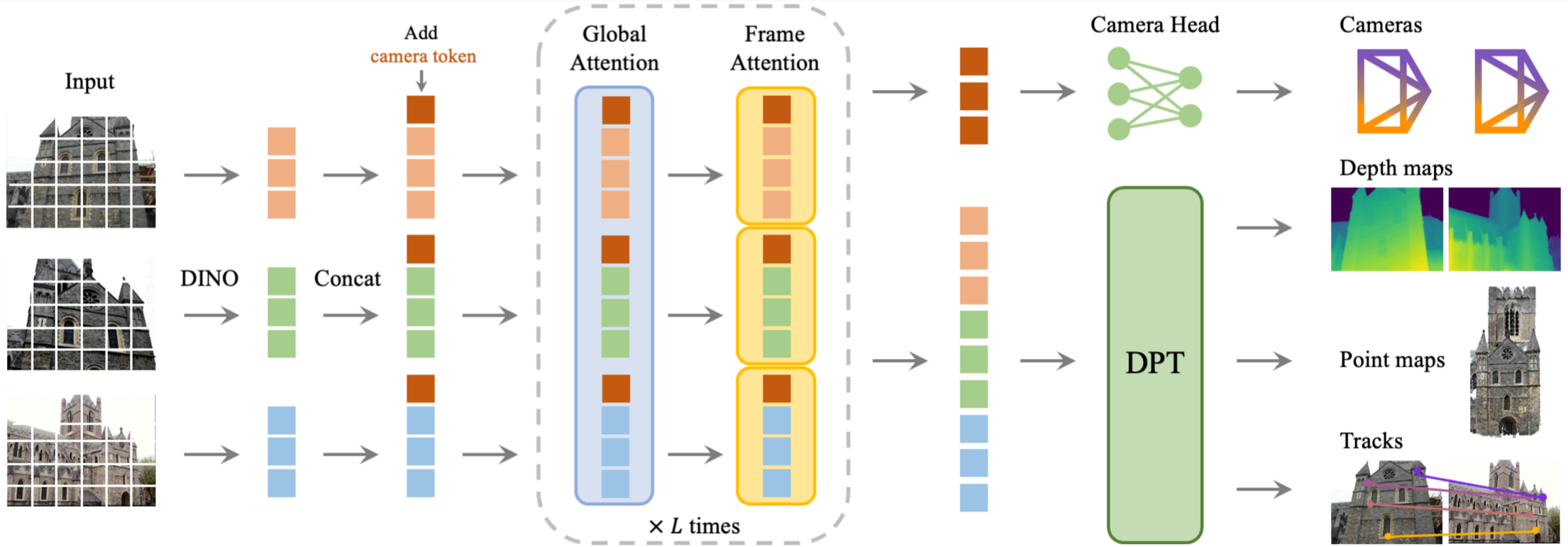
### Remaining ambiguity

Hidden shape and absolute scale may remain uncertain.

**A coherent reconstruction is a useful estimate of the scene, not proof that every hidden surface is correct.**

Wang et al., DUSt3R (2024) · Huang et al., PointWorld (2026)

# VGGT

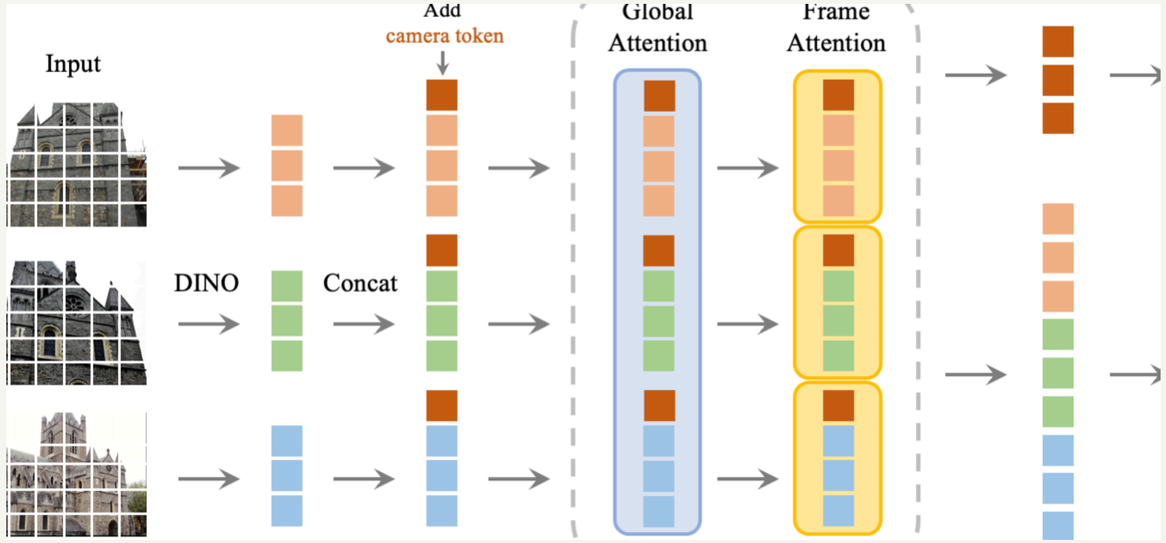



One feed-forward model jointly predicts several geometric attributes from a collection of views.

**Camera parameters, depth, pointmaps, and tracking features share a visual backbone.**

Wang et al., VGGT (2025)

# Frame and global attention




VGGT alternates within-frame and cross-frame information exchange.

**Frame attention**

Relate patches within one image.

**Global attention**

Relate evidence across the image collection.

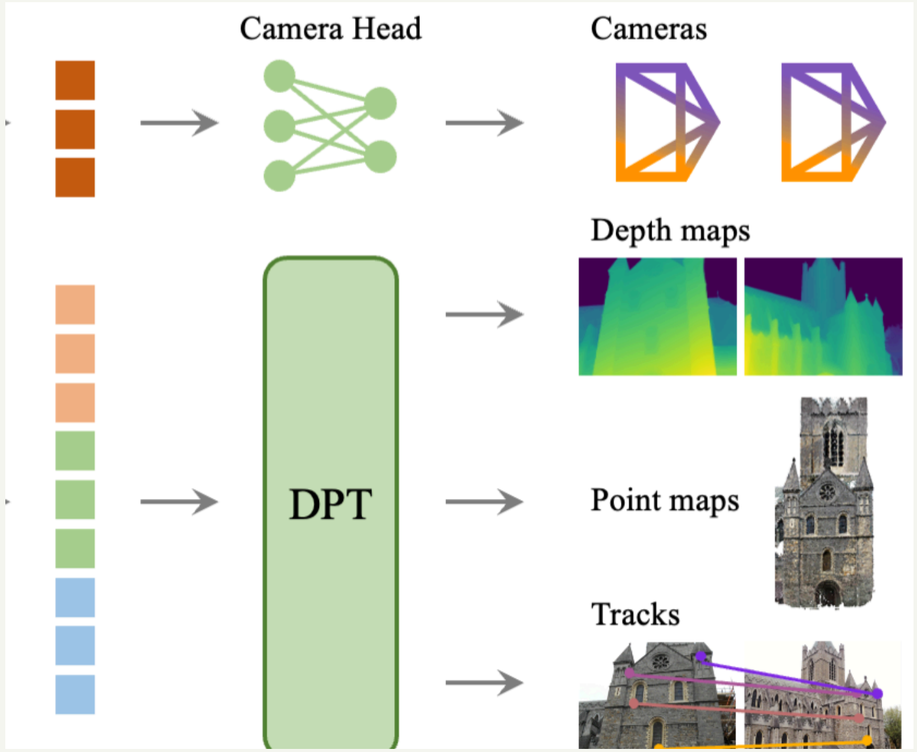
**Camera tokens**

Aggregate information for camera prediction.

**Cross-view information helps a patch acquire a spatial interpretation.**

Wang et al., VGGT (2025)

# Learning geometry together



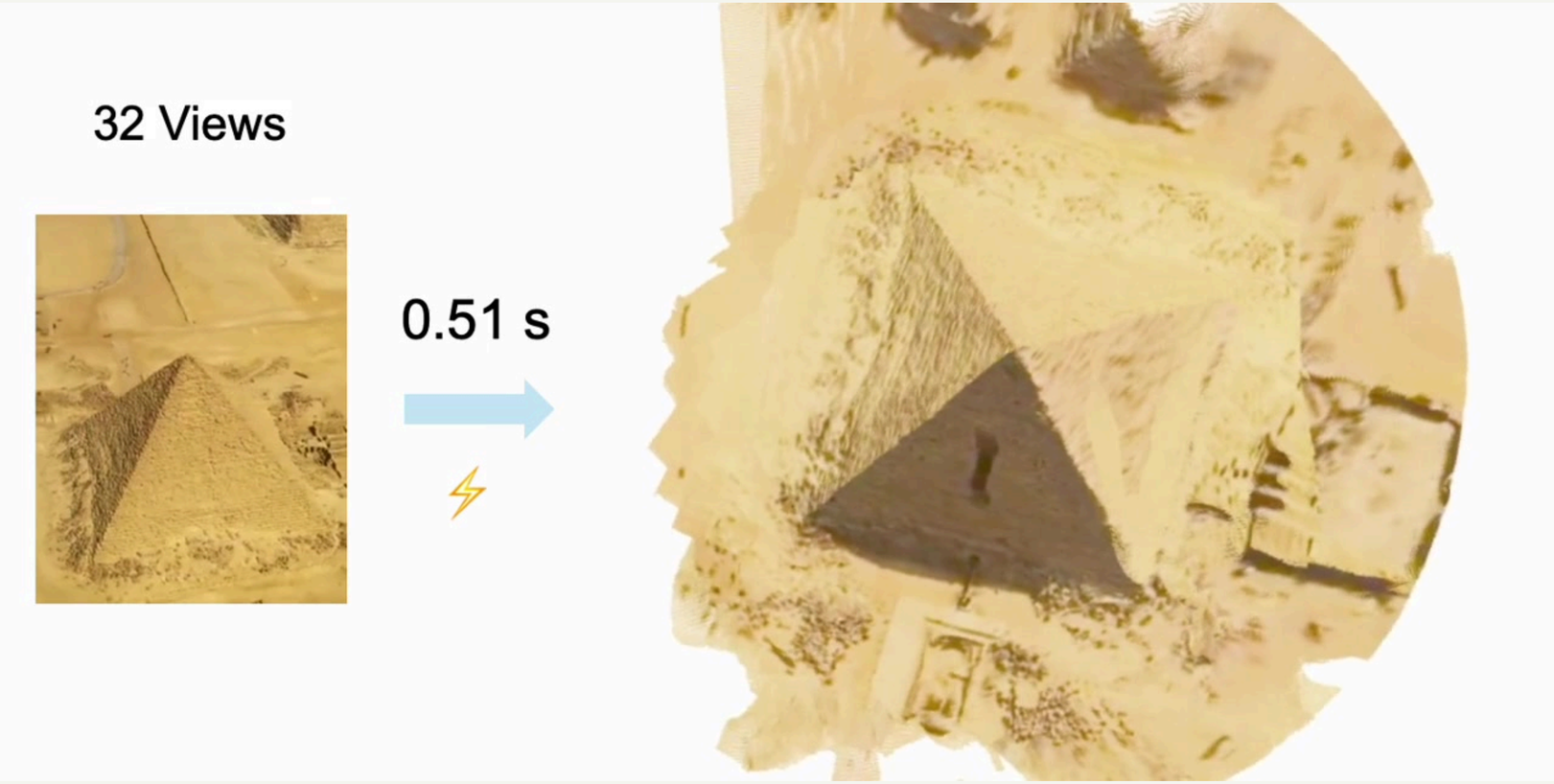

The common representation supports related geometric predictions.

