From 2D Pixels to 3D Futures

Object Detection, Pose, and Trajectory Prediction
in Modern Computer Vision

Rustin Soraki

Paul G. Allen School of Computer Science & Engineering
University of Washington

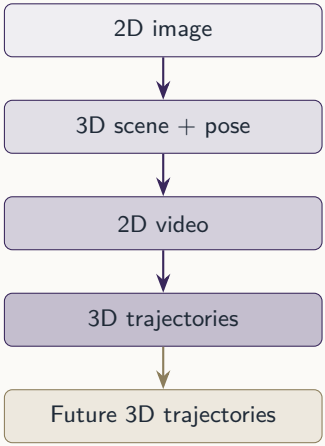
May 18, 2026

Today's Arc

1. **2D foundations:** What we already know
2. **Going to 3D:** Coordinate frames, projection, $\mathrm{SO}(3)$, 6-DoF pose
3. **Video in 2D:** Temporal structure, flow, tracking
4. **Video $\rightarrow$ 3D trajectories:** Lifting, pseudo-labels, generative forecasting
5. **Closing:** Open problems

Two papers will anchor the technical sections:

- **WildDet3D** (Ai2 / UW): *Static* promptable 3D detection
- **ObjectForesight** (UW): 3D trajectory forecasting from egocentric video

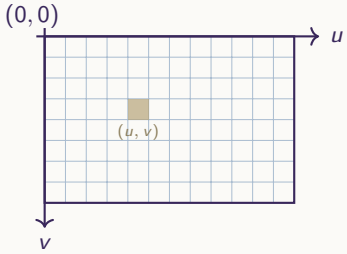


Part 1

2D Foundations

Images Live in a 2D Grid

- An image is a function $I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}^c$
- Discretized on a pixel grid $(u, v) \in \{1, \ldots, W\} \times \{1, \ldots, H\}$
- Coordinates are typically with v pointing *down* (image convention)
- For CV models, the input is a tensor $\mathbb{R}^{H \times W \times 3}$



Takeaway

Everything we build today eventually grounds out in operating on this 2D grid; even when we reason in 3D.

The Hierarchy of 2D Transformations

Transform	DoF	Preserves
Translation	2	lengths + angles
Rotation	1	lengths + angles
Rigid (SE(2))	3	lengths + angles
Similarity	4	angles
Affine	6	parallelism
Homography	8	straight lines

Homographies appear in: Panorama stitching, Planar AR, Rectification.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \underbrace{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{\mathbf{H} \in \mathbb{R}^{3\times 3}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Why homogeneous coordinates?

Linearizes translation and projection. We will lean on this heavily in 3D.

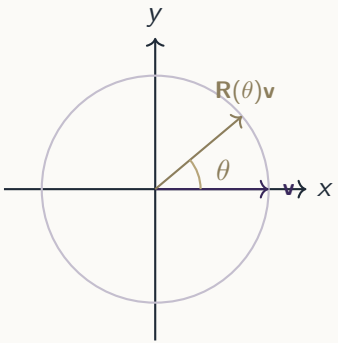
2D Rotations Are Easy

$\mathrm{SO}(2)$, the group of 2D rotations, is a *one-parameter* group every rotation is fixed by a single angle θ:

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \quad \theta \in [0, 2\pi)$$

Applied to $\mathbf{v} \in \mathbb{R}^2$, the product $\mathbf{R}(\theta)\mathbf{v}$ is just $\mathbf{v}$ spun by θ (right).

- Topologically a circle S^1
- Composition adds angles: $\mathbf{R}(\theta_1)\mathbf{R}(\theta_2) = \mathbf{R}(\theta_1 + \theta_2)$
- Single parameter, no singularities
- Distance: $d(\theta_1, \theta_2) = |\theta_1 - \theta_2|_{2\pi}$



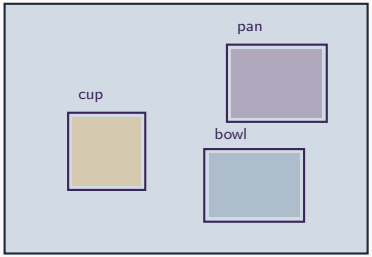
In 3D, none of this niceness will survive.

Detection in 2D: A Compact Output Space

Object detection in 2D returns:

$$\mathcal{D} = \{(c_i, \mathbf{b}_i, s_i)\}_{i=1}^{N}$$

- c_i: Class label
- $\mathbf{b}_i = (x, y, w, h)$: Axis-aligned bbox
- s_i: Confidence



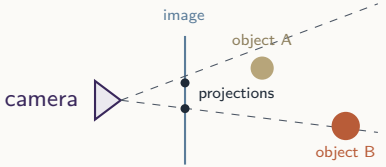
4 numbers per box. Output space is well-behaved: pixel coordinates, image-space scale.

This compactness is precisely what we lose in 3D.

Why 2D Isn't Enough

The 2D world view breaks down whenever:

- A robot needs to **grasp** an object (needs 3D position + orientation)
- AR overlays need to be **anchored in space**
- Two objects at different depths share pixels (occlusion is a 3D phenomenon)
- We want to **predict the future** (motion is in 3D, not in pixels)



A pixel can't tell you which object it came from.

