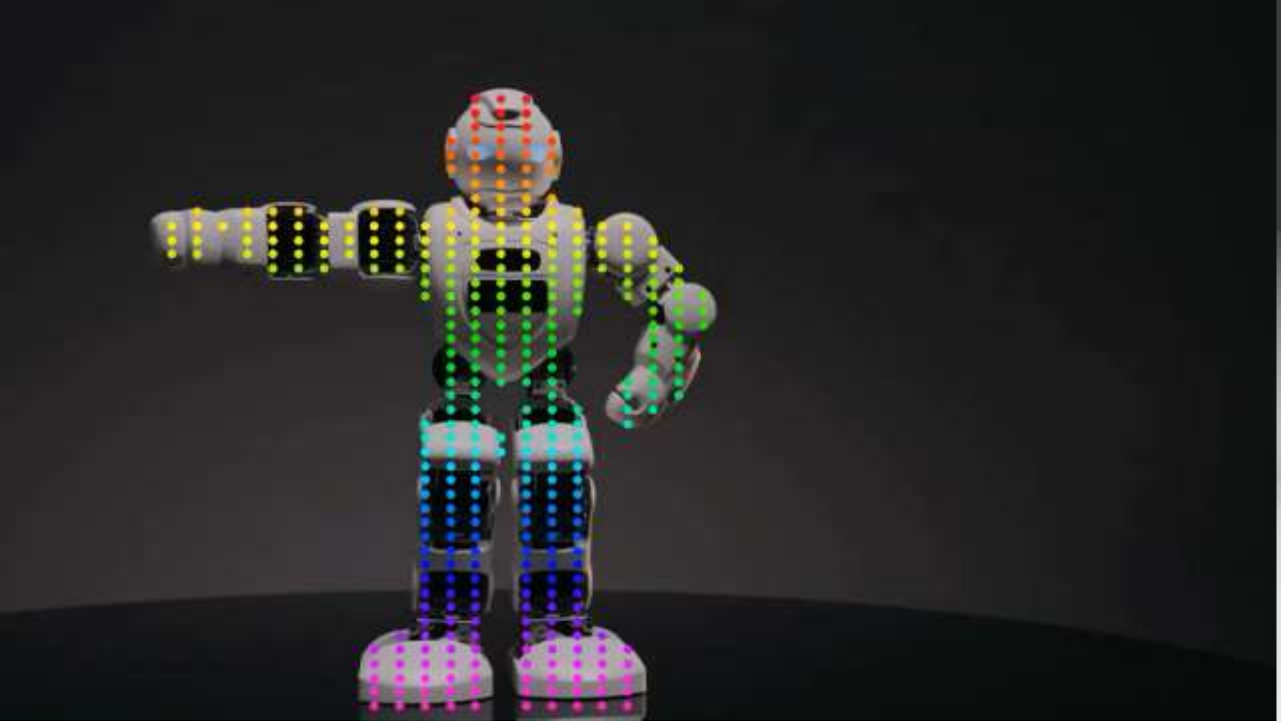
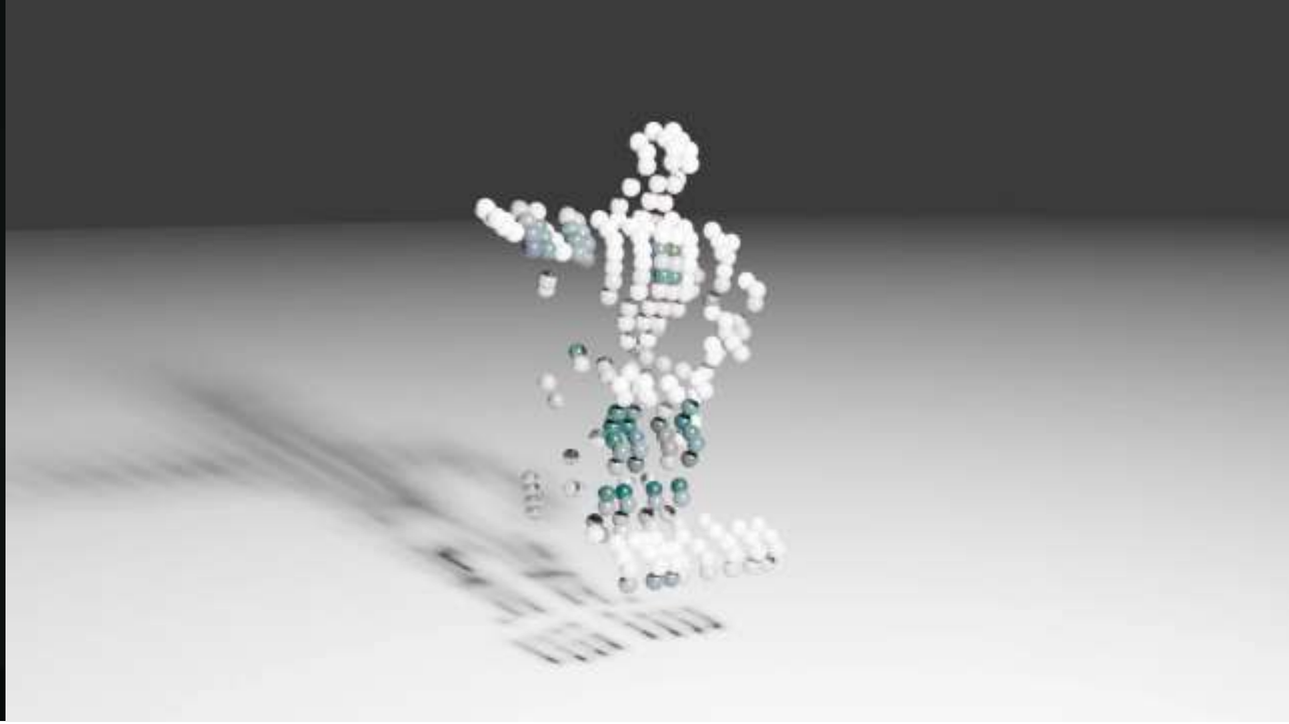