| Output | Supervision asks |
| --- | --- |
| Camera | Reference camera calibration |
| Depth / pointmap | Reference depth / 3D positions |
| Correspondence | Reference tracks across views |

**Images are the network input; camera, depth, point and track targets supply training supervision.**

Wang et al., VGGT (2025) · Wang et al., VGGT-Ω (2026)

# VGGT reconstruction



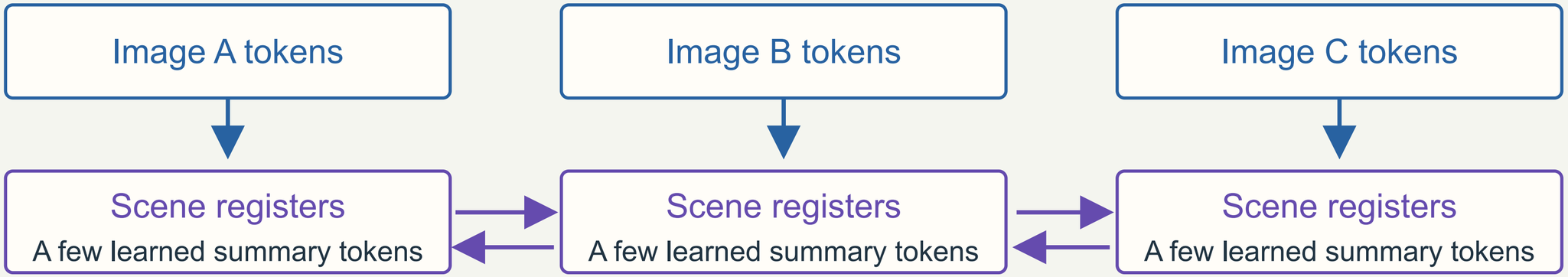

VGGT official demonstration: images and inferred geometry across varied scenes.

**Read the result as estimated cameras plus spatially organized scene information.**

Wang et al., VGGT (2025)

# VGGT-Ω

Full cross-view attention repeatedly exchanges many image tokens.




Exchange shared scene information through these summaries.

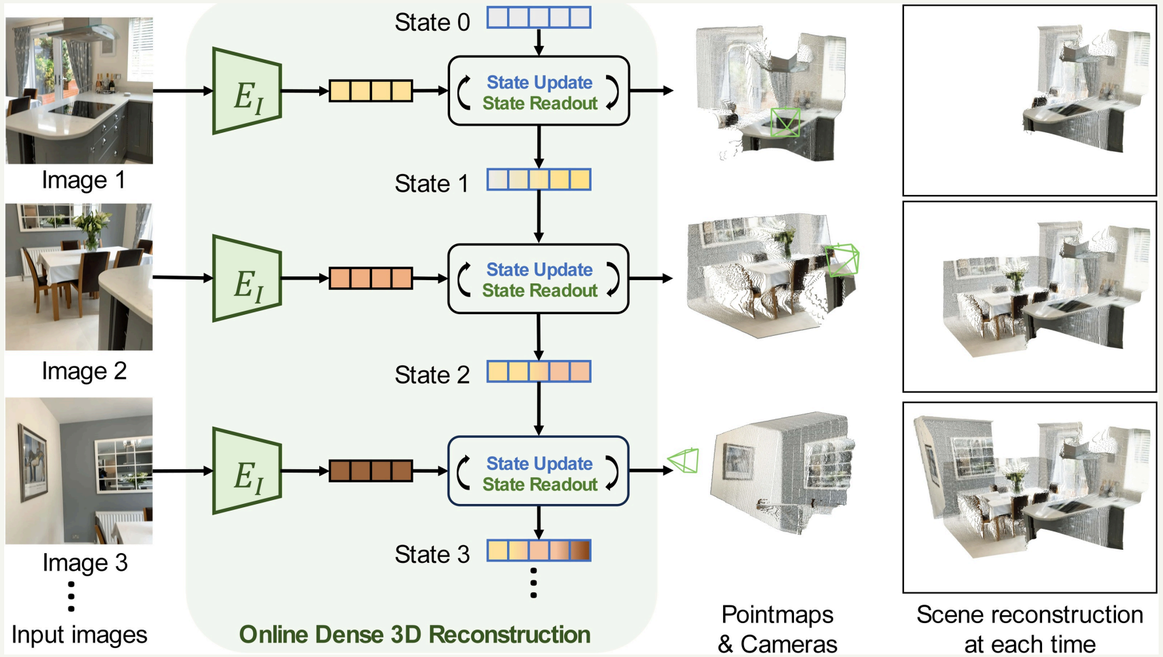
**What is retained?** Image features and shared scene information still support camera and depth prediction.

**What changes?** Selected layers replace full inter-frame attention with register attention.

**Compact summaries reduce cross-view communication; they do not replace the scene with a fixed 3D grid.**

Wang et al., VGGT-Ω (2026)

# CUT3R




A recurrent state is updated as new image observations arrive.

$$h_t = U(h_{t-1}, I_t)$$

$h_t$: persistent state. $I_t$: new image. $U$: learned update.

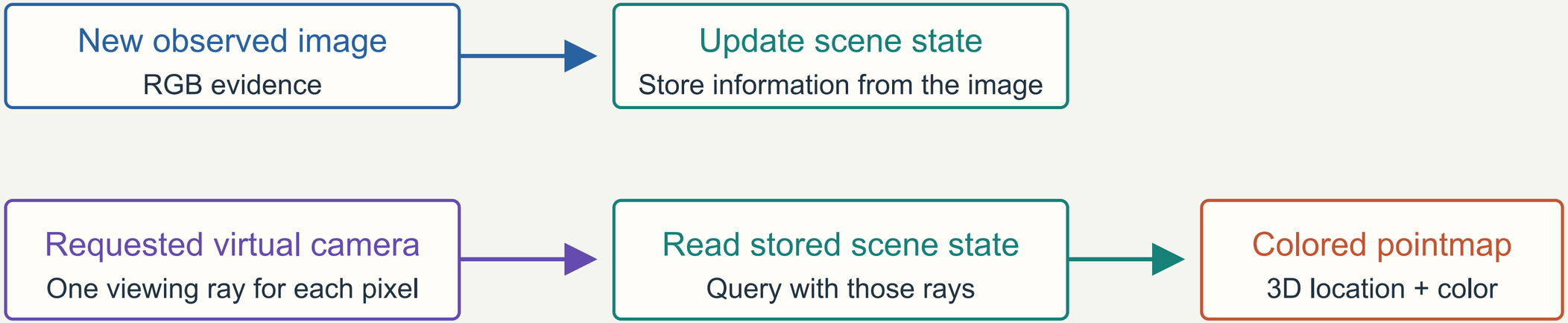
Persistent state combines past evidence with the current view.

Read out geometry in a common coordinate system.

**The state update here is driven by new observations.**

Wang et al., CUT3R (2025)

# Querying the persistent state




**Update:** an actual new image supplies evidence.

**Query:** camera rays request a view; no new RGB image is provided.

**An unseen view is inferred from stored evidence and learned priors.**

Wang et al., CUT3R (2025)

# From 3D targets to image supervision

## DUSt3R / VGGT / CUT3R

Images → learned predictor → 3D estimate

**Compare with reference geometry**

Depth, point positions, cameras, tracks

Train shared network weights across scenes.

## NeRF / Gaussian Splatting

Scene parameters → renderer → image

**Compare with the observed photo**

Same camera and same pixels

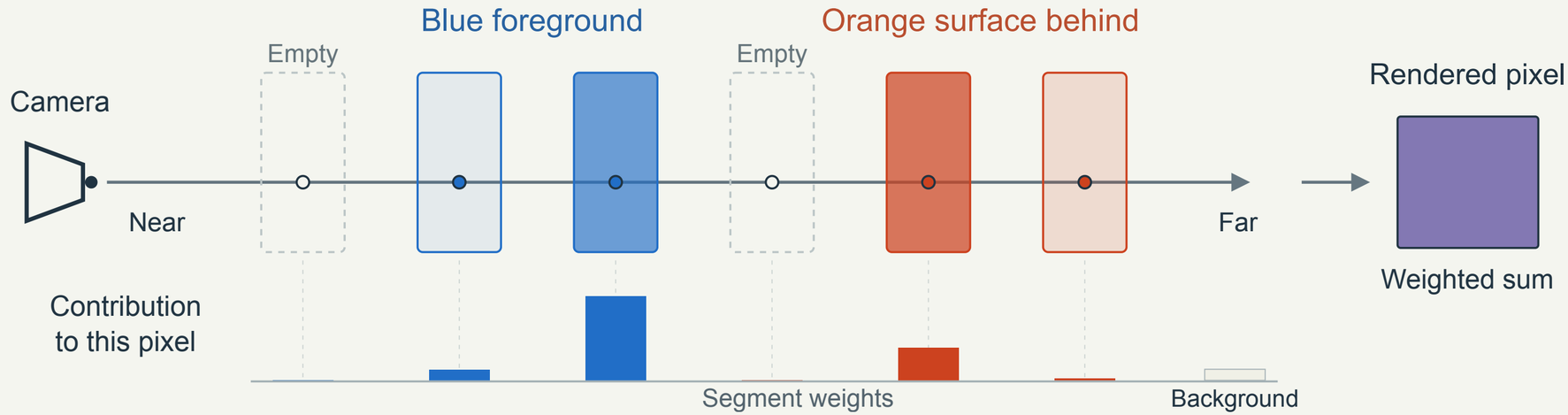
Standard fitting optimizes one scene.

**A 3D representation can be supervised through the images it produces.**

Wang et al., DUSt3R (2024) · Mildenhall et al., NeRF (2020) · Kerbl et al., 3D Gaussian Splatting (2023)

# Volume rendering

Along one camera ray, NeRF predicts **color** and **density** at sampled 3D locations.




$$\widehat{\mathbf{C}} = \sum_i \underbrace{T_i}_{\text{visibility}} \underbrace{\alpha_i}_{\text{opacity}} \underbrace{\mathbf{c}_i}_{\text{color}} + \underbrace{T_{\text{end}}\mathbf{c}_{\text{bg}}}_{\text{remaining background}}$$

**α: opacity here.** Increases with density and segment length.

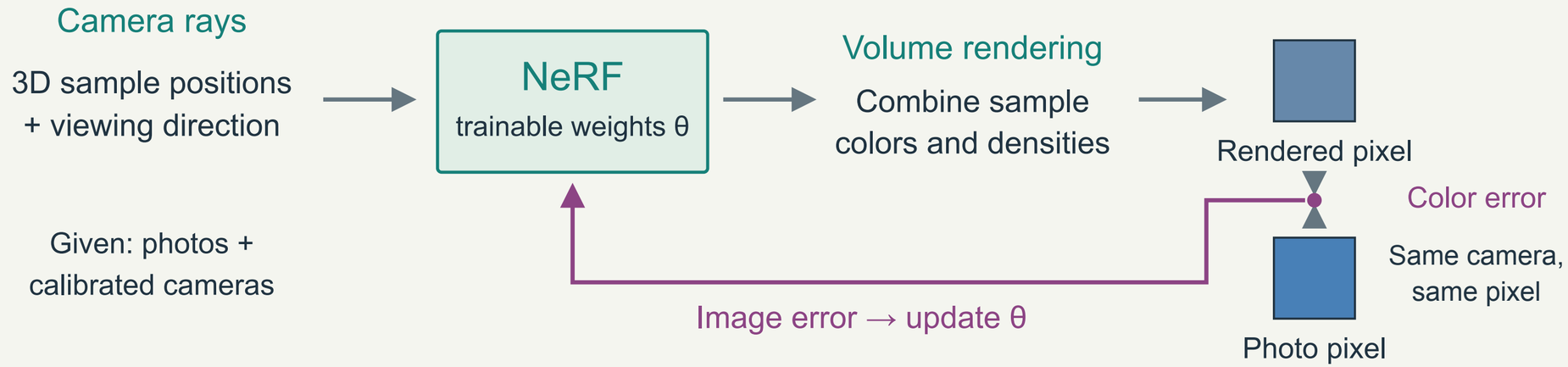
**T: visibility through the foreground.** Decreases as earlier segments become opaque.