The fundamental gap

Pixels are a *projection*. To act in the world, we need to invert that projection, at least approximately.

Part 2

Going to 3D

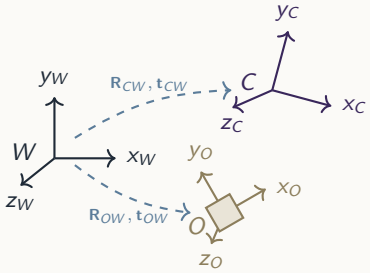
Three Coordinate Frames

- **World frame** W: Fixed, arbitrary origin
- **Camera frame** C: Origin at optical center, z along viewing direction
- **Object frame** O: Attached to each object's canonical pose

A point $\mathbf{p}$ has different coordinates in each:

$$\mathbf{p}^{(C)} = \mathbf{R}_{CW}\,\mathbf{p}^{(W)} + \mathbf{t}_{CW}$$

Subscript $_{CW}$ reads: "from W to C."



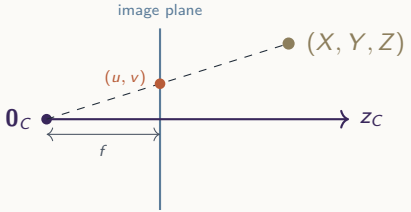
The Pinhole Camera

For a point $\mathbf{p}^{(C)} = (X, Y, Z)$ in the camera frame:

$$u = f_x \frac{X}{Z} + c_x, \quad v = f_y \frac{Y}{Z} + c_y$$

In matrix form:

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$



Projection collapses depth: Many 3D points map to the same pixel.

f_x, f_y: Focal lengths in pixels; (c_x, c_y): Principal point;

$\mathbf{K}$ is the **intrinsics matrix**.

Q: *Why is Z in the denominator? What does that imply as a point moves away?*

Intrinsics + Extrinsics: The Full Projection

For a world point $\mathbf{p}^{(W)} = (X_w, Y_w, Z_w)$:

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{pixel}} \propto \underbrace{\mathbf{K}}_{\substack{\text{intrinsics} \\ 3\times3}} \underbrace{\begin{bmatrix} \mathbf{R} \mid \mathbf{t} \end{bmatrix}}_{\substack{\text{extrinsics} \\ 3\times4}} \underbrace{\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}}_{\text{world point}}$$

Intrinsics $\mathbf{K}$: Properties of the camera itself

- Focal length, principal point, (skew)

Extrinsics $[\mathbf{R} \mid \mathbf{t}]$: Where the camera *is*

- Rotation $\mathbf{R} \in \mathrm{SO}(3)$, translation $\mathbf{t} \in \mathbb{R}^3$

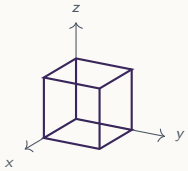
The extrinsics live in $\mathrm{SE}(3)$ built on $\mathrm{SO}(3)$, which, unlike $\mathrm{SO}(2)$, is genuinely awkward.

▷ *Right-to-left: world point → camera frame ($[\mathbf{R}|\mathbf{t}]$) → pixels ($\mathbf{K}$). The $\propto$ hides the divide-by-Z.*

3D Rotations Are Hard

$\mathrm{SO}(3)$ is the set of 3×3 matrices $\mathbf{R}$ satisfying $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$ (*orthonormal* columns preserves lengths and angles) and $\det \mathbf{R} = +1$ (right-handed; rules out reflections).

- **9 numbers but only 3 DoF** (6 orthonormality constraints)
- A 3-manifold inside $\mathbb{R}^9$
- *Non-commutative*: $\mathbf{R}_1 \mathbf{R}_2 \neq \mathbf{R}_2 \mathbf{R}_1$
- *Non-Euclidean*: no global 3-number parameterization without singularities



$\mathbf{R}_x(90^\circ)\mathbf{R}_y(90^\circ) \neq \mathbf{R}_y(90^\circ)\mathbf{R}_x(90^\circ)$

Consequence: *Every parameterization of* $\mathrm{SO}(3)$ *has tradeoffs.*

Q: *Tip your phone forward, then yaw left. Reverse the order. Same pose?*

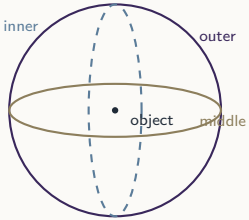
Rotation Representations I: Matrices and Euler Angles

Rotation matrices $\mathbf{R} \in \mathbb{R}^{3\times3}$

- \+ No singularities, easy composition (matrix mult)
- – 9 parameters with 6 constraints: Bad for learning
- – Loss must enforce orthogonality

Euler angles (α, β, γ)

- \+ Just 3 numbers, intuitive (roll/pitch/yaw)
- – Many conventions (XYZ, ZYX, intrinsic, extrinsic. . .)
- – **Gimbal lock** loses a DoF at $\beta = \pm\pi/2$
- – Discontinuous and ill-conditioned



When two rings align, one DoF disappears.

