Tracking

Computer Vision – Lecture 13

Further Reading

- Lecture from A Vedaldi and A Zisserman
- Open for suggestions…

Visual Tracking

Visual tracking involves the identification of some characteristic of the scene in successive images.

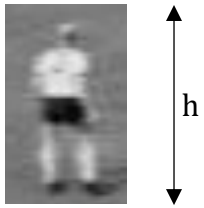
2D tracking

- follow and perhaps control the image position of some entity as it moves from frame to frame over time.

3D (or pose) tracking

- use image measurements (possibly involving 2D tracking) to update the 6 degrees of freedom (3 translation + 3 rotation) which define 3D pose.

Recap: Template Tracking



Template T

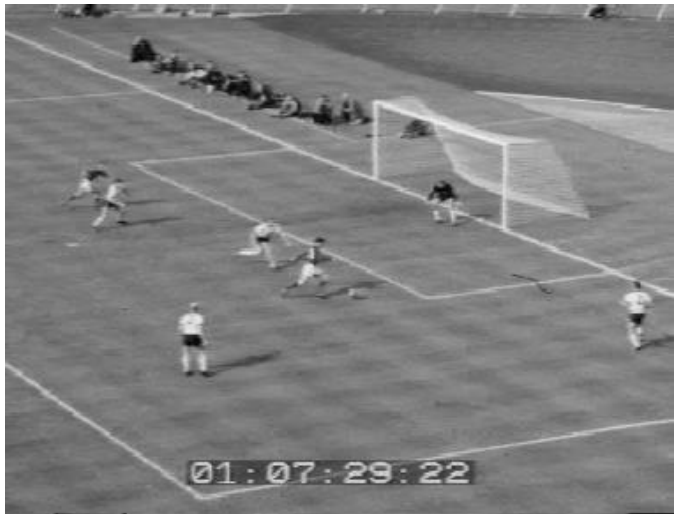
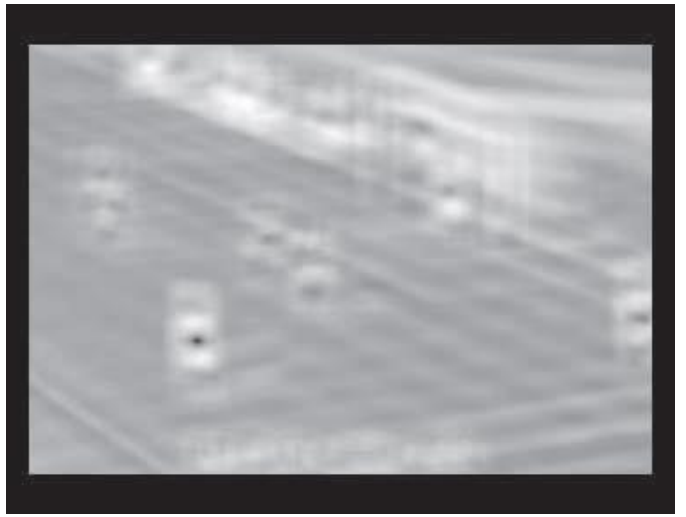


Image I

Sum of squared differences between image I and template T at every location (u, v)



Energy E

$$E(u,v) = \sum_{(x,y)\in\left[-\frac{w}{2},\frac{w}{2}\right]\times\left[-\frac{h}{2},\frac{h}{2}\right]} \left(I(u+x, v+y) - T\left(x+\frac{w}{2}, y+\frac{h}{2}\right)\right)^2$$

- Tracking by Detection (each frame processed individually)
- Very slow: (#pixels_image * #pixels_template) comparisons