**Dense material near the camera reduces the contribution of colors behind it.**

Mildenhall et al., NeRF (2020)

# Learning a NeRF from images

Fit one shared 3D field so that its rendered views match photographs of the scene.




$$\mathcal{L}_{\text{RGB}} = \sum_{\mathbf{r}} \left\| \underbrace{\widehat{\mathbf{C}}_\theta(\mathbf{r})}_{\text{rendered pixel}} - \underbrace{\mathbf{C}_{\text{obs}}(\mathbf{r})}_{\text{photo pixel}} \right\|^2$$

The same field must explain rays from many camera views. Supervision comes from image colors.

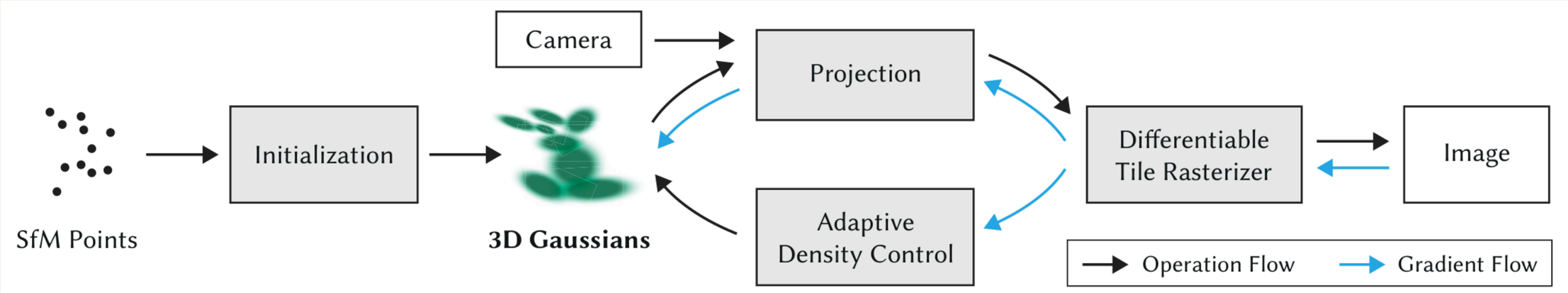
---

**NeRF updates network weights for one scene. Gaussian splatting will update explicit scene parameters.**

Mildenhall et al., NeRF (2020)

# Fitting 3D Gaussians from images

Initialize from sparse SfM points and cameras recovered from the photos.




Black arrows: forward operations and density control. Blue arrows: gradient flow.

**Optimize:** positions · scales and rotations · opacity · view-dependent color

$$\mathcal{L} = (1-\lambda)\underbrace{\mathcal{L}_1}_{\text{pixel color error}} + \lambda\underbrace{\mathcal{L}_{\text{D-SSIM}}}_{\text{local image structure}}$$

Both terms compare the rendered view with its photo; λ balances the two errors.

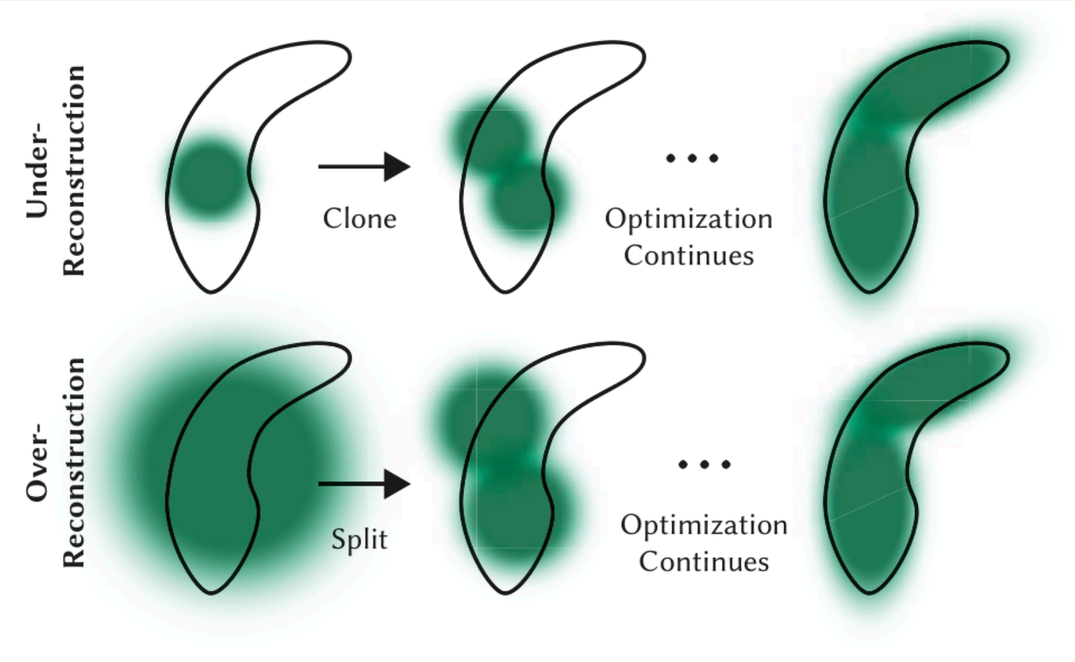
---

**Backpropagate through projection and splatting to update the Gaussians themselves.**

Kerbl et al., 3D Gaussian Splatting (2023)

# Adding, splitting and removing Gaussians

Image-error gradients select candidates; size decides whether to clone or split.




Black outline: illustrative shape, not a 3D training target. Green: Gaussian coverage.

### Clone

Add small Gaussians where coverage is missing.

### Split

Replace a large Gaussian with smaller ones to fit detail.

### Prune

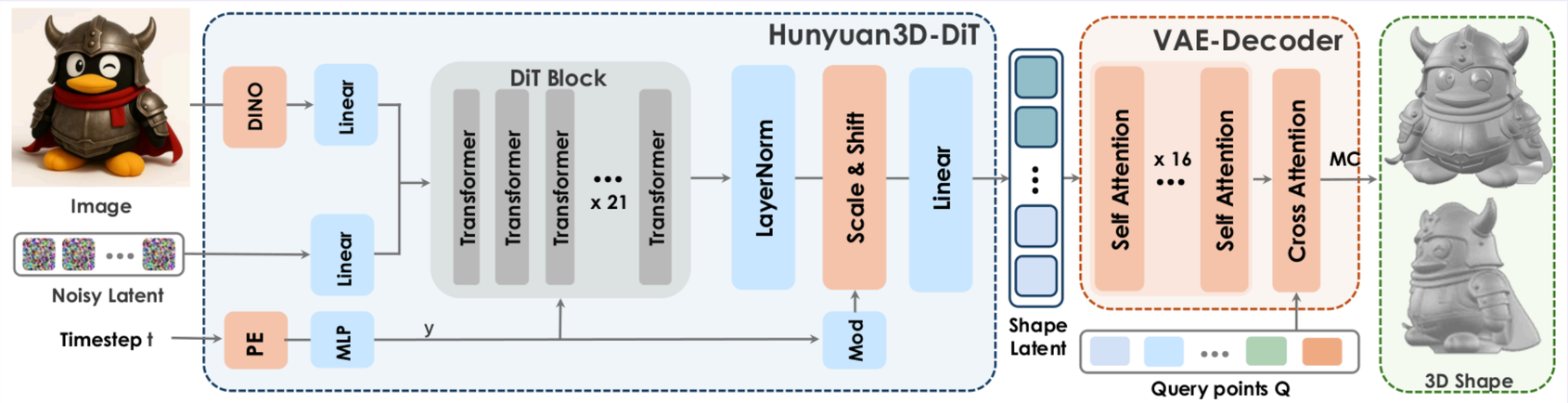
Remove nearly transparent or oversized Gaussians.

**Alternate parameter optimization with changes to the number and placement of Gaussians.**

Kerbl et al., 3D Gaussian Splatting (2023)

# Native 3D latent generation

**Image condition** → **Generate a 3D latent** → **Decode a 3D asset**




Hunyuan3D 2.1: image-conditioned shape latents are decoded into geometry.

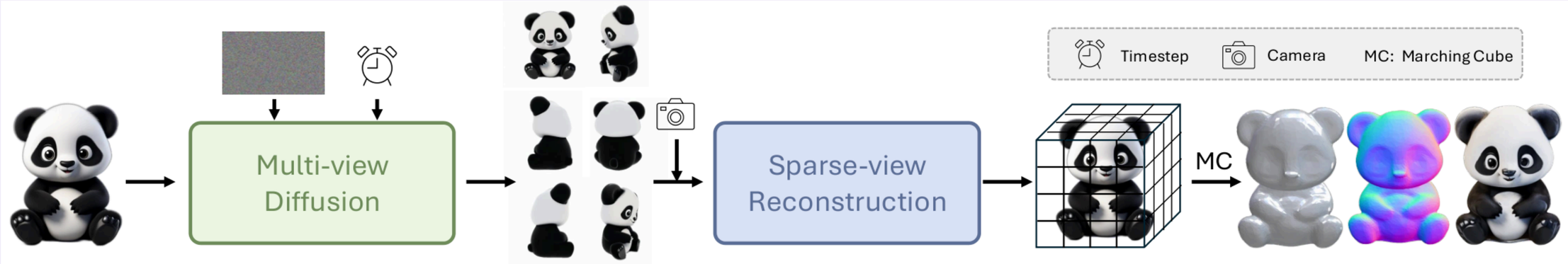
**Learn the code:** a 3D autoencoder compresses and reconstructs training shapes.

**Generate the code:** diffusion or flow learns its conditional distribution.

**The image is the condition; the generated variable is a 3D representation.**

# Generated views → a shared 3D scene

**Image or camera request** → **Generate related views** → **Reconstruct shared geometry**




Hunyuan3D 1.0: multiview diffusion followed by feed-forward reconstruction.

## A set of views

Hunyuan3D 1.0 generates related images, then reconstructs one asset.

## A camera-controlled video

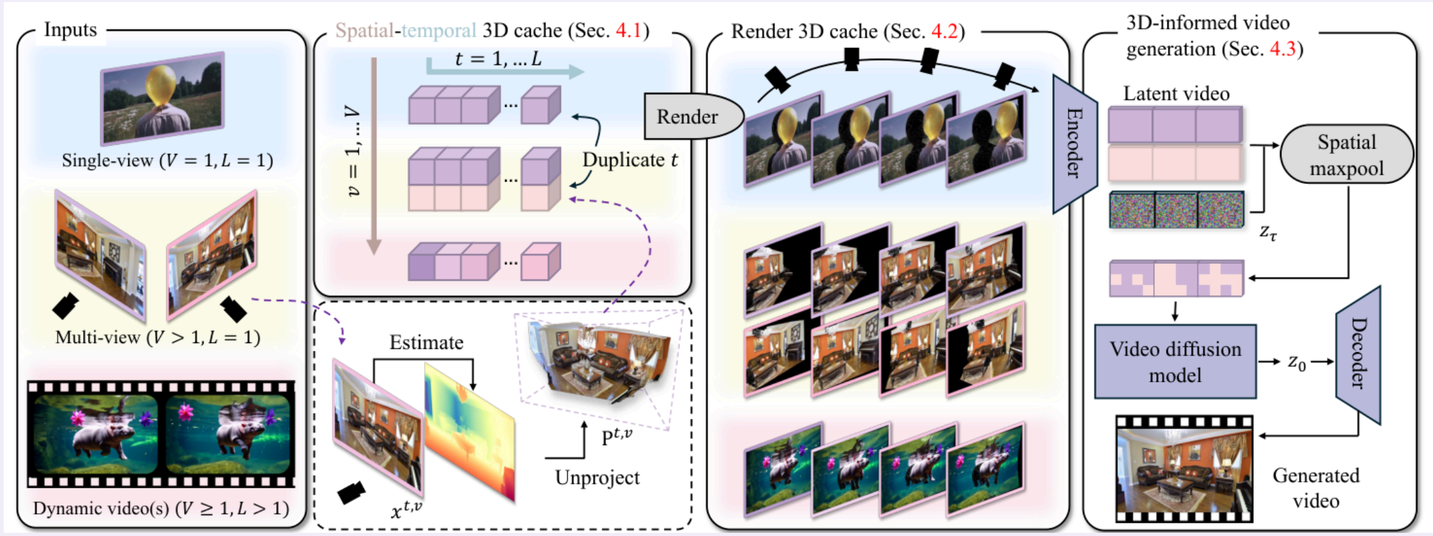
Generate a walkthrough, then recover the geometry shared by its frames.