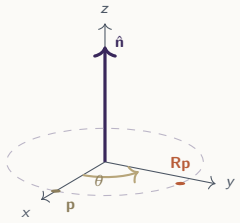
Rotation Representations II: Axis-Angle

Every rotation in 3D fixes some axis. The pair $(\hat{\mathbf{n}}, \theta)$ unit axis $\hat{\mathbf{n}} \in S^2$, rotation angle θ names a rotation directly. Packed as a single 3-vector:

$$\mathbf{r} = \theta\,\hat{\mathbf{n}} \in \mathbb{R}^3$$

Rodrigues' formula reconstructs the matrix ($[\hat{\mathbf{n}}]_\times$ is the cross-product matrix):

$$\mathbf{R} = \mathbf{I} + \sin\theta\,[\hat{\mathbf{n}}]_\times + (1 - \cos\theta)\,[\hat{\mathbf{n}}]_\times^2$$



- \+ 3 numbers, no gimbal lock
- – Singularity at $\theta = 0$ (axis undefined)
- – Non-unique at $\theta = \pm\pi$ ($\hat{\mathbf{n}}$ and $-\hat{\mathbf{n}}$ agree)

Rotation Representations III: Unit Quaternions

Unit quaternions live on the 4-sphere:

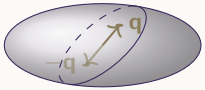
$$\mathbf{q} = (w,\, x,\, y,\, z) \in S^3, \quad \|\mathbf{q}\| = 1$$

The same axis/angle, lifted via a half-angle:

$$\mathbf{q} = \cos\tfrac{\theta}{2} \;+\; \sin\tfrac{\theta}{2}\,(n_x\, i + n_y\, j + n_z\, k)$$

Apply: $\mathbf{Rp}$ is computed by $\mathbf{q}\,\mathbf{p}\,\mathbf{q}^{-1}$ (with $\mathbf{p}$ embedded as a pure imaginary).

- \+ 4 numbers, 1 unit-norm constraint
- \+ Composition is the Hamilton product $\mathbf{q}_1 \otimes \mathbf{q}_2$
- \+ Smooth interpolation via SLERP
- − **Double cover**: $\mathbf{q}$ and $-\mathbf{q}$ are the same rotation



S^3: unit-norm 4-vectors

The 6D Continuous Representation

Zhou et al., CVPR 2019 continuity theorem

Any rep of $\mathrm{SO}(3)$ with ≤ 4 reals is *discontinuous*. Networks regressing them hit a **Lipschitz problem**.

Idea: represent $\mathbf{R}$ by its first two columns.

$$\mathbf{R} = [\mathbf{r}_1 \mid \mathbf{r}_2 \mid \mathbf{r}_3] \longmapsto (\mathbf{r}_1, \mathbf{r}_2) \in \mathbb{R}^6$$

Decoding via Gram–Schmidt (normalize, orthogonalize, cross-product):

$$\mathbf{r}'_1 = \frac{\mathbf{r}_1}{\|\mathbf{r}_1\|}, \quad \mathbf{r}'_2 = \frac{\mathbf{r}_2 - (\mathbf{r}'^{\top}_1 \mathbf{r}_2)\mathbf{r}'_1}{\|\cdots\|}, \quad \mathbf{r}'_3 = \mathbf{r}'_1 \times \mathbf{r}'_2$$

A continuous embedding $\mathrm{SO}(3) \hookrightarrow \mathbb{R}^6$; dominant in modern pose regression.

Practical recipe

1. Network outputs $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{R}^3$
2. Gram–Schmidt for orthonormal basis
3. Build $\mathbf{R}$; Frobenius or geodesic loss

▷ *Topology: Can't continuously embed* $\mathrm{SO}(3)$ *in* < 5*D. 6D leaves room.*

6-DoF Pose: Putting It Together as $\mathrm{SE}(3)$

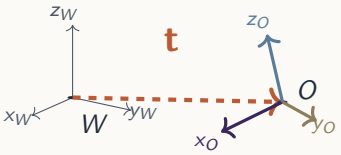
A rigid-body pose is a rotation plus a translation:

$$g = (\mathbf{R}, \mathbf{t}) \in \mathrm{SE}(3), \quad \mathbf{R} \in \mathrm{SO}(3),\ \mathbf{t} \in \mathbb{R}^3$$

Body-to-world: Apply $\mathbf{R}$, then add $\mathbf{t}$; equivalently, in homogeneous coords:

$$\mathbf{p}^{(\text{world})} = \mathbf{R}\,\mathbf{p}^{(\text{body})} + \mathbf{t} \iff \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}^{(w)} = \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}}_{\mathbf{T} \in \mathbb{R}^{4\times 4}} \begin{bmatrix} \mathbf{p} \\ 1 \end{bmatrix}^{(b)}$$

- **6 DoF** = 3 rotation + 3 translation
- Composition: $\mathbf{T}_1\mathbf{T}_2$ (matrix mult)
- Tangent space at identity: $\mathfrak{se}(3) \cong \mathbb{R}^6$
- Geodesic distance: $d(g_1, g_2)^2 = \|\log(g_1^{-1} g_2)\|^2$

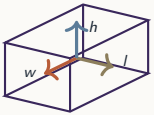


3D Detection: A 9-Parameter Output

3D oriented bounding box, per object:

$$\underbrace{(c)}_{\text{class}} + \underbrace{(t_x, t_y, t_z)}_{\text{center}} + \underbrace{(\mathbf{R} \in \mathrm{SO}(3))}_{\text{orientation}} + \underbrace{(w, h, l)}_{\text{extents}}$$

$\Rightarrow$ **9 continuous DoF** per object
(3 translation + 3 rotation + 3 size)



- Compared to 2D's 4 numbers
- Output space is no longer flat
- Loss design now matters

And from a single image, this is genuinely under-determined.