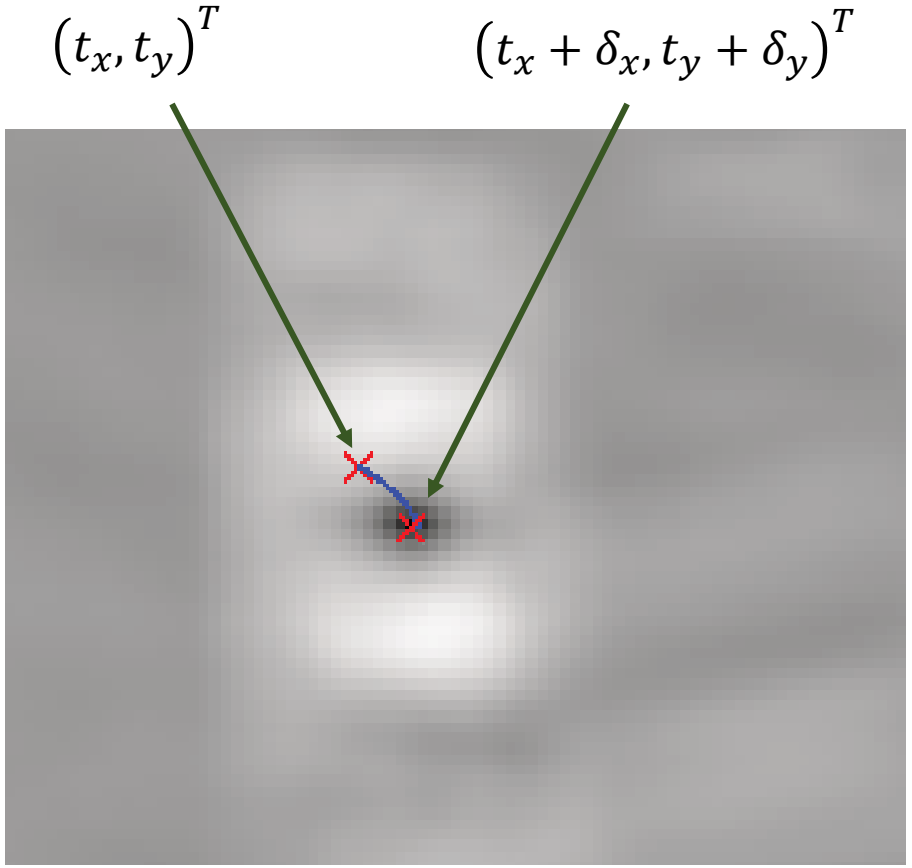
Lucas-Kanade Template Tracking

$$E(u,v) = \sum_{x,y} \big(I(u+x, v+y) - T(x,y)\big)^2$$

Improving the efficiency:

- Formulate search as an optimisation problem using brightness constancy as our objective function
- E is a non-convex function over the image I
- Suppose the starting point is $(t_x, t_y)^T$ e.g., detection in prev. frame
- LK searches for an update $(\delta_x, \delta_y)^T$ of the starting point

Lucas-Kanade Template Tracking



Reformulate the energy relative to the update (δ_x, δ_y):

$$E(\delta_x, \delta_y) = \sum_{x,y \in T} \left(I(t_x + x + \delta_x, t_y + y + \delta_y) - T(x,y) \right)^2$$

1st order Taylor expansion ($\nabla I = (I_x, I_y)^T \in \mathbb{R}^2$ is the image gradient):