---

**Coherent images help; a common 3D scene still has to be established.**

# GEN3C: geometry guides video generation

**Estimate depth → 3D cache** → **Render requested cameras** → **Condition video generation**




GEN3C: images or video → spatial-temporal cache → camera-conditioned video.

**Supported regions:** cache projections constrain the generated views.

**Newly exposed regions:** the video prior supplies plausible missing appearance.

**A cache rendering provides geometric evidence; generated completion remains an inference.**

# Lyra 2.0: persistent spatial memory




Lyra 2.0 official scene-exploration demonstration.

### Store

Keep past frames and their recovered 3D geometry.

### Retrieve

The requested camera selects relevant past views and correspondences.

### Generate and add

Generate new frames, recover geometry, and add them to the history.

**Useful history depends on where the camera goes, as well as when a frame was seen.**

# Changing the camera or changing the action

## View query

**The same scene or event**

**Camera A** → View A

**Camera B** → View B

Preserve the scene’s spatial or space-time structure.

## Intervention query

**The same observed history**

**Action A** → Predicted response

**Action B** → Predicted response

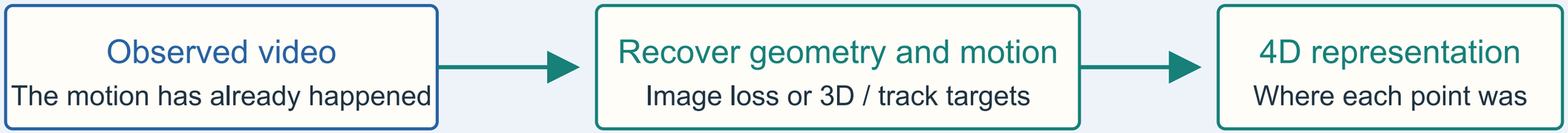
Predict how the future responds to the specified action.

**An explorable generated scene establishes a spatial capability. Action response needs its own training and evaluation evidence.**

# Learning geometry and motion

**Fit an evolving scene**
Render a time-dependent representation and compare it with the recorded images.

**Learn a tracking model**
Predict the same surface point's 3D position across frames.

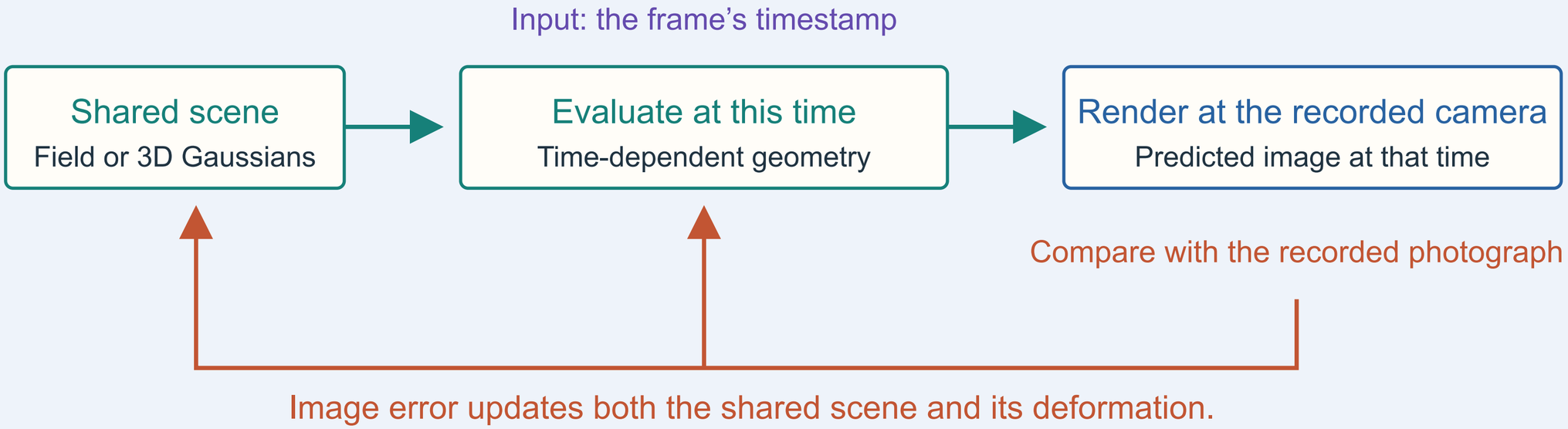



**In both cases, the video being explained is available to the reconstruction system.**

Feng et al., St4RTrack (2025) · Xiao et al., SpatialTrackerV2 (2025) · Zhang et al., D4RT (2026)

# Learning a time-varying scene

Use time-stamped photographs to fit one scene that changes over time.



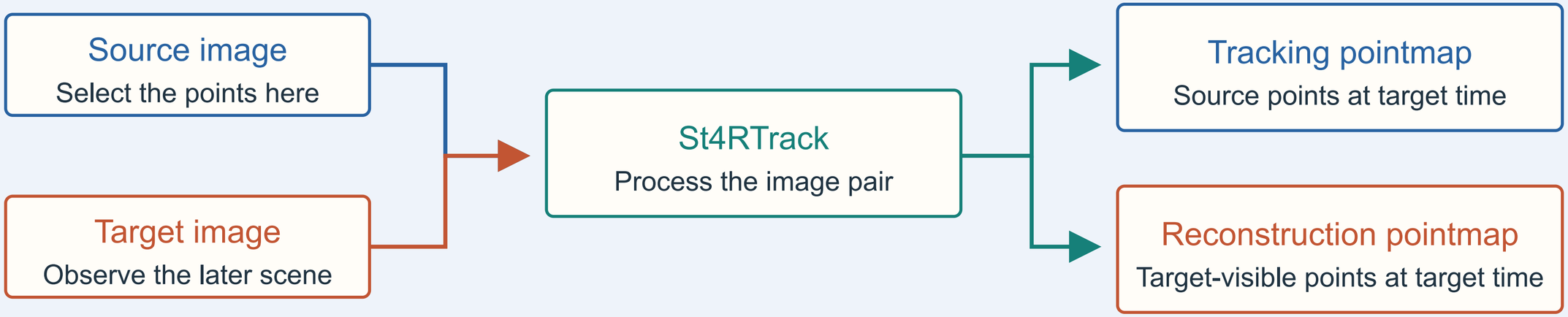

**D-NeRF:** map ray samples back to a shared field, then volume-render.

**4D Gaussian Splatting:** deform Gaussians at the requested time, then rasterize.

**This fits a recorded sequence. Predicting responses to new actions also requires action-conditioned training.**

Pumarola et al., D-NeRF (2021) · Wu et al., 4D Gaussian Splatting (2024)

# St4RTrack



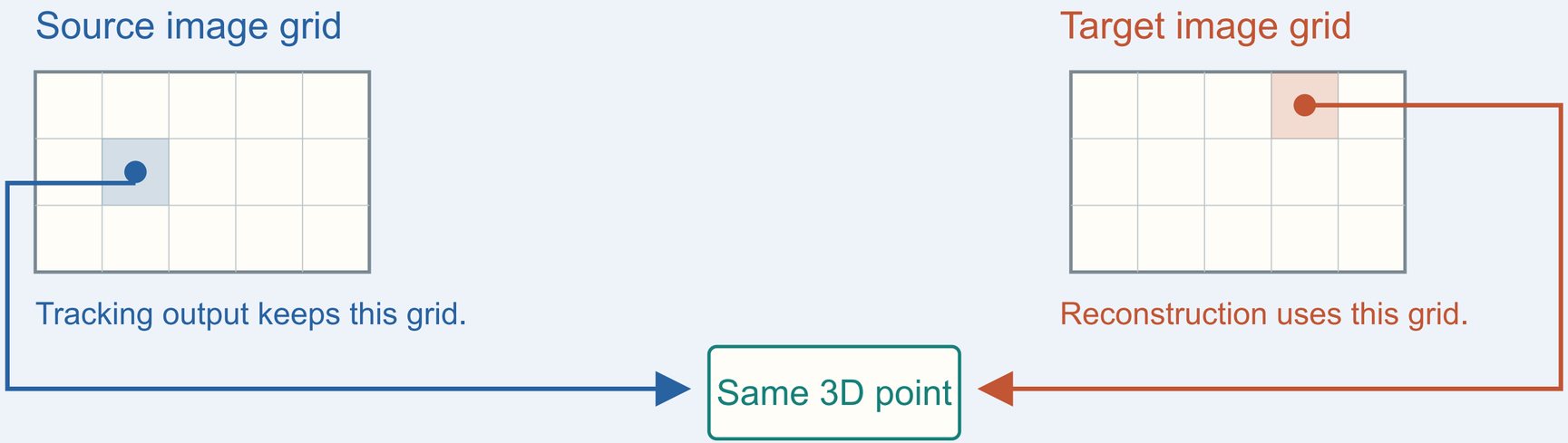

**Tracking:** where did the source-image surface points go?

**Reconstruction:** where are the surfaces visible in the target image?

**Both outputs report 3D positions at the same target time, in the same reference frame.**

Feng et al., St4RTrack (2025)

# Reconstruction and correspondence




Different image pixels can refer to the same surface point at the target time.

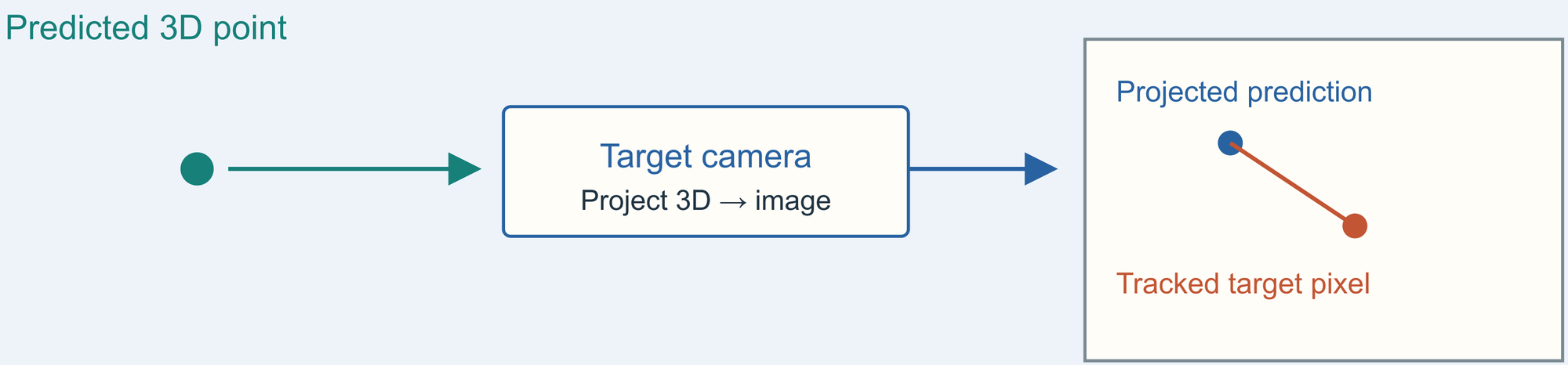
**A cell stores a 3D coordinate.** Its image-grid location says which surface point it refers to.