Spatial Tracker: Tracking Any 2D Pixels in 3D Space

A novel Framework for Dense and Long- Range Motion Estimation.

Presented By
Kishore Reddy Mamidi
Mohammed Fazil Khasim
Alex Chilaka

Why Does Pixel Tracking Fail in Real Life?

- 📷 Cameras capture the **3D world as flat 2D images**.
- 🔍 When an object **rotates, occludes**, or moves out-of-plane —
  - **Traditional 2D methods lose track.**
- 🎯 Tracking fails because:
  - **No depth awareness**
  - **Gets confused by occlusions**
  - **Struggles with long-range motion**
- 💡 **Real life happens in 3D** — so why do we still track in 2D?

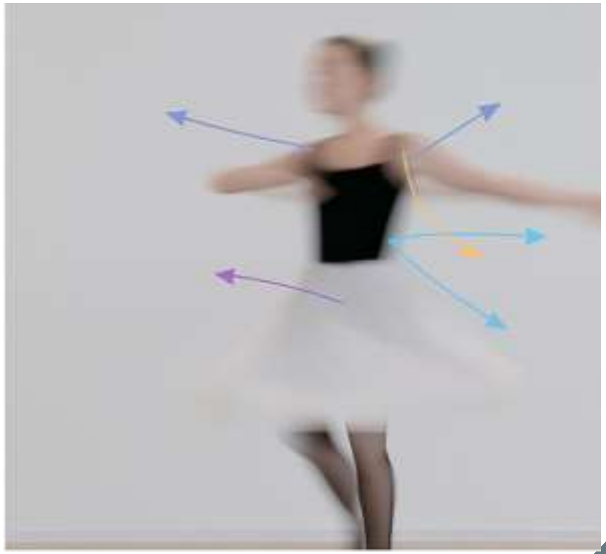
Previous Work: Optical Flow

🔍 What is Optical Flow?