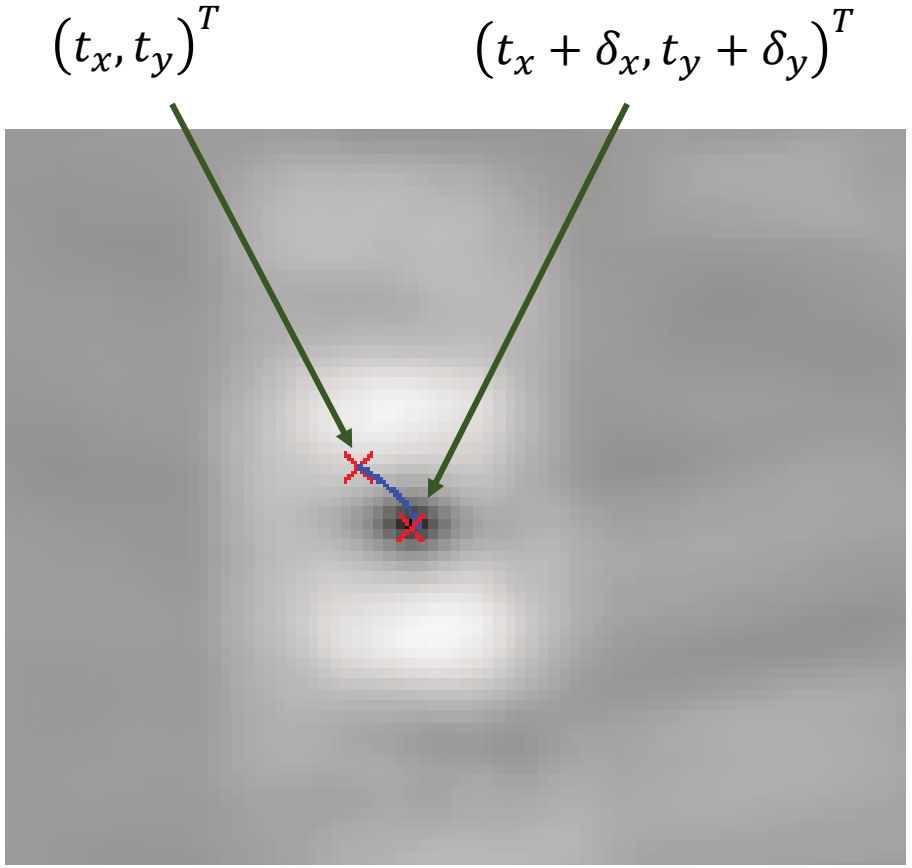
$$E(\delta_x, \delta_y) \approx \sum_{x,y \in T} \left(I(t_x + x, t_y + y) + \nabla I^T \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} - T(x,y) \right)^2$$

Local minimum where partial derivatives of E w.r.t to δ are 0:

$$\frac{\partial E}{\partial \delta_x} = 2I_x \sum_{x,y \in T} \left(I(t_x + x, t_y + y) + \nabla I^T \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} - T(x,y) \right) = 0$$

$$\frac{\partial E}{\partial \delta_y} = 2I_y \sum_{x,y \in T} \left(I(t_x + x, t_y + y) + \nabla I^T \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} - T(x,y) \right) = 0$$

Lucas-Kanade Template Tracking



Rewrite in matrix form:

$$\underbrace{\left[\sum_{x,y\in T} \nabla I \nabla I^T\right]}_{\text{2x2 matrix}} \underbrace{\begin{bmatrix}\delta_x \\ \delta_y\end{bmatrix}}_{\text{2x1 vector}} = \underbrace{-\sum_{x,y\in T} \nabla I \left(I(t_x + x, t_y + y) - T(x, y)\right)}_{\text{2x1 vector}}$$

- We can solve for (δ_x, δ_y) in closed form
- Update the current estimate (t_x, t_y)
- Keep iterating until convergence, i.e. (δ_x, δ_y) is small

- This idea is **very** general
- The example was using translation only $(t_x + \delta_x, t_y + \delta_y)$
- Let's generalise it to any transform W