**When both views see the same point,** the two corresponding cells should agree on its target-time position.

**The grid indexes the point. The coordinate stored inside describes where it is.**

Feng et al., St4RTrack (2025)

# Learning from reprojection




Reduce the distance between these two image locations.

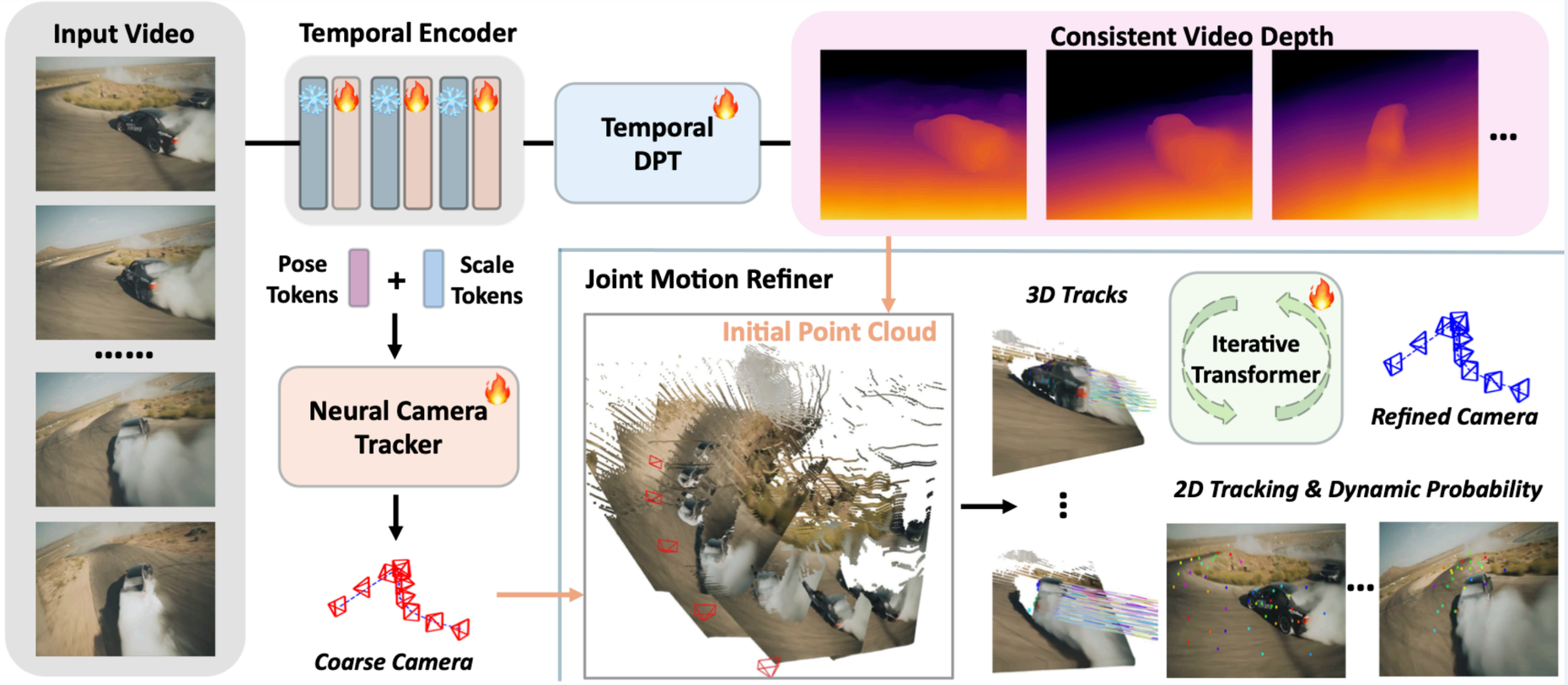
**Start with synthetic data:** supervise 3D positions and correspondence directly.

**Adapt to real video:** use 2D tracks, depth estimates, and camera consistency.

**A 2D match constrains a viewing ray; depth provides an additional constraint along that ray.**

Feng et al., St4RTrack (2025)

# SpatialTrackerV2

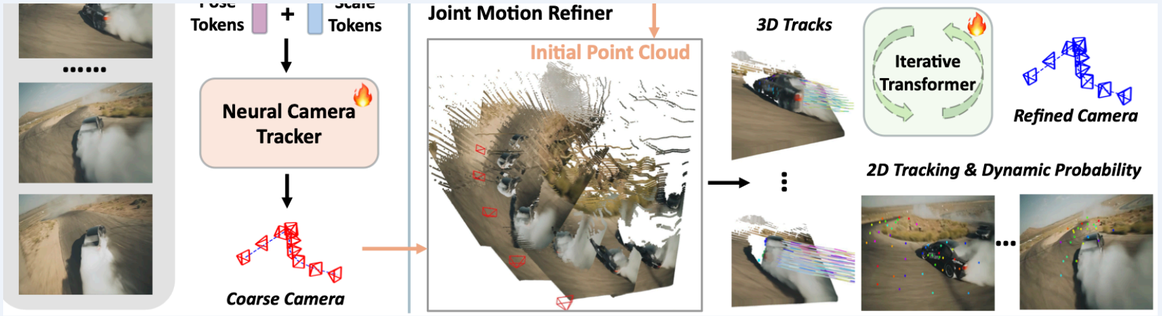



Geometry and camera initialization feed a joint refinement of motion and scene structure.

**The model estimates geometry, camera motion, and world-space trajectories together.**

Xiao et al., SpatialTrackerV2 (2025) · Wang et al., VGGT (2025)

# Refining geometry and tracks




Joint motion refinement reads evidence from the shared geometric representation.

**Initialize**

Estimate cameras and a scene geometry.

**Match**

Relate point evidence across the video.

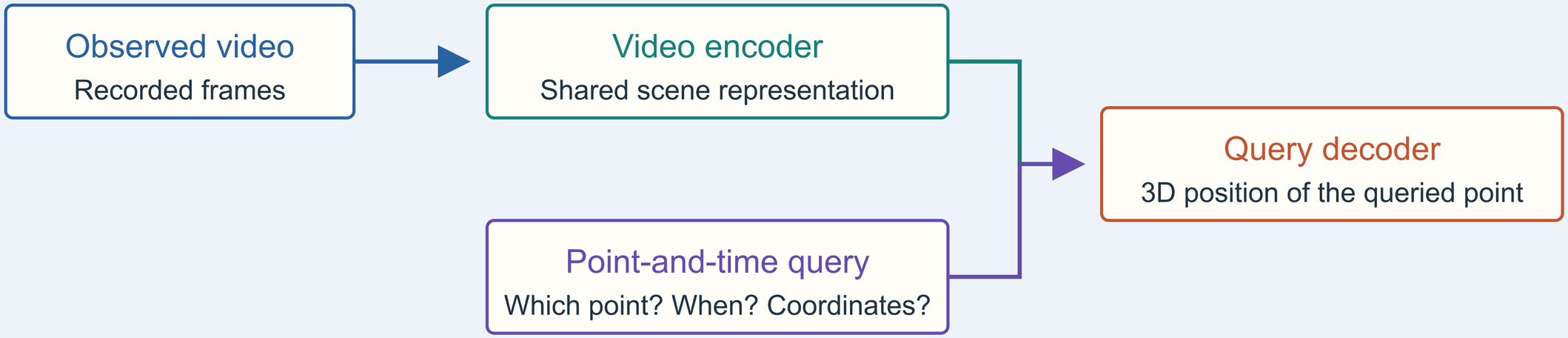
**Refine together**

Update motion and geometric estimates.

**The same observed changes must be explained by compatible camera and scene estimates.**

Xiao et al., SpatialTrackerV2 (2025)

# D4RT



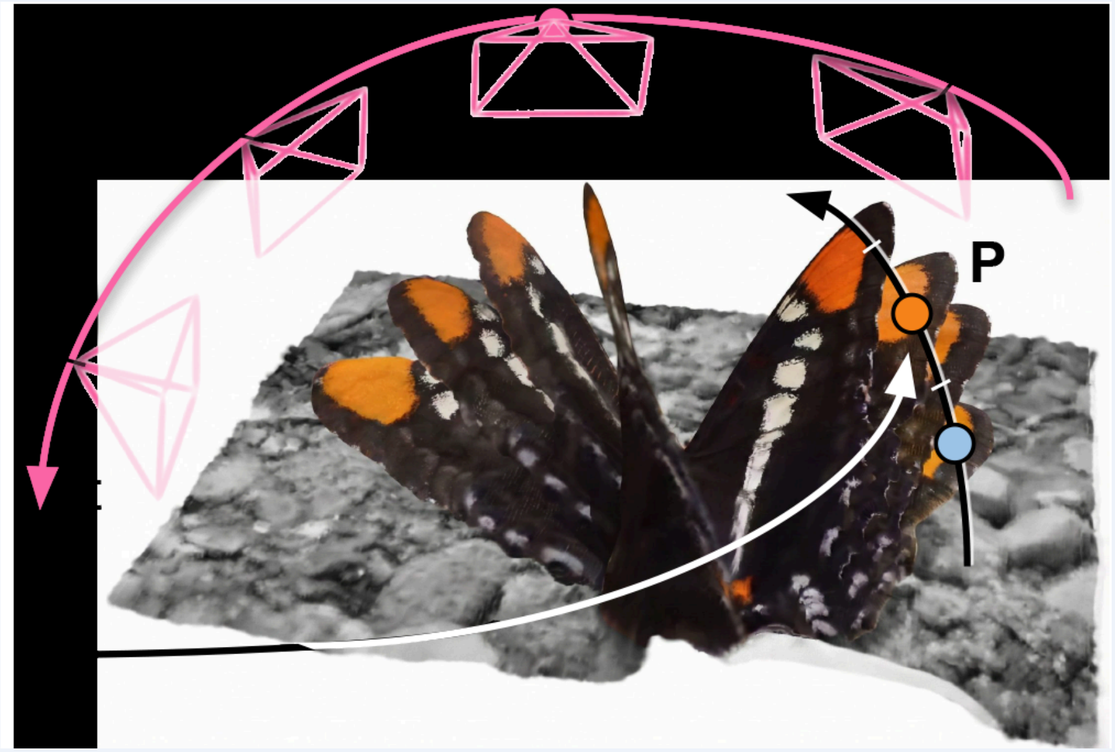

**Encode the video once.** Reuse the representation for many queries.

**Decode positions.** Many pixels give geometry; repeated times give tracks.

**D4RT makes the output request explicit instead of requiring a separate interface for every task.**

Zhang et al., D4RT (2026)

# Querying a 4D scene




Select one point on the wing, locate it later, and choose the camera axes used to report it.

### Select the point

A source pixel and its source frame.

### Choose the time

When should that same point be located?