Q: *Going from 4 numbers to 9, what new prior information must our model bring?*

Paper 1: WildDet3D: Scaling Promptable 3D Detection

** Huang, Zhang, Li, . . . , **Soraki**, . . . , Farhadi, Hariharan, Krishna. Ai2 & UW, 2026.*

Problem. Monocular 3D object detection that works *in the wild*:

- Open-vocabulary (not a fixed list of classes)
- Promptable (text, point click, or 2D box)
- Optionally uses depth signal when available

Two contributions:

1. **Architecture**: Unified geometry-aware model accepting text / point / box prompts; ingests optional depth
2. **Data**: WildDet3D-Data: >1M images, 13.5K categories, human-verified 3D boxes

Fig. 1 from the paper

WildDet3D: Architecture

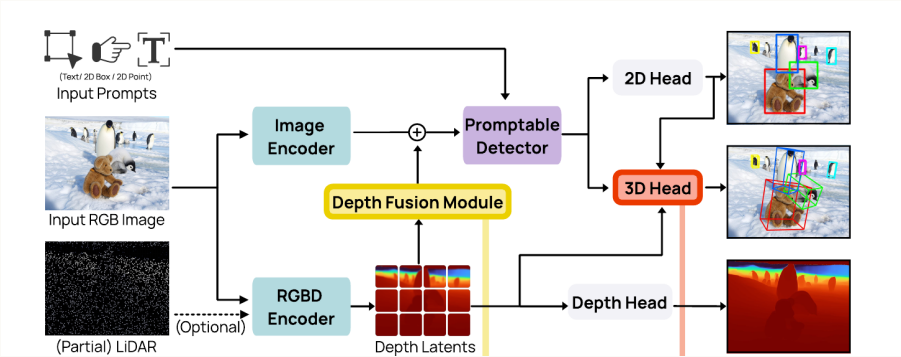


Fig. 3 from the paper

- Parallel **image encoder** (RGB) and **RGBD encoder** (optional depth/LiDAR input); depth latents fused into image features via a learned gating module
- Cascaded **2D head** and **3D head**; depth is folded in at *inference time* as optional
- Camera ray embeddings + depth latents condition the 3D head

WildDet3D: Building the Dataset

Prior 3D detection datasets cover narrow categories in controlled scenes.

WildDet3D-Data: **1M+ images, 13.5K categories** via candidate-and-verify

1. Generate candidate 3D boxes from existing 2D annotations using multiple methods
2. Have humans **verify or reject** the 3D proposals

Fig. 5: WildDet3D-Data examples spanning indoor / urban / nature

WildDet3D: Scene Diversity

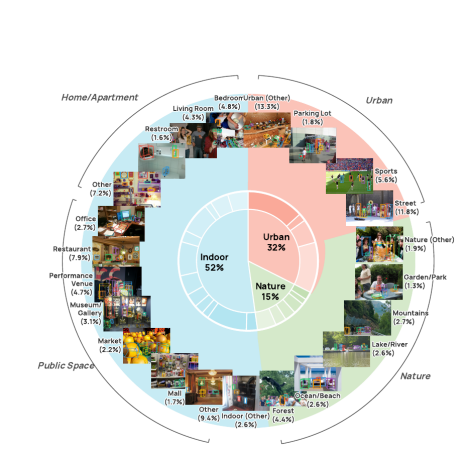


Fig. 6 from the paper

Three macro-categories:

- **Indoor** (52%) bedrooms, living rooms, offices, restaurants, markets, malls
- **Urban** (32%) streets, sports, parking lots
- **Nature** (15%) forests, beaches, lakes, mountains

Pattern

Generate cheap labels, gate them with a human or VLM. The same recipe shows up later in the other work.

WildDet3D: Results

Benchmarks:

- **WildDet3D-Bench** (new, open-world): 22.6 / 24.8 AP_{3D} (text / box); **41.6 / 47.2** AP_{3D} with GT depth
- **Omni3D**: 34.2 / 36.4 AP_{3D} (trained *12 epochs* vs. 80120 for baselines)
- **Zero-shot**: Argoverse 2 & ScanNet: 40.3 / 48.9 ODS
- Adding sparse depth at inference: +20.7 AP avg

Fig. 8: WildDet3D (top) vs. 3D-MOOD (bottom), text prompts only

Takeaway

A single promptable monocular model is competitive across very different domains *when you have the data*. Depth, when available, is a major lever.

WildDet3D: Real-World Applications

(b) Augmented Reality (Meta Quest 3): passthrough AR with 3D bounding boxes rendered in real time across three different desk scenes.