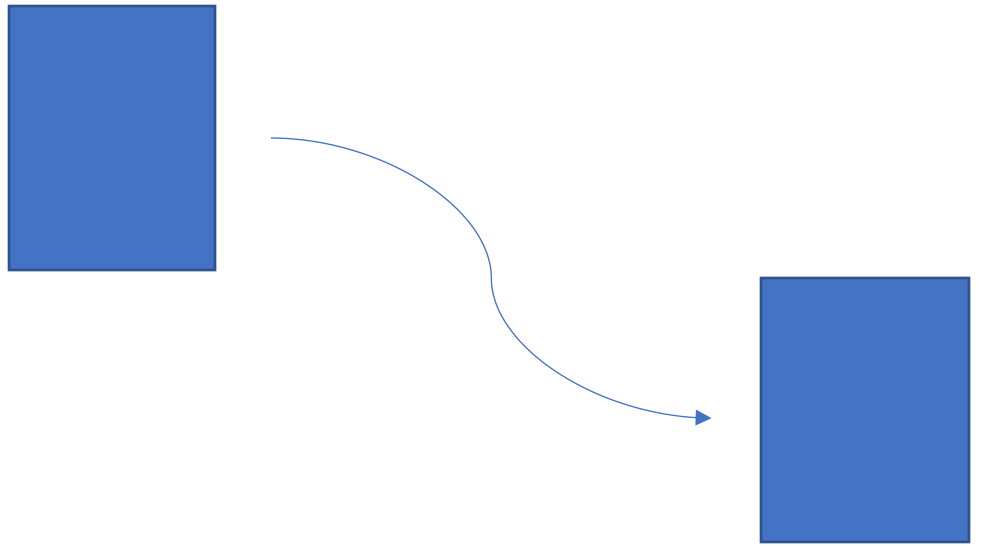
Generalised LK tracking

- Transform W
- Parameters $\boldsymbol{p}$
- Updates $\Delta\boldsymbol{p}$
- Pixel $\boldsymbol{x} = (x, y)^T$

- Previous example: translation

$$\boldsymbol{p} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \qquad \Delta\boldsymbol{p} = \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix}$$

$$W(\boldsymbol{x}, \boldsymbol{p}) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



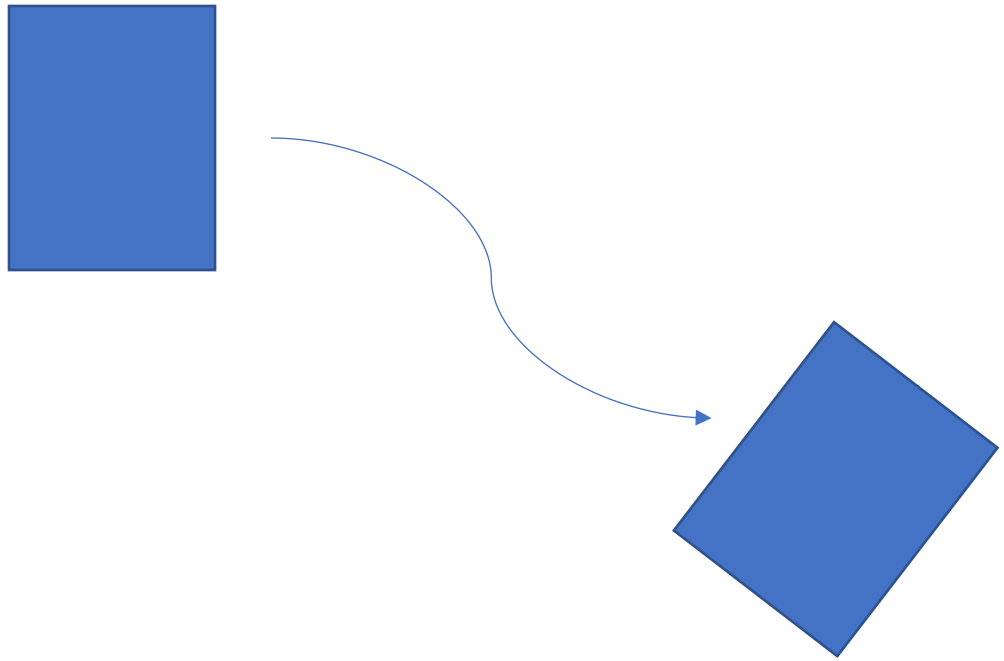
Generalised LK tracking

- Transform W
- Parameters $\boldsymbol{p}$
- Updates $\Delta\boldsymbol{p}$
- Pixel $\boldsymbol{x} = (x, y)^T$

- Example: translation + rotation

$$\boldsymbol{p} = \begin{bmatrix} t_x \\ t_y \\ \theta \end{bmatrix}$$

$$W(\boldsymbol{x}, \boldsymbol{p}) = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