### Choose the coordinates

Which observed camera’s axes express the 3D answer?

$$q = (u, v, t_{\text{src}}, t_{\text{tgt}}, t_{\text{cam}})$$

**Hold the source pixel and source time fixed to follow the same point.**

Zhang et al., D4RT (2026)

# Supervision for 4D perception

| Evidence | What it directly constrains | What remains inferred |
|---|---|---|
| Synthetic geometry and tracks | Positions, identities, visibility, cameras | Transfer to real observations |
| Posed RGB-D video | Visible geometry and image consistency | Hidden surfaces and uncertain matches |
| Real RGB video | Image correspondences and reprojection | Metric structure and occluded motion |

**Different training signals provide different parts of the geometric explanation.**

Feng et al., St4RTrack (2025) · Xiao et al., SpatialTrackerV2 (2025) · Zhang et al., D4RT (2026)

# Tracking and forecasting

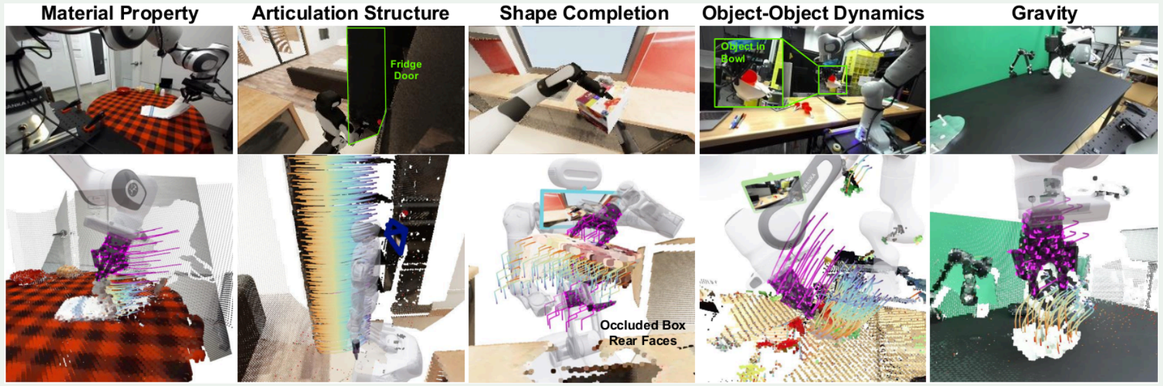
| Task | Available at inference | Requested output |
| --- | --- | --- |
| Tracking / reconstruction | The video containing the target event | Recover its geometry and motion |
| Forecasting | Only the observed history and conditions | Predict unobserved future motion |

$$\underbrace{I_{\leq t}}_{\text{observed}} \;\Big|\; \underbrace{I_{>t}}_{\text{not provided for forecasting}}$$

**Identical trajectory visualizations can represent different modeling tasks.**

Zhang et al., D4RT (2026) · Huang et al., PointWorld (2026) · Jeon et al., Video-Point Model (October 2026)

# Predicting future 3D



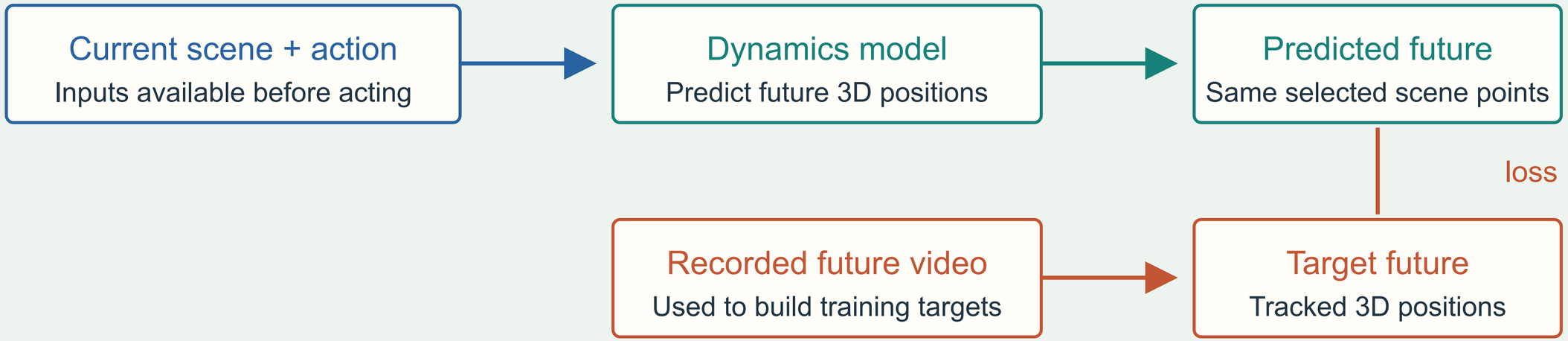

A proposed interaction can change shape, articulation, contact, and object motion.

## The future observations are no longer available.

Can we predict the same points' future positions before observing them?

Huang et al., PointWorld (2026)

# Learning future point motion



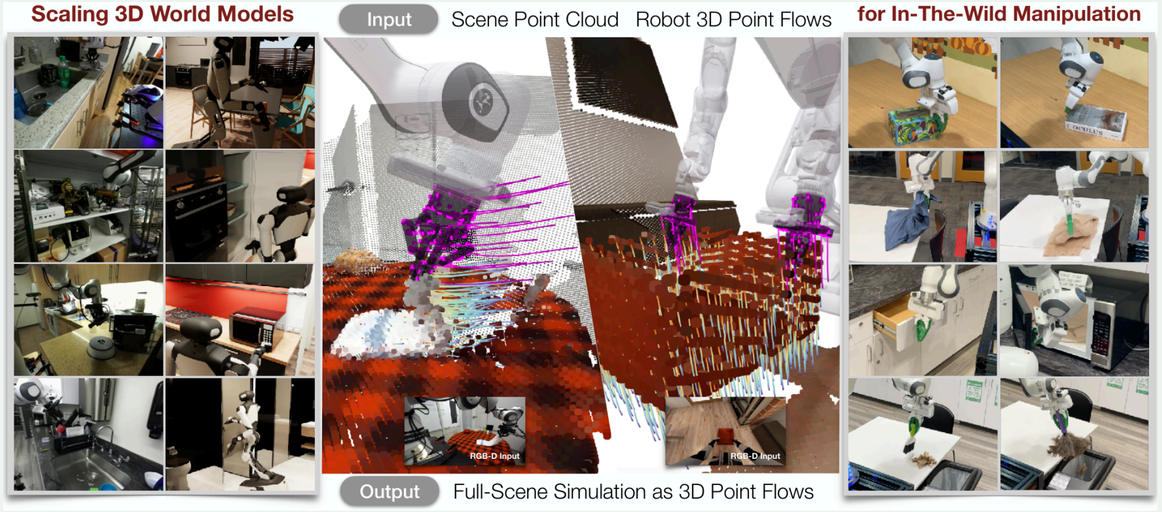

**Training:** observe what happened after the recorded action.

**Inference:** predict before those future observations exist.

**The dynamics model receives the current scene and action. Future frames supply training targets only.**

Huang et al., PointWorld (2026)

# Predicting point motion




The model predicts future positions for the initially sampled scene points.

$$\widehat{\mathbf{X}}_{i,t+k} = \mathbf{X}_{i,t} + \widehat{\Delta\mathbf{X}}_{i,k}$$

$\widehat{\Delta\mathbf{X}}_{i,k}$ is measured from the start of this prediction chunk.

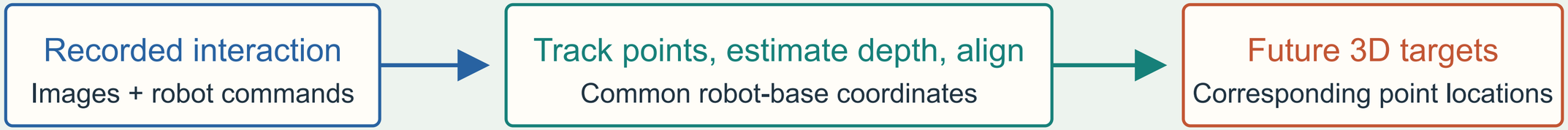
Point identity *i* stays fixed; *k* selects a future time in the chunk.

**Corresponding targets make point prediction different from generating an unordered new cloud.**

Here “point flow” means 3D displacement—not a diffusion velocity.

Huang et al., PointWorld (2026)

# Learning from recorded interactions



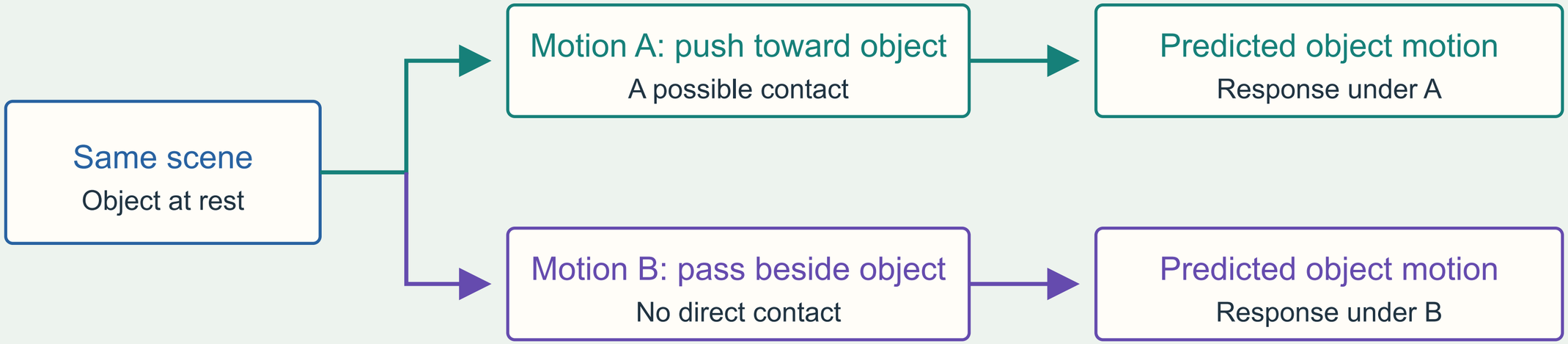

**Real video:** tracking provides identity; depth and camera alignment provide 3D location. Exclude invalid or occluded targets.

**Simulation:** the environment can supply point positions, correspondence, and visibility directly.

**Earlier geometry and tracking models supply labels; the dynamics model predicts without future frames.**

# Action-conditioned futures

Keep the initial observation fixed; query two proposed robot motions.

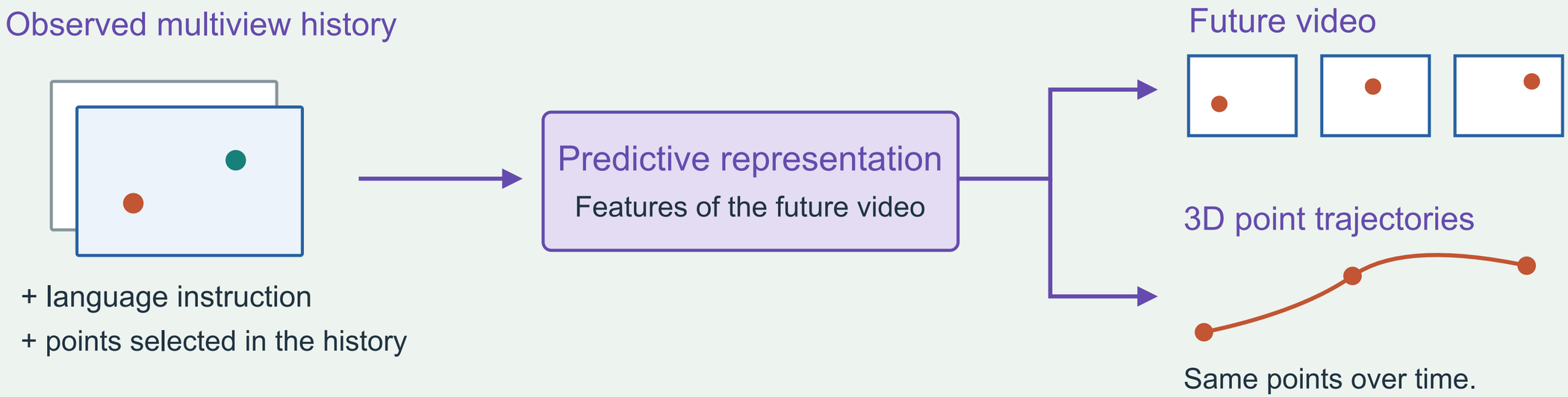



**Contact geometry matters:** where the robot moves relative to the object.

**Training needs paired evidence:** recorded actions and the motion that followed.

**The forecast should respond correctly to the action; different actions need not always produce different outcomes.**

# VPM: future video and 3D point trajectories




Conceptual diagram: video features support the point-trajectory forecast.