- Estimates dense pixel-wise motion between two adjacent video frames
- Commonly used in video analysis, object detection, and action recognition
- Popular models: RAFT, FlowNet, PWC-Net

❌ **Limitations:**

- Only works for short-term motion
- Breaks under occlusion, large displacements, or rotation
- Cannot reason about depth or 3D structure
- Treats motion as purely 2D — leads to incorrect tracking in real-world scenes



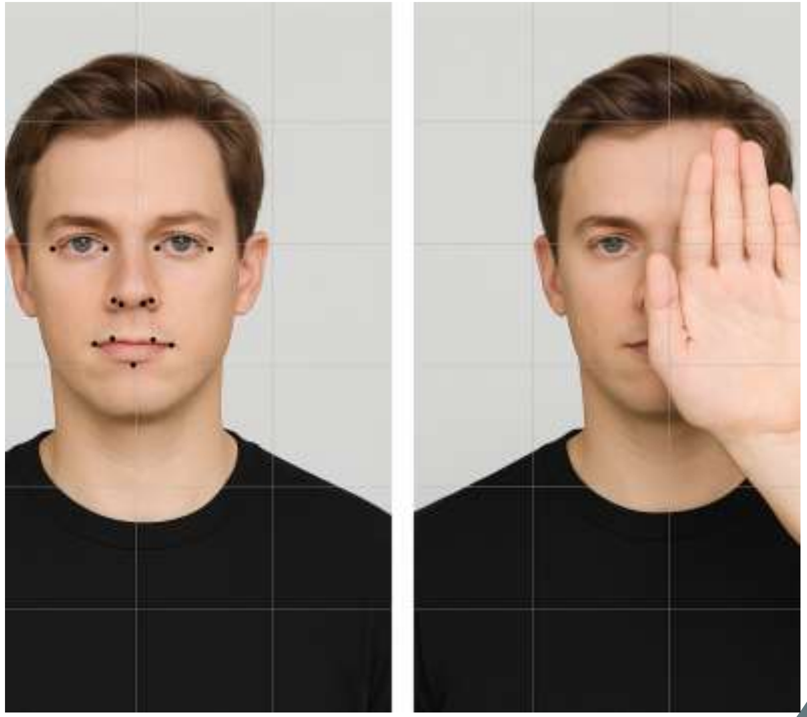
Previous Work: Point Tracking

🔍 What is Point Tracking?

- Follows a few selected key points across multiple frames
- Tracks motion over longer time spans than optical flow
- Examples: Particle Video, PIPs, TAPIR

❌ Limitations:

- Only tracks sparse points — not every pixel
- Fails if the keypoint is occluded or goes out of view
- Often treats points independently, ignoring spatial context
- Limited temporal window — can't handle long occlusion