(c) Robotics (manipulation): Franka Emika Panda autonomously grasping objects specified by open-vocabulary text prompts ("Green Chips", "Can", "Orange").

Figure 9 Real-world deployment demos. Each row shows three frames from a different deployment platform, demonstrating WildDet3D across diverse interaction modes and environments.

Fig. 9: Meta Quest 3 AR (top), Franka Panda manipulation (bottom)

- **iOS app** on App Store; AR passthrough on Meta Quest 3 with live 3D boxes
- Franka robot grasps objects via open-vocab text prompts
- Open-vocab 3D detection *is* the perception module robots / AR need

Part 3

Video in 2D

Video as a Tensor

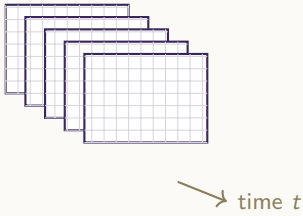
A video is just an extra axis on top of an image:

$$V \in \mathbb{R}^{T \times H \times W \times 3}$$

- Strong **temporal redundancy**: Adjacent frames are similar
- Motion is the *differential* signal
- Scaling: a 30-second clip at 30 fps is $\sim$900$\times$ more pixels than a single image

Two complementary views:

- Frame-by-frame (extends 2D methods)
- **Per-point tracks** over time (motion-centric)



From Optical Flow to Point Tracking

Optical flow: Dense pairwise 2D motion field

$$\mathbf{u}(x, y) \in \mathbb{R}^2, \quad \text{per pixel, per frame pair}$$

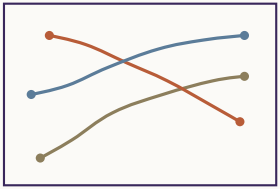
Local, brittle under occlusion, no long-range correspondence.

Point tracking (TAP, CoTracker)

- Track *specific query points* across an entire clip
- Handles occlusion (predicts visibility)
- Output: 2D trajectories $\{(u_t^i, v_t^i)\}$

The shift

From *dense local* flow to *sparse long-range* trajectories.



Long-range 2D tracks

2D Video Object Tracking

Tracking-by-detection

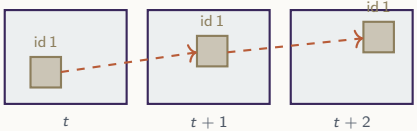
1. Run 2D detector per frame
2. Associate boxes across time (Hungarian, ReID)
3. Smooth with Kalman / motion model

Joint detection-and-tracking

- Transformer-based, end-to-end (TrackFormer, MOTR)
- Tracks as queries that persist over time

Segmentation tracking (SAM 2)

- Per-pixel masks across time
- Used as a building block by both papers today



Same ID maintained across frames ⇒ the track.

Temporal Architectures

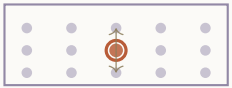
3D convolutions *classical*
Kernels of shape $t \times k \times k$. Quadratic in time. *I3D, S3D.*

Factorized attention *modern*

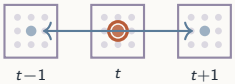
- Spatial attention within frames
- Temporal attention across frames
- TimeSformer, ViViT, Video Swin

Joint space-time attention
Expensive but expressive. Needed when objects move significantly.

Spatial: Within a single frame



Temporal: Same cell across frames



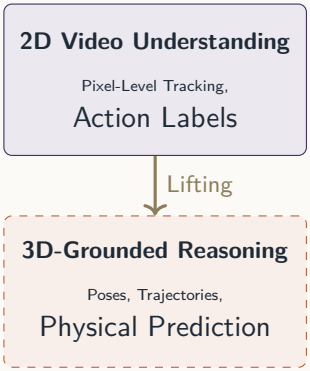
What 2D Video Gives Us - And What's Missing

We can already do:

- Robust 2D detection across time
- Long-range point tracks
- Action recognition
- Video segmentation

We still can't (in 2D):

- Know *where* something is in metric space
- Predict where it *will be* in metric space
- Reason about contact, dynamics, affordances
- Provide a robot a target pose



Part 4

Video to 3D Trajectories

The Lifting Problem

$$\underbrace{\text{2D pixels over time}}_{\text{what we observe}} \xrightarrow{?} \underbrace{\text{3D object poses over time}}_{\text{what we want}}$$

Three things must be recovered:

1. **Depth**: Per frame (per pixel or per object)
2. **Camera motion**: Between frames (ego-motion)
3. **Object motion**: Relative to the world

Same image
Same image, different 3D

Scale ambiguity

A monocular video is consistent with infinitely many 3D explanations differing by an overall scale. Resolving this needs either a known size, depth prior, or a metric depth model.