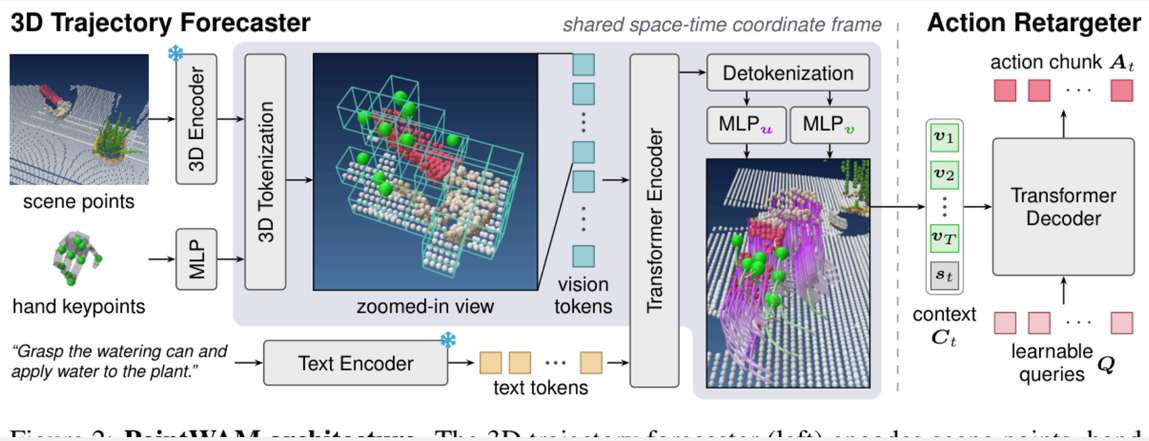
**Video supervision:** predict future appearance across views.

**Trajectory supervision:** predict the same points' future 3D positions.

---

**The target future frames are unavailable at inference: this is forecasting.**

# Predicting the hands and the scene




PointWAM learns scene and hand trajectories in a shared 3D frame.

**Condition**

Observed geometry and a language instruction.

**Forecast**

Scene and hand point trajectories.

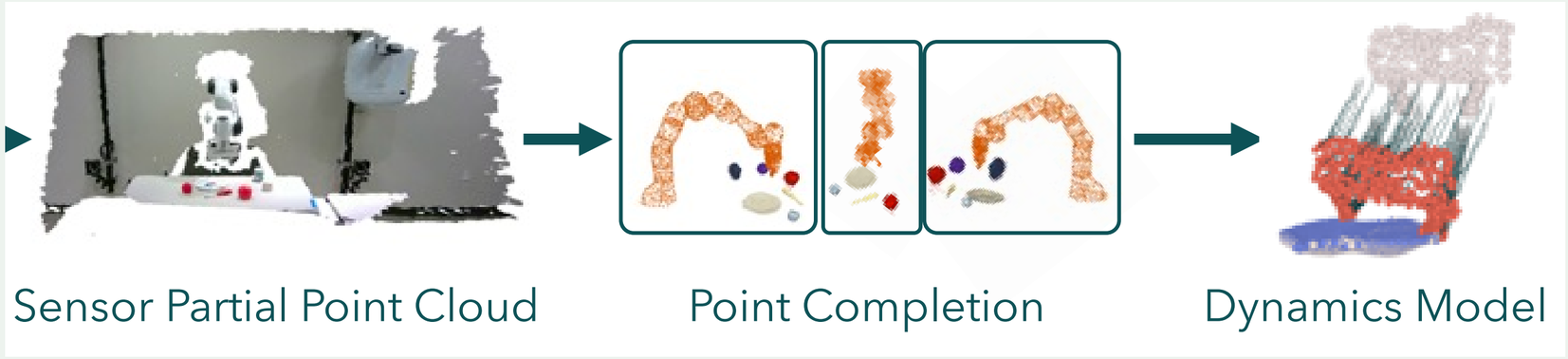
**Retarget**

Convert the hand forecast to robot actions.

**Here the learned output includes action-related motion, not only a response to a supplied action.**

Park et al., PointWAM (2026)

# Completing the scene before prediction




3DPWM: partial observed cloud → inferred completed cloud → predicted dynamics.

**Observed:** surfaces visible to the sensor.

**Completed:** inferred geometry behind or outside those surfaces.

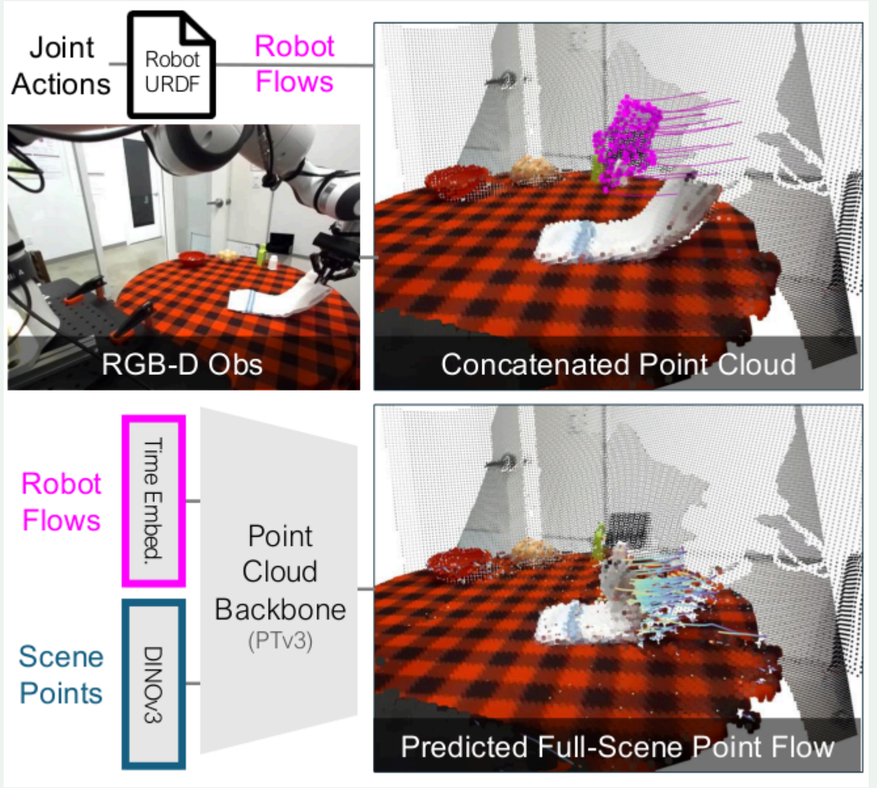


Official demonstration: completed geometry supports predicted rollouts.

**Completion supplies missing spatial context. It remains an inference, not a new observation.**

Peri et al., 3D Point World Models (2026)

# PointWorld

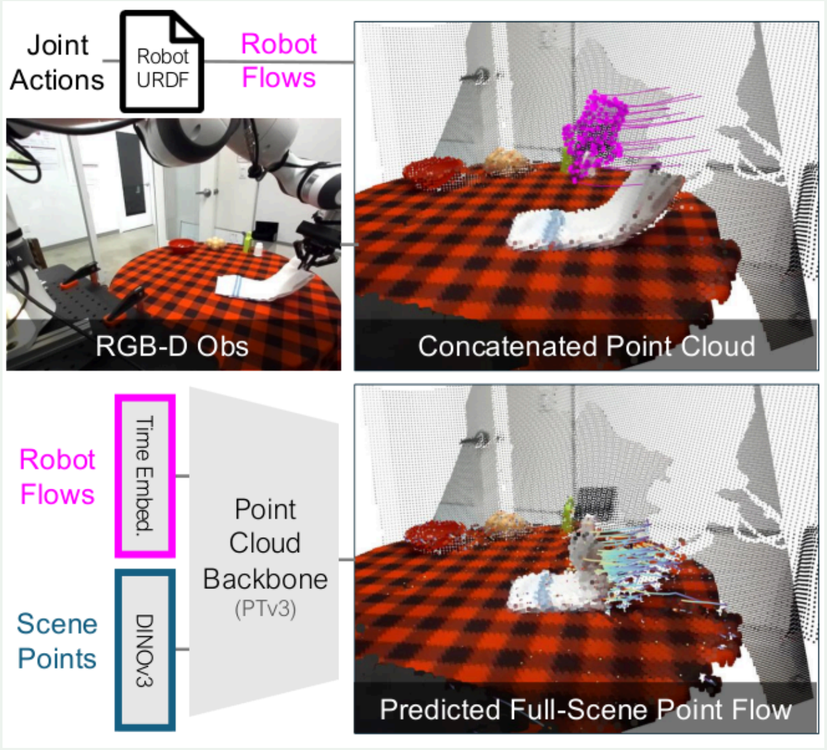



RGB-D scene points and robot point flows condition a shared point-cloud backbone.

**Predict the action-conditioned motion of scene points in a single forward pass.**

Huang et al., PointWorld (2026)

# State and action in 3D




PointWorld combines an initial scene cloud and proposed robot point motion.

**Scene**

A partial 3D observation of the environment.

**Action**

How the robot geometry would move.

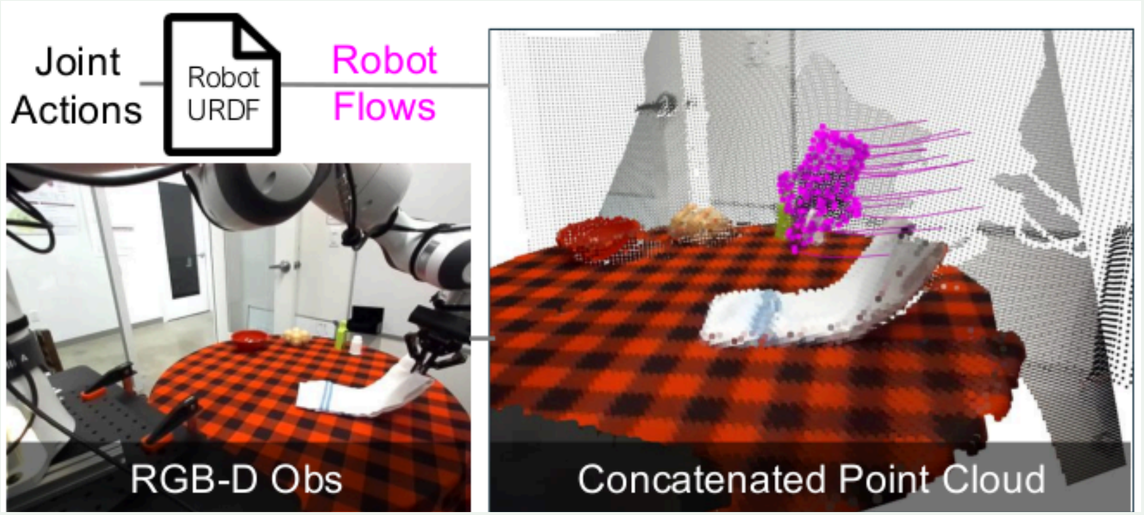
**Response**

How the environment points would move.

**Both the scene and the proposed interaction can be expressed in one spatial frame.**

Huang et al., PointWorld (2026)

# Robot motion as an action




Magenta trajectories describe proposed robot-point motion.