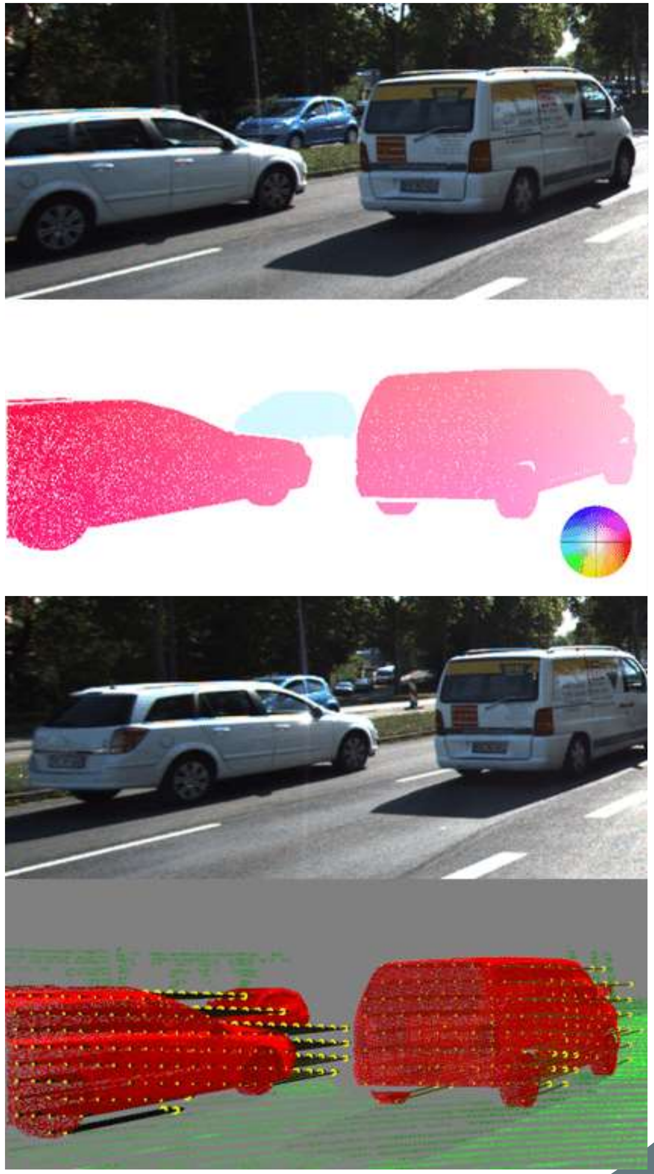
Previous Work: Scene Flow

🔍 **What is Scene Flow?**

- Estimates **dense 3D motion** of points in the scene
- Uses inputs like **RGB-D video** or **stereo camera views**
- Generates 3D trajectories across time (like optical flow, but in 3D)
- Examples: **RAFT-3D**, **Dynamic Fusion**, **FlowNet3D**

❌ **Limitations:**

- Requires **depth sensors or stereo camera setups**
- Often **slow** and **computationally expensive**
- Many models are designed only for **structured environments** (e.g., self-driving)
- Doesn't generalize well to **unconstrained or monocular video**



What is SpatialTracker?

📌 **Core Problem:**

- Pixel motion tracking in videos often fails due to:

 - Occlusion, Rotation, Complex 3D motion, Lack of depth in 2D projections

💡 **SpatialTracker's Solution:**

- Lifts 2D pixels into 3D space using depth estimation.
- Tracks motion across time using 3D trajectories.
- Uses a Transformer model to predict and refine movement.
- Applies ARAP (As-Rigid-As-Possible) constraints to handle occlusions and enforce spatial consistency.

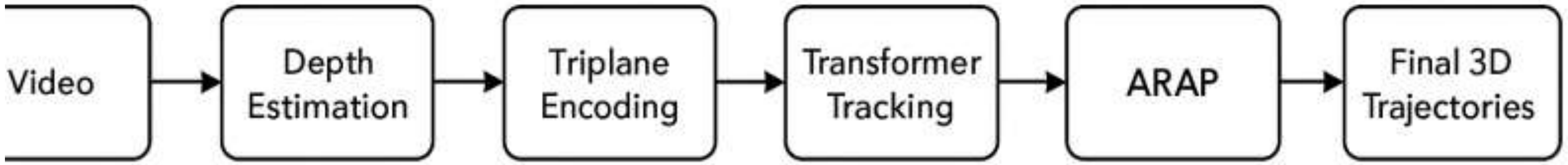
🔬 **Why It Works:**

- 3D motion is simpler, smoother, and physically meaningful.
- Avoids 2D issues like flattening and irrelevant neighboring context.

How Does SpatialTracker Work?