Building Blocks for Lifting

Monocular metric depth

- Depth Anything V2, Metric3D, UniDepth
- Per-frame depth maps, increasingly metric

Camera motion / SLAM

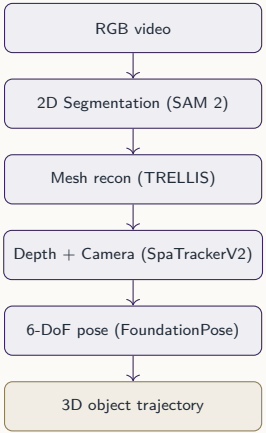
- DROID-SLAM, MonST3R, SpaTrackerV2
- Joint per-pixel depth + camera pose

Object segmentation & mesh

- SAM 2 for masks over time
- TRELLIS / one-shot reconstruction for meshes

6-DoF pose tracking

- FoundationPose, MegaPose
- Given a mesh + RGB(D), estimate pose per frame

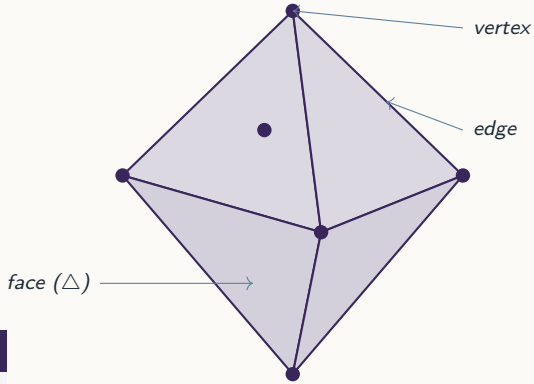


What is a 3D Mesh?

A 3D mesh $\mathcal{M} = (V, E, F)$ is a discrete representation of a surface.

- **Vertices** $V \subset \mathbb{R}^3$ points in 3D space
- **Edges** $E \subseteq V \times V$ pairs of connected vertices
- **Faces** F usually triangles (sometimes quads)

Why this representation? Compact, sparse, real-time renderable; the canonical interface for graphics, physics, and robotics tooling.



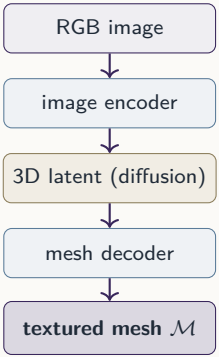
Mesh vs. alternatives

Point clouds (no connectivity), *voxels* (grids), *SDFs / NeRFs* (implicit fields), *Gaussian splats* (volumetric primitives). Meshes remain the standard interface for downstream tools.

Building Block: Image-to-3D Mesh Models

Task. Given a single (or few) RGB image(s) of an object, produce a 3D mesh.

- Recent breakthroughs: **TRELLIS**, **Hunyuan3D**, **InstantMesh**, **SAM 3D**
- Image encoder $\rightarrow$ latent 3D rep $\rightarrow$ decoder to mesh / Gaussian splat / SDF
- Most are diffusion-based on a structured 3D latent



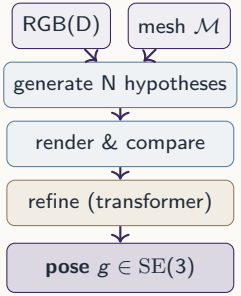
Building Block: 6-DoF Pose Estimation

Task. Given an RGB(D) image + object model, estimate $g = (\mathbf{R}, \mathbf{t}) \in \mathrm{SE}(3)$. **Two flavors:**

- **Model-based** known mesh, render-and-compare
  FoundationPose, MegaPose
- **Model-free** only reference images
  Gen6D, OnePose

FoundationPose recipe:

1. Detect / segment the object
2. Generate N pose hypotheses
3. Render each, compare via transformer
4. Refine, score, pick the best



From 2D Tracks to 3D Trajectories

Given a 2D point track (u_t, v_t) and a depth map D_t:

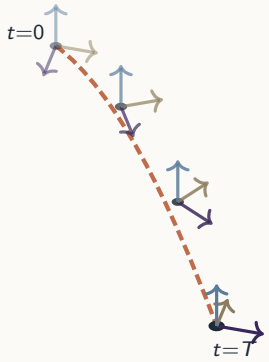
$$\mathbf{p}_t^{(C)} = D_t(u_t, v_t)\,\mathbf{K}^{-1}\begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix}$$

Then convert to world frame using camera pose $\mathbf{T}_{WC}^{(t)}$:

$$\mathbf{p}_t^{(W)} = \mathbf{R}_{WC}^{(t)}\mathbf{p}_t^{(C)} + \mathbf{t}_{WC}^{(t)}$$

For an object (not just a point), we estimate $\mathbf{T}_{WO}^{(t)} \in \mathrm{SE}(3)$ per frame; the sequence $\{\mathbf{T}_{WO}^{(0)}, \ldots, \mathbf{T}_{WO}^{(T)}\}$ is a 6-DoF *trajectory*.

▷ *First eq right-to-left: pixel → ray (via $\mathbf{K}^{-1}$) → 3D point.*



A 6-DoF trajectory: position and orientation per frame.

Designing Losses on Trajectories

We want *accurate* poses at every frame and *smooth* motion across frames.

