Project Report: Object Tracking using CoTracker v3

Introduction

Object tracking is a fundamental task in computer vision where an object is identified and its position is continuously estimated across video frames. Real-time, long-term tracking is critical for applications like gesture recognition, activity monitoring, AR/VR, and robotics. Traditional trackers often struggle with occlusion, abrupt motion, or complex backgrounds. The CoTracker-v3 model (by Meta AI) overcomes these challenges by using a transformer-based, space-time attention mechanism to robustly track multiple points or regions over long video sequences.

Problem Statement

Given a collection of self-recorded videos captured using mobile or webcam, track a user-specified object (e.g. hand, face, pet, phone) throughout the video using the CoTracker-v3 model. The goal is to extract accurate, continuous trajectories of the object, even in cases of occlusion, motion blur, or background clutter.

Methodology

- **Upload Video 📽️ & Load Frames 🖼️**
- Open video using OpenCV
- Extract frames → NumPy array (T, H, W, 3)
- **Convert Frames → Tensor ⚙️**
- Convert to PyTorch tensor
- Shape: (T, H, W, 3) → (1, T, 3, H, W)
- Move to GPU (cuda)
- **Load CoTracker (Offline) 🤖**
- Load pretrained CoTracker v3 (offline mode)
- Processes full video at once
- **Define Queries (Points) 📍**

- Format: [frame, x, y]
- Manually select keypoints on any frame
- **Run Model & Predictions** 📈
- Predict tracked positions (T, N, 2)
- Predict visibility (T, N)
- **Generate Tracked Video Output** 💾
- Draw trajectories with OpenCV
- Apply smoothing filter
- Save final output .mp4

Evaluation

Smoothness: Measures how **stable** and **consistent** the trajectories are

Continuity: Indicates how **long points remain visible** across frames

Trajectory Length: Represents the **total motion** covered by tracked points

https://drive.google.com/drive/folders/1_N8-Ew6-2zcAEMyYcTEvFwcLnyH4jPLA?usp=drive_link

Challenges

Non-rigid Object Tracking

- CoTracker struggles with objects that **change shape** significantly (like a waving hand or deforming cloth).

Occlusion Handling

- Tracking becomes less accurate when the object is **temporarily hidden** behind other objects.

Lack of Ground Truth Data

- Using **self-recorded videos** means no labeled data for metrics like PCK, MSE, or EPE, making quantitative evaluation harder.

Computational Requirements

- Processing long videos with high resolution requires **significant GPU memory and time**.

Initialization Dependency

- The offline model requires **manual selection of query points**, limiting full automation.

Complex Motion Patterns

- Fast or unpredictable movements can reduce tracking accuracy.

Future Scope

- Add **ground truth dataset** for metrics like **PCK(Percentage of correct keypoints), MSE(Mean squared Error), EPE(End point error)**.
- Track **more complex motions**: multi-object, occlusions, fast movements.
- Improve robustness for **blurred frames** and **low visibility** points.
- Explore **re-detection or interpolation** to handle point loss.
- Detects objects in each frame using **object detection models** (e.g., YOLO, Faster R-CNN)
- Maintains consistent IDs across frames using **SORT, DeepSORT, or Kalman Filter**
- Can handle moving, appearing, or disappearing objects
- Challenges: **occlusion, motion blur, ID switching**, and **real-time performance**
- Future scope: **combine with multi-point tracking for higher accuracy**
- **In Progress :Tracks objects automatically without manually defining points**
- https://drive.google.com/file/d/15QlhpN_U8F5Bsqx9mfVgMhAExXmarr2A/view?usp=sharing

Conclusion

- CoTracker demonstrates strong ability in **multi-point tracking** with **smooth and stable trajectories**, even without ground truth.
- However, **continuity challenges** arise during occlusion, blur, or fast motion, leading to keypoint loss.
- **Memory usage** remains a concern, requiring high-performance GPUs for long or high-resolution videos.

- Despite these limitations, the model provides **meaningful and practical results** for real-world tracking tasks.
- Future improvements could focus on **better handling of occlusion, reducing memory requirements, and integrating ground truth datasets** for robust evaluation.