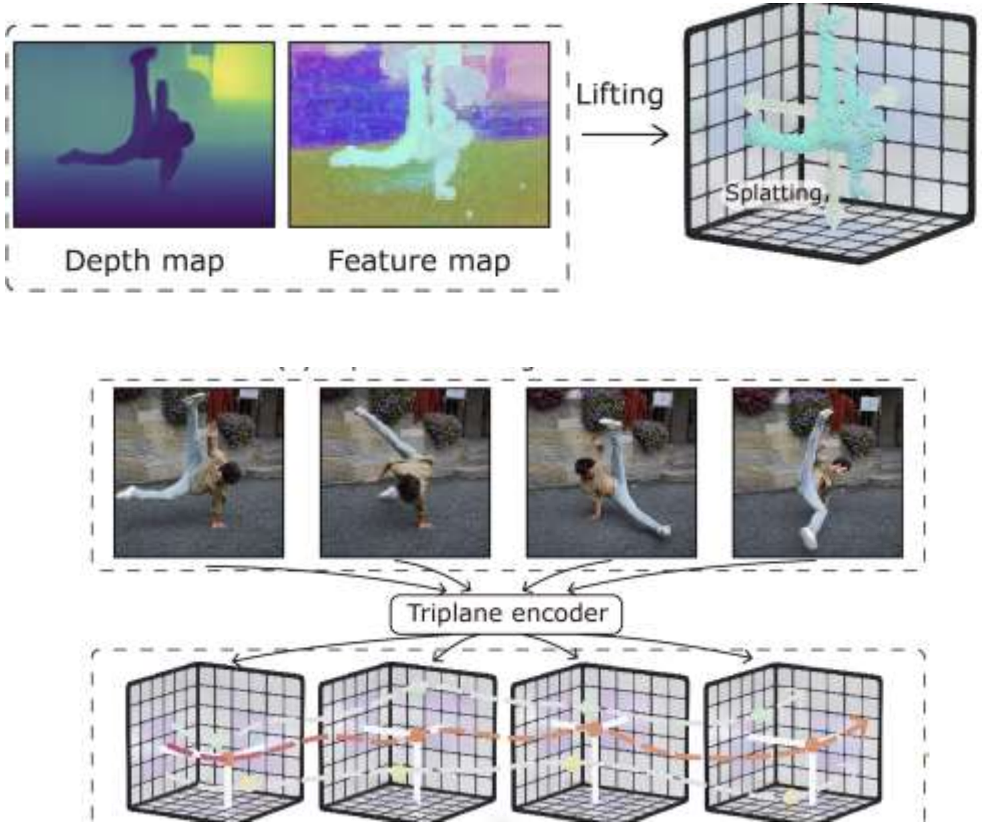
 Step-by-Step Pipeline:

1. **Start with monocular video** – no depth sensors needed.
2. **Estimate depth** for each frame using a model like ZoeDepth.
3. **Lift 2D pixels into 3D** → each pixel becomes a 3D point.
4. **Encode scene with Triplanes** (XY, XZ, YZ) to preserve spatial information.
5. **Track pixels over time** using a Transformer-based trajectory model.
6. **Enforce rigidity** using ARAP constraints to maintain structure, even under occlusion.



Triplane Encoding of Input Video Frames

- **Process:**
  - **Obtain monocular depth maps and multi-scale feature maps for each frame.**
  - **Unproject 2D pixels into 3D point clouds.**
  - **Splat 3D points onto three orthogonal planes (XY, XZ, YZ) to create triplane feature maps.**

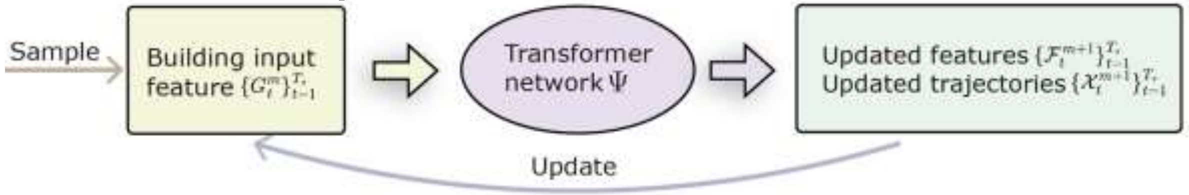


Iterative Trajectory Prediction

- **Input features:**

$$G^m_t = [\gamma(X^m_t), F^m_t, C^m_t, \gamma(X^m_t - X_1)] \in R^D$$

  - **Positional encoding : Captures the current 3D location.**
  - **Feature vector : Extracted from triplane representation.**
  - **Correlation features : Compare local triplane features around the point.**
  - **Offset from starting position: Encodes motion relative to the initial position.**