### Proposed commands

Joint configurations over the prediction horizon.

### Known robot geometry

Forward kinematics places robot surface points.

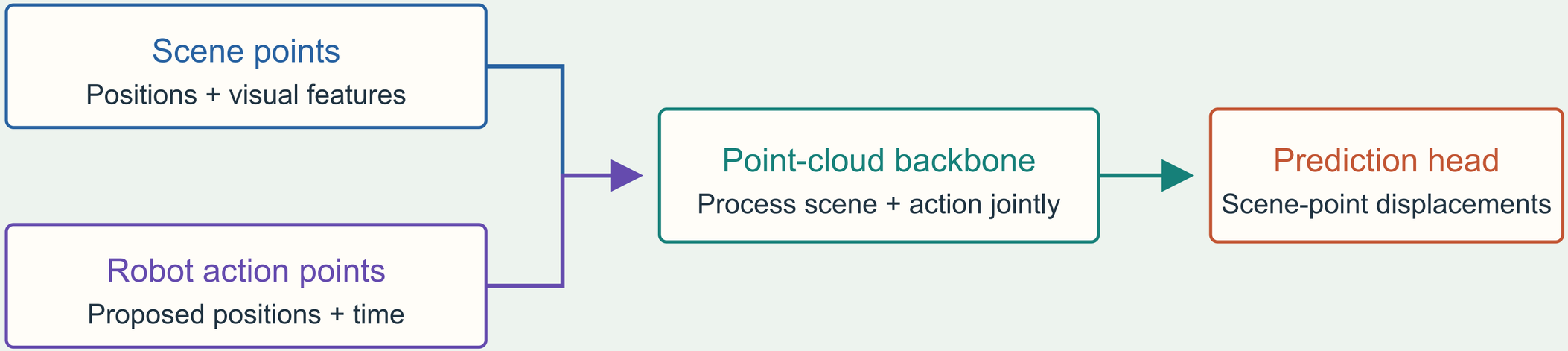
### Action input

Their 3D trajectories are supplied to the model.

**Robot motion is specified. The environment’s response must be predicted.**

Huang et al., PointWorld (2026)

# Predicting the scene response



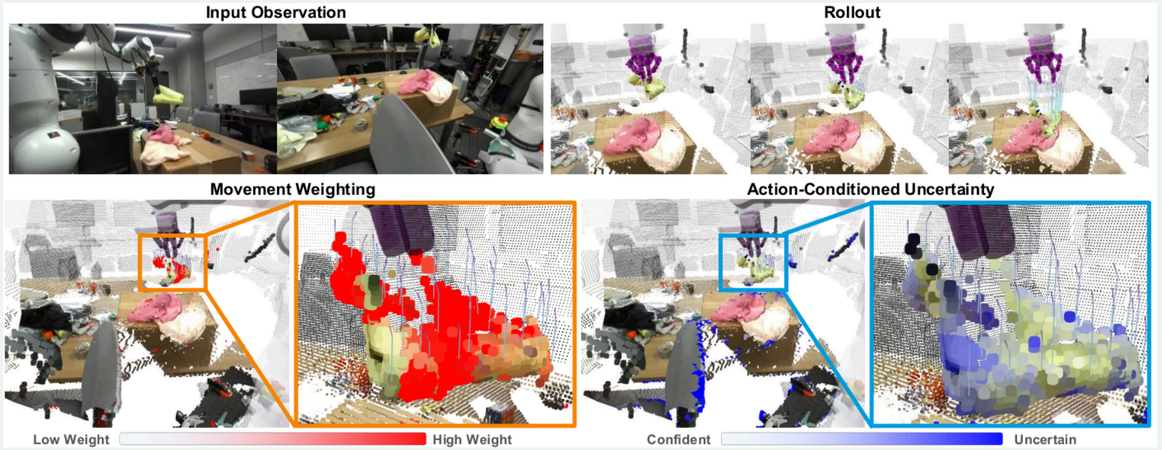

**Scene samples** describe what is currently observed.

**Action samples** describe what the robot is proposed to do.

**One forward pass predicts a short chunk of scene motion. Reobserve before the next planning step.**

Huang et al., PointWorld (2026)

# Training the dynamics model




Movement weighting and uncertainty estimates address imbalanced motion and unreliable targets.

$$\mathcal{L}_{\text{motion}} = \sum_{i,k} w_{i,k}\, \rho\left(\widehat{\mathbf{X}}_{i,t+k} - \mathbf{X}^{*}_{i,t+k}\right)$$

$\rho$: robust 3D residual. $w_{i,k}$: movement-based weighting.

Use valid correspondences. Give moving points enough training signal.

**The target is the actual future location of the same scene point.**

Huang et al., PointWorld (2026)

# Planning with 3D predictions

**Propose**

Candidate robot motions

→

**Predict**

Scene response for each candidate

→

**Evaluate**

Goal cost on predicted outcomes

→

**Execute**

Act briefly, observe, and repeat

The learned dynamics model can be reused with different goals.

**The model predicts consequences; the planning objective selects preferred consequences.**

Huang et al., PointWorld (2026)

# PointWorld in use



PointWorld articulated-object manipulation demonstration.

**Watch the contact, the drawer’s response, and the repeated observation–action loop.**

Huang et al., PointWorld (2026)

# Predicting deformable geometry



PointWorld demonstration of deformable-object manipulation.

**Points / mesh**

Predict future point or vertex positions.

**Volumes**

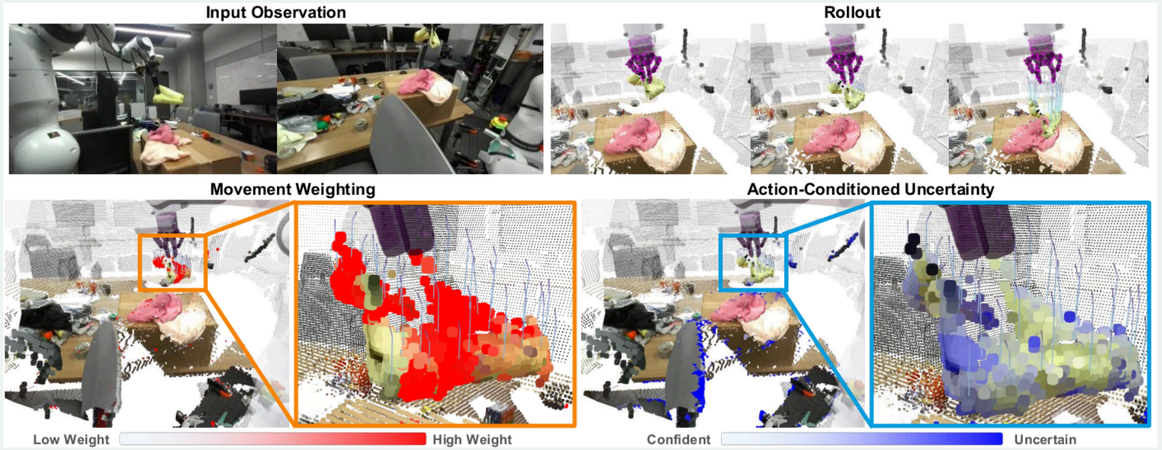
Predict future spatial attributes.

**Deformation field**

Predict a mapping of a reference representation.

**Connectivity, contact, and material properties add structure beyond visible point displacement.**

# Limits of point predictions




Occlusion, target noise, and action-dependent variability remain visible in the learning problem.

## Partial observation

Some geometry and physical properties are hidden.

## Temporal scope

Correspondence is maintained within a prediction chunk.

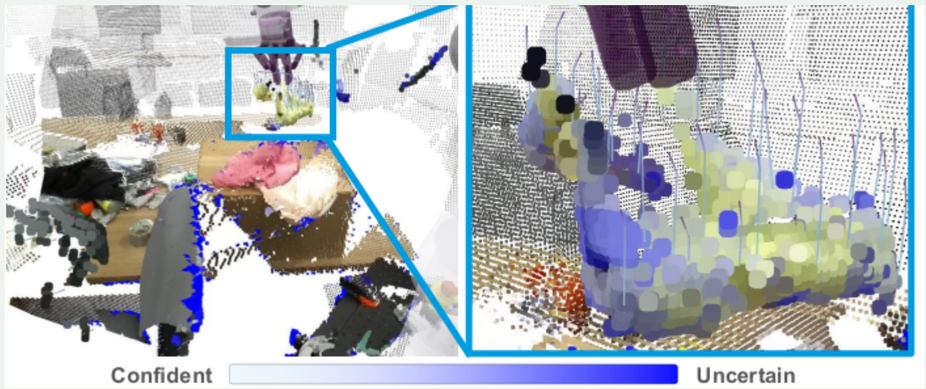
## Rollout error

Predictions and executed actions can diverge over time.

**A 3D coordinate output does not remove uncertainty or guarantee physical correctness.**

Huang et al., PointWorld (2026)

# Uncertainty in 3D futures




PointWorld illustrates action-conditioned spatial variability in its uncertainty estimates.

### Unobserved properties

Friction, compliance, hidden geometry, and contact state.

### Uncontrolled factors

Disturbances and other agents can change the outcome.

### Model and data limits

Finite experience and imperfect labels add error.

**A similar observation can support more than one plausible future.**

Huang et al., PointWorld (2026)

# Evaluating a 3D world model

| Check | Question |
| --- | --- |
| Correspondence | Does the same point move to the correct place? |
| Action response | Do different proposed actions change the forecast correctly? |
| Long rollouts | Does geometry and identity remain coherent? |
| Closed-loop use | Do predictions help decisions under real observations? |

**Visual realism, geometric accuracy, and decision usefulness need separate evidence.**

Huang et al., PointWorld (2026) · Jeon et al., Video-Point Model (October 2026)

# Representations, queries and training targets

| Route | Representation | Training asks |
| --- | --- | --- |
| Geometry recovery | Pointmaps, cameras, depth, fields | Explain observed views |
| Spatial generation | Native 3D latents, generated views, spatial memory | Generate plausible shared 3D content |
| 4D perception | Corresponding points and time-dependent geometry | Recover motion in observed video |
| Future dynamics | Scene geometry + proposed actions or instructions | Predict motion before future observations arrive |

**The training inputs and targets determine what “prediction” means.**

Wang et al., DUSt3R (2024) · Zhang et al., D4RT (2026) · Huang et al., PointWorld (2026)

# Testing spatial and temporal consistency

| What is held fixed? | What changes? | What must agree? |
| --- | --- | --- |
| One scene or recorded event | The camera viewpoint | Corresponding surfaces project consistently into both views. |
| One selected surface point | The time within the observed video | The track follows that same point, including through occlusion. |
| One initial observation | The proposed robot motion | The forecast matches the measured response to that action. |

**Check a reconstruction against views, a track against correspondence, and a forecast against held-out outcomes.**

Jeon et al., Video-Point Model (October 2026) · Huang et al., PointWorld (2026)

# Open directions

| | |
|---|---|
| **Long-term spatial memory** | Maintain geometry and identity through extended exploration. |
| **Hidden structure and uncertainty** | Represent what is not yet observed and what may happen next. |
| **Physical interactions** | Connect geometry to contact, materials, and reliable action consequences. |

Scalable learning must connect perceptual data, generative priors, and closed-loop experience.

Huang et al., PointWorld (2026) · Shen et al., Lyra 2.0 (2026) · Jeon et al., Video-Point Model (October 2026)

# From point responses to physical dynamics

**PointWorld learns how scene points respond to a proposed robot motion.**

**Physical state**

Geometry, positions, and velocities

→

**Physical update**

Laws, material properties, forces, and contact constraints

→

**Next state**

Updated positions and velocities

**Neural physics: where can neural representations and learned updates enter this state-evolution loop?**