1. **Per-step pose loss**: Translation in metric space, rotation on $\mathrm{SO}(3)$ (geodesic or 6D-rep); λ balances scales:

$$\mathcal{L}_{\text{pose}} = \frac{1}{T}\sum_t \underbrace{\|\mathbf{t}_t - \mathbf{t}_t^{gt}\|^2}_{\text{translation}} + \lambda \cdot \underbrace{d_{\mathrm{SO}(3)}(\mathbf{R}_t, \mathbf{R}_t^{gt})^2}_{\text{rotation}}$$

2. **Trajectory smoothness**: Penalize jerky motion via discrete acceleration (ω_t: frame-to-frame angular velocity):

$$\mathcal{L}_{\text{smooth}} = \sum_t \underbrace{\|\mathbf{t}_{t+1} - 2\mathbf{t}_t + \mathbf{t}_{t-1}\|^2}_{\text{translation accel.}} + \mu \cdot \underbrace{\|\omega_t - \omega_{t-1}\|^2}_{\text{angular accel.}}, \quad \omega_t = \log(\mathbf{R}_t^{-1}\mathbf{R}_{t+1}) \in \mathfrak{so}(3)$$

Pose token tricks. Many papers use a 9D token: 6D rotation + 3 translation, often *depth-normalized* for numerical stability.

Q: *If λ, μ are wrong, what fails first?*

Why Generative Models for Trajectories?

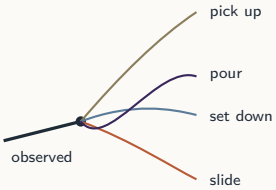
Future trajectories are **inherently multi-modal**.

Given the same first 1 second of a cup being picked up, the next 2 seconds could be:

- Drinking
- Pouring into another vessel
- Setting it down somewhere else
- Sliding it across a table

A regression model averaging these, outputs **none of them**.

Diffusion / flow matching: Model the *distribution* of futures.



Paper 2: ObjectForesight

Predicting Future 3D Object Trajectories from Human Videos.

**Soraki, Bharadhwaj, Farhadi, Mottaghi. 2026.*

Goal. Given a short egocentric video and an object mesh, predict the object's **future 6-DoF trajectory**.

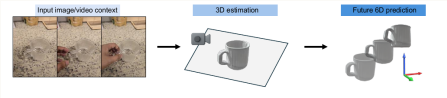


Fig. 1 from the paper

Why this matters:

- Embodied AI / robotics: Anticipate object motion
- Object-centric world model: 3D, not pixels
- Trained on *passive* human videos

Three contributions.

1. Formalize the task
2. Geometry-aware diffusion model
3. 2M+ pseudo-labeled trajectory dataset

ObjectForesight: Problem Setup

Input:

- Short history of egocentric RGB frames $\{I_{-P}, \ldots, I_0\}$
- Object mesh $\mathcal{M}$ (e.g. from one-shot recon)
- Object mask in the anchor frame
- Past poses $\{g_{-P}, \ldots, g_0\} \subset \mathrm{SE}(3)$

Output:

$$p_\theta(g_1, g_2, \ldots, g_H \mid \text{history})$$

A **distribution over future 6-DoF pose sequences** over horizon H (history length P, anchor at $t = 0$).

Why anchor-frame canonicalization?

Pose trajectories are predicted relative to the anchor frame; making the prediction problem invariant to where the camera happened to be.

ObjectForesight: Architecture

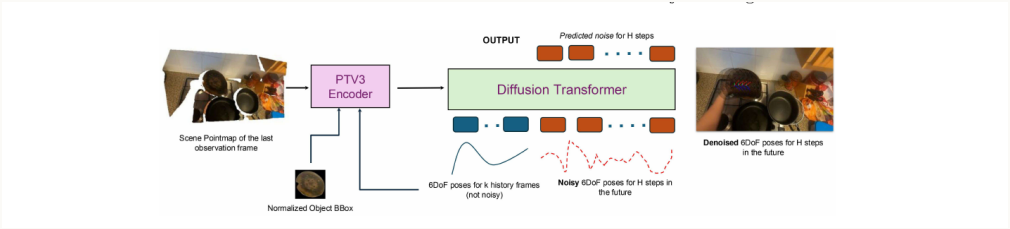


Fig. 3 from the paper

- **PointTransformerV3 encoder** produces a 3D-aware scene embedding from the anchor-frame point cloud
- Past poses + normalized object bboxes → context prefix; **Diffusion Transformer** denoises noisy future pose tokens conditioned on the scene
- Operates on **depth-normalized 9D pose tokens** (translation + 6D rotation rep); DDIM sampling at inference

ObjectForesight: Data Curation

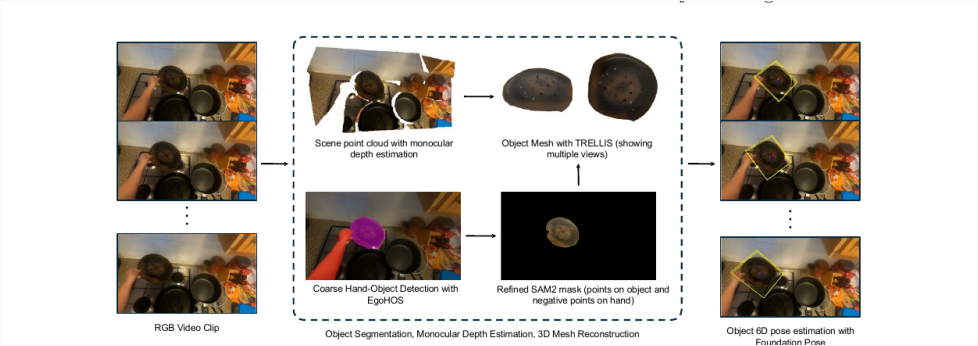


Fig. 2 from the paper

- EgoHOS → SAM 2 → VLM gating for manipulation + viewpoint quality
- TRELLIS for mesh, SpaTrackerV2 for metric depth + camera
- FoundationPose with bidirectional tracking + re-registration → **2M+ trajectories**