- **Transformer Updates:**

$$G^m \in R^{N \times T_s \times D} = \{G^m_{i,t} \mid i = 1, ... ,N; t = 1, ..., T_s\}$$

  - **Transformer predicts new positions and features**

$$X^{m+1}, F^{m+1} = \Psi(G^m)$$

  - **Repeat for M iterations to refine trajectories.**

ARAP – As Rigid As Possible Constraint

1. What is ARAP?

- ARAP enforces that points belonging to the same rigid part maintain constant distances over time.
- Enhances spatial consistency during occlusions and complex motions.

2. Why Use ARAP?

- Handles occlusions by leveraging neighboring visible points in the same rigid group.
- Prevents unrealistic deformations in tracked trajectories.

3. Rigidity Embedding

- At each iteration m, compute a rigidity embedding E^m_i for each trajectory i.
- Aggregates input features across all the frames.
- Calculate affinity S^m_{ij} between any two trajectories i and j, Uses cosine similarity to measure motion similarity:

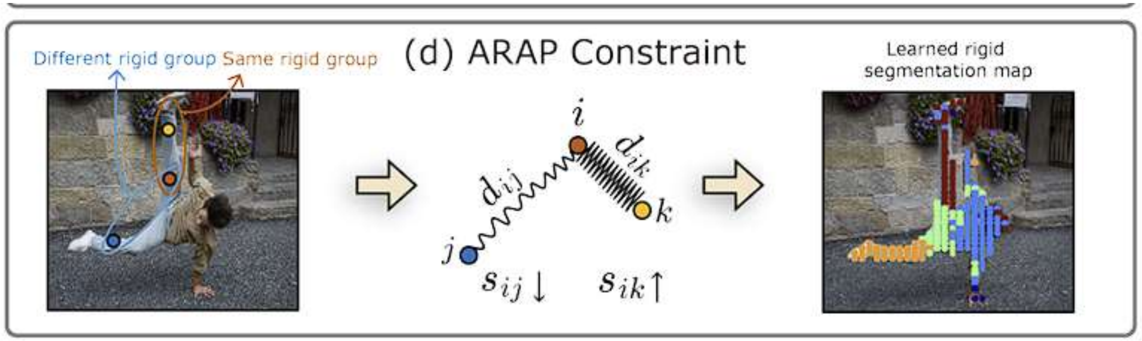
$$S^m_{ij} = \mathrm{sim}(E^m_i, E^m_j)$$

ARAP Loss

- **Loss Function:**
  - Encourages distances between points with high rigidity affinity to remain constant over time:

$$L_{arap} = \sum^{M}_{m=1} \sum^{Ts}_{t=1} \sum_{\Omega ij} w^m s^m_{ij} \|d(X^m_{i,t}, X^m_{j,t}) - d(X_{i,1},X_{j,1})\|_1$$

  - Rigidity embeddings are learned self-supervisedly.
  - Spectral clustering on affinity scores produces meaningful segmentation of rigid parts.



Loss Calculation

- **Trajectory Loss:**

$$\mathbf{L_{traj} = \sum^{M}_{m=1} \sum^{N}_{i=1} \sum^{Ts}_{t=1} w^{m} \| X^{m}_{i,t} - X\hat{}^{m}_{i,t} \|_{1}}$$

- **Visibility Loss:**

$$\mathbf{L_{vis} = \sum^{N}_{i=1} \sum^{Ts}_{t=1} CE(v_{i,t}, v\hat{}_{I,t})}$$

- **Total Loss:**

$$\mathbf{L_{total} = L_{traj} + \alpha L_{vis} + \beta L_{arap}}$$

where α and β are weighting coefficients. In practice, they are set as 10 and 0.1, respectively