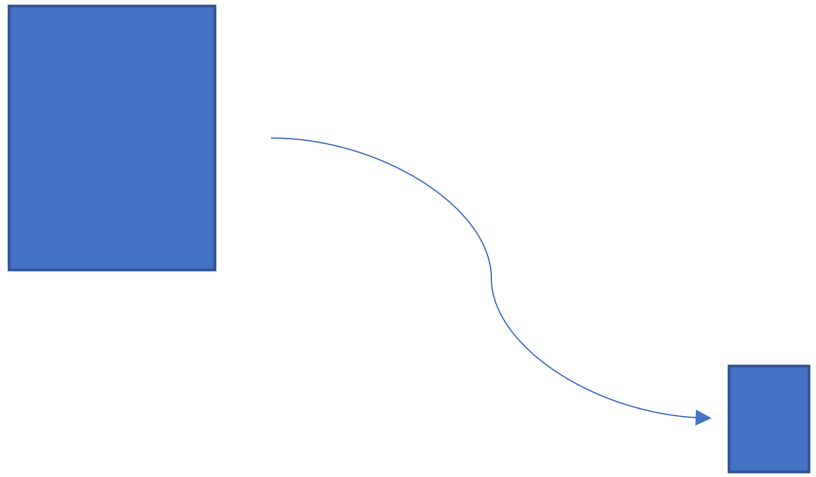
Generalised LK tracking

- Transform W
- Parameters $\boldsymbol{p}$
- Updates $\Delta\boldsymbol{p}$
- Pixel $\boldsymbol{x} = (x, y)^T$

- Example: translation + scaling

$$\boldsymbol{p} = \begin{bmatrix} t_x \\ t_y \\ s \end{bmatrix}$$

$$W(\boldsymbol{x}, \boldsymbol{p}) = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



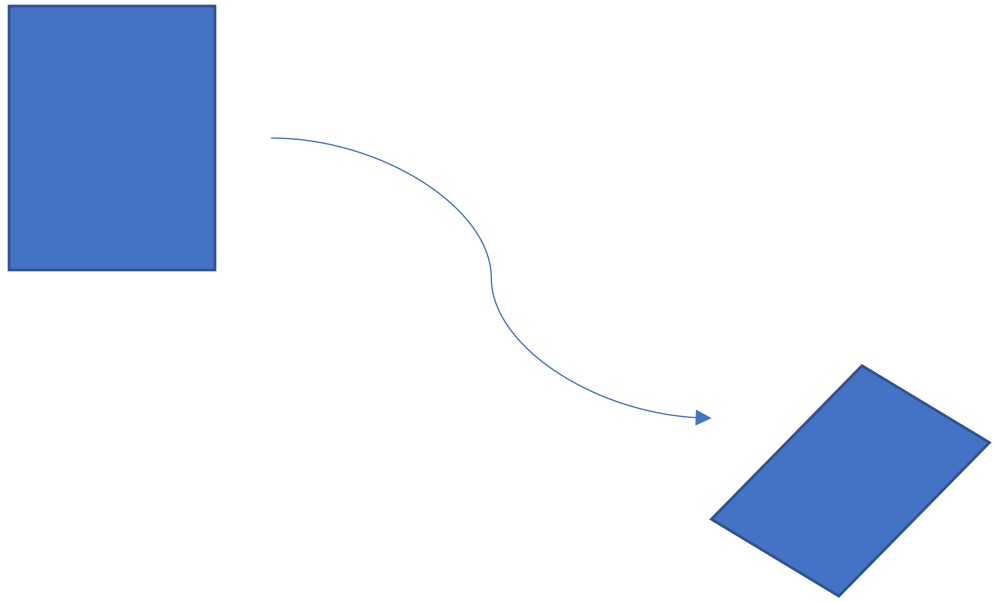
Generalised LK tracking

- Transform W
- Parameters $\boldsymbol{p}$
- Updates $\Delta\boldsymbol{p}$
- Pixel $\boldsymbol{x} = (x, y)^T$

- Example: affine

$$\boldsymbol{p} = \begin{bmatrix} p_{11} \\ \vdots \\ p_{23} \end{bmatrix}$$

$$W(\boldsymbol{x}, \boldsymbol{p}) = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Generalised LK tracking

Energy in general form:

$$E(\boldsymbol{p}) = \sum_{\boldsymbol{x} \in T} \left(I(W(\boldsymbol{x}, \boldsymbol{p})) - T(\boldsymbol{x}) \right)^2$$

Warp each point $\boldsymbol{x}$ in the template to its image location using the warp W and its parameters $\boldsymbol{p}$.