ObjectForesight: Qualitative Results

Two-domain evaluation:

- **HOT3D** lab-quality 3D, sanity check
- **Epic-Kitchens** in-the-wild, harder

What you should see in the qualitative videos:

- Plausible *multi-modal* futures from the same context
- Temporally coherent rotations, not just translations
- Generalizes to objects unseen during training

Compared to Luma Ray 3 + post-hoc reconstruction:

- Direct 3D prediction is more stable
- No appearance hallucinations to filter out

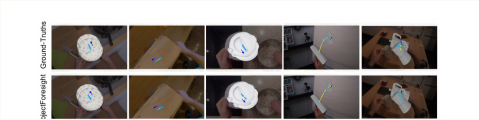


Fig. 5: top=ground truth, bottom=predictions

ObjectForesight: Quantitative Results

	ADE ↓	FDE ↓	DES ↓	ARE ↓	FRE ↓	RES ↓
Epic-Kitchens (in-the-wild)						
Constant Velocity	0.027	0.053	0.007	2.47°	5.60°	0.80°
ObjectForesight-AR	0.067	0.074	**0.002**	9.48°	12.58°	0.93°
ObjectForesight-DiT	**0.016**	**0.029**	0.004	**2.30°**	**4.82°**	**0.66°**
vs. Video Generation (subset)						
Luma AI Ray3	0.084	0.149	0.020	12.86°	20.90°	2.62°
ObjectForesight-DiT	**0.029**	**0.059**	**0.008**	**7.29°**	**13.98°**	**1.77°**
HOT3D-Clips (lab-quality)						
Constant Velocity	0.136	0.280	0.04	38.70°	68.53°	9.85°
ObjectForesight-AR	0.055	0.082	0.007	9.80°	14.95°	1.55°
ObjectForesight-DiT	**0.021**	**0.026**	**0.003**	**8.92°**	**12.58°**	**1.16°**

Takeaway

−41% ADE vs. Constant Velocity; $\sim 3\times$ better translation than Luma Ray3. *The diffusion formulation matters* the AR variant underperforms even constant velocity.

ObjectForesight: Why an Object-Centric 3D Model?

Pixel-space world models

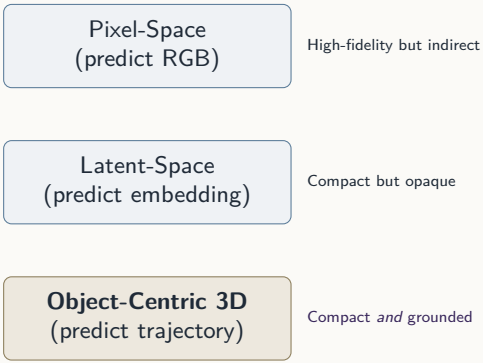
- Predict future frames as RGB
- Strong appearance prior; weak geometry
- Need to *re-extract* 3D after the fact

Latent-space world models

- Predict in an opaque embedding
- Hard to ground in physical actions

ObjectForesight: object-centric 3D

- Output *is* a 6-DoF trajectory
- Geometrically grounded, robot-actionable
- Decouples *what* from *how it looks*



Part 5

Closing

Recap: The Arc

Concepts in sequence

1. **2D foundations**: Pixels, bboxes, $\mathrm{SO}(2)$
2. **Going to 3D**: $\mathrm{SE}(3)$, quaternions, 6D rep, projection
3. **Video in 2D**: Tracking, point tracks, temporal architectures
4. **Video → 3D trajectories**: Lifting, pseudo-labels, generative forecasting

The conceptual move

Each step adds one form of structure:
2D → 3D adds *geometry*; image → video adds *time*; observation → prediction adds *uncertainty*.

Two anchor papers

WildDet3D

Single-image, promptable, in-the-wild 3D detection

Anchored Part 2

ObjectForesight

Future 6-DoF object trajectories from egocentric video

Anchored Part 4

Open Problems

Geometry

- Genuine *metric* 3D from monocular video
- Articulated and deformable objects (we did rigid today)
- Contact and physics, not just kinematics

Learning

- Closing the loop with action *prediction-as-policy*?
- Multi-object trajectories with interactions
- Long-horizon coherence

Data

- Pseudo-labels are great; what's missing?
- Cross-embodiment: human video $\rightarrow$ robot use

Representation

- Continuous representations beyond 6D
- Equivariance to camera frame more aggressively
- Unified models across detection, pose, tracking, prediction

Thank you!

Questions?

rustin@cs.washington.edu

Papers

ObjectForesight: arxiv.org/abs/2601.05237 • WildDet3D: arxiv.org/abs/2604.08626