Training & Implementation details

- **Training:**
  - **Trained on TAP-Vid-Kubric dataset with 11,000 RGB-D sequences.**
  - **Used ground truth depth maps and camera intrinsics during training.**
  - **Inference uses ZoeDepth for metric depth estimation.**
- **Parameters:**
  - **Iteration steps M=6.**
  - **Sliding window length Ts=8.**
  - **Transformer consists of 6 blocks with spatial and temporal attention layers.**

Experiments Overview

- Conducted evaluations on three long-range tracking benchmarks.
- TAP-Vid : Real and synthetic videos.
- BADJA : Videos of moving animals with key points.
- Point Odyssey : Synthetic dataset with diverse animated characters.

- Compared against baseline methods.
- TAP-Net, PIPs, Omni Motion, TAPIR , CoTracker .
- Evaluated using metrics like AJ (Average Jaccard), <δx_avg (average position accuracy), OA (Occlusion Accuracy), MTE (Median Trajectory Error), and Survival Rate.

2D Pixel Tracking

- Input: RGB video without known depth or camera intrinsics.
- Depth Estimation: Uses ZoeDepth to estimate depth maps.
- Evaluation: Only evaluates the 2D projection of 3D trajectories onto the image plane.
- Datasets: TAP-Vid benchmark, which includes Kinetics , DAVIS , and RGB-Stacking ,BADJA and Point Odyssey.
- Average Position Accuracy (<δx_avg), Average Jaccard (AJ), Occlusion Accuracy (OA).

Results on TAP-Vid Benchmark

- Reported results on Kinetics , DAVIS , and RGB-Stacking datasets.
- Spatial Tracker consistently outperforms baselines across all metrics.
- Follows the "queried first" protocol from CoTracker, use the first frame as the query frame and predict the 2D locations of query pixels in all subsequent frames.

Methods	Kinetics [9]			DAVIS [50]			RGB-Stacking [33]			Average		
	AJ ↑	$< \delta_{avg}$ ↑	OA ↑	AJ ↑	$< \delta_{avg}$ ↑	OA ↑	AJ ↑	$< \delta_{avg}$ ↑	OA ↑	AJ ↑	$< \delta_{avg}$ ↑	OA ↑
TAP-Net [11]	38.5	54.4	80.6	33.0	48.6	78.8	54.6	68.3	87.7	42.0	57.1	82.4
PIPs [17]	31.7	53.7	72.9	42.2	64.8	77.7	15.7	28.4	77.1	29.9	50.0	75.9
OmniMotion [71]		-	-	46.4	62.7	85.3	**69.5**	**82.5**	**90.3**	-	-	-
TAPIR [12]	49.6	64.2	85.0	56.2	70.0	86.5	54.2	69.8	84.4	53.3	68.0	85.3
CoTracker [29]	48.7	64.3	86.5	60.6	75.4	89.3	63.1	77.0	87.8	57.4	72.2	87.8
Ours	**50.1**	**65.9**	**86.9**	**61.1**	**76.3**	**89.5**	63.5	77.6	88.2	**58.2**	**73.3**	**88.2**

Table 1. **2D Tracking Results on the TAP-Vid Benchmark.** We report the average jaccard (AJ), average position accuracy ($< \delta^{x}_{avg}$), and occlusion accuracy (OA) on Kinetics [9], DAVIS [50] and RGB-Stacking [33] datasets.

Results on BADJA dataset

- BADJA contains videos of moving animals with annotated keypoints.
- Evaluated using metrics like segA(segment based accuracy), **δ3px(** Percentage of keypoints within 3 pixels of ground truth.)
- Predicted keypoint positions are deemed accurate if their distance from the ground truth is less than 0.2A^1/2 , where A is the summation of the area of the segmentation mask.