Assume a current estimate $\boldsymbol{p}$ is known, solve for an update $\Delta\boldsymbol{p}$:

$$E(\boldsymbol{\Delta p}) = \sum_{\boldsymbol{x} \in T} \left(I(W(\boldsymbol{x}, \boldsymbol{p} + \boldsymbol{\Delta} p)) - T(\boldsymbol{x}) \right)^2$$

Same as before, update $\boldsymbol{p}$ with $\boldsymbol{p} + \Delta\boldsymbol{p}$ and iterate.

To solve for $\Delta\boldsymbol{p}$, approximate with 1st order Taylor expansion:

$$E(\boldsymbol{\Delta p}) \approx \sum_{\boldsymbol{x} \in T} \left(I(W(\boldsymbol{x}, \boldsymbol{p})) + \nabla I^T \frac{\partial W}{\partial \boldsymbol{p}} \Delta\boldsymbol{p} - T(\boldsymbol{x}) \right)^2$$

$\frac{\partial W}{\partial \boldsymbol{p}}$ is the **Jacobian** of the warp.

Jacobian

The Jacobian is the matrix of all 1st order derivatives of a function.

In our case:

$$\frac{\partial W}{\partial \boldsymbol{p}} = \begin{bmatrix} \frac{\partial W_x}{\partial \boldsymbol{p_1}} & \cdots & \frac{\partial W_x}{\partial \boldsymbol{p_n}} \\ \frac{\partial W_y}{\partial \boldsymbol{p_1}} & \cdots & \frac{\partial W_y}{\partial \boldsymbol{p_n}} \end{bmatrix}$$

Example: translation + scaling

$$\boldsymbol{p} = \begin{bmatrix} t_x \\ t_y \\ s \end{bmatrix}$$

$$W(\boldsymbol{x}, \boldsymbol{p}) = \begin{bmatrix} s & 0 & t_x \\ 0 & s & t_y \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\frac{\partial W}{\partial \boldsymbol{p}} = \begin{bmatrix} 1 & 0 & x \\ 0 & 1 & y \end{bmatrix}$$

Generalised LK tracking

$$E(\boldsymbol{\Delta p}) \approx \sum_{\boldsymbol{x} \in T} \left(I(W(\boldsymbol{x}, \boldsymbol{p})) + \nabla I^T \frac{\partial W}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right)^2$$

Partial derivative with respect to $\Delta \boldsymbol{p}$:

$$2 \sum_{\boldsymbol{x} \in T} \left(\nabla I^T \frac{\partial W}{\partial \boldsymbol{p}} \right)^T \left(I(W(\boldsymbol{x}, \boldsymbol{p})) + \nabla I^T \frac{\partial W}{\partial \boldsymbol{p}} \Delta \boldsymbol{p} - T(\boldsymbol{x}) \right)$$

Setting this to equal zero leads to:

$$\sum_{\boldsymbol{x} \in T} \left(\nabla I^T \frac{\partial W}{\partial \boldsymbol{p}} \right)^T \left(\nabla I^T \frac{\partial W}{\partial \boldsymbol{p}} \right) \Delta p = \sum_{\boldsymbol{x} \in T} \left(\nabla I^T \frac{\partial W}{\partial \boldsymbol{p}} \right)^T \left(T(\boldsymbol{x}) - I(W(\boldsymbol{x}, \boldsymbol{p})) \right)$$

Generalised LK tracking

$$\sum_{\boldsymbol{x}\in T}\left(\nabla I^T\frac{\partial W}{\partial \boldsymbol{p}}\right)^T\left(\nabla I^T\frac{\partial W}{\partial \boldsymbol{p}}\right)\Delta p=\sum_{\boldsymbol{x}\in T}\left(\nabla I^T\frac{\partial W}{\partial \boldsymbol{p}}\right)^T\left(T(\boldsymbol{x})\ -\ I(W(\boldsymbol{x},\boldsymbol{p}))\right)$$

Simplifying:

$$\boldsymbol{M}=\sum_{\boldsymbol