Methods	**segA** ↓	δ^{3px} ↑
TAP-Net [11]	54.4	6.3
PIPs [17]	61.9	13.5
TAPIR [12]	66.9	15.2
OmniMotion [71]	57.2	13.2
CoTracker [29]	63.6	**18.0**
Ours	**69.2**	17.1

Results on the Point Odyssey Dataset

- Point Odyssey features diverse animated characters in complex environments.
- Evaluated using metrics like MTE (Median trajectory error) ,<δx_avg (Average position accuracy), Survival (Average number of frames until tracking failure).
- Tracking failure is identified when the L2error exceeds 50 pixels at a resolution of 256 ×256.

Methods	MTE↓	$<\delta^{x}_{avg}$ ↑	Survival↑
TAP-Net [11]	37.8	29.2	52.8
PIPs [81]	41.0	30.4	67.0
CoTracker [29]	30.5	56.2	76.1
Ours *w/* ZoeDepth [2]	28.3	58.4	**78.6**
Ours *w/* GT depth	**26.6**	**64.1**	78.0

Qualitative Comparisons

- Handling self-occlusions better.
- Tracking small, fast-moving objects accurately.

3D Tracking Evaluation

- Evaluated on Point Odyssey using known depth and intrinsics.
- Chained RAFT-3D : Chains pairwise scene flow predictions.
- Lifted CoTracker : Lifts 2D trajectories into 3D using depth maps.
- Create 231 testing sequences from the test set, each consisting of 24 frames with reduced frame rate (one-fifth of original).

Methods	$ATE_{3D} \downarrow$	$\delta_{0.1} \uparrow$	$\delta_{0.2} \uparrow$
Chained RAFT3D [63]	0.70	0.12	0.25
Lifted CoTracker [29]	0.77	0.51	0.64
Ours	**0.22**	**0.59**	**0.76**

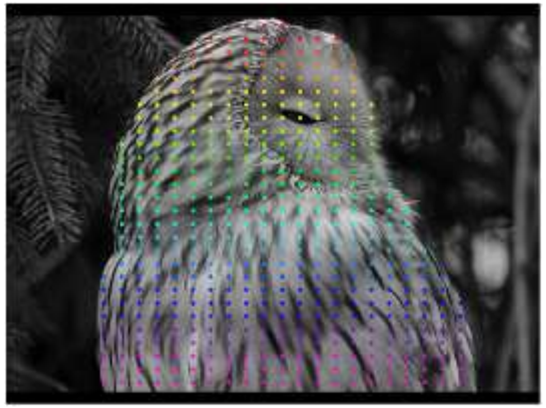
Ablation Study

- Removing ARAP loss reduces performance significantly.
- Confirms the effectiveness of ARAP constraints.
- Tested with ZoeDepth, MiDaS, and DPT.
- ZoeDepth performs best due to its metric depth and temporal consistency.

Methods	AJ↑	$<\delta^x_{avg}$↑	OA↑
Ours *w/o* ARAP	55.1	71.6	87.4
Ours *w/* DPT [53]	51.4	70.7	83.3
Ours *w/* MiDaS [5]	56.3	73.9	86.6
Ours *w/* ZoeDepth [2] (default)	**61.1**	**76.3**	**89.5**

Rigid Part Segmentation

- Used spectral clustering on rigidity embeddings to identify rigid groups.
- Each color represents a distinct rigid group (e.g., car door, wheels).



Reference Image **Tracking Results** **Estimated Rigid Parts**

Conclusion

- Proposed a novel framework for dense and long-range motion estimation in videos.
- Tracking in 3D space improves accuracy and handles occlusions better than 2D tracking.
- ARAP constraints enforce spatial consistency and improve tracking during occlusions.
- Triplane representations enable efficient 3D tracking.
- Achieved state-of-the-art performance on multiple benchmarks.

Future Work

- Enhance monocular depth estimation to further improve tracking performance.
- Investigate closer interplay between motion estimation and depth reconstruction.
- Extend the method to handle multi-view scenarios for richer 3D context.

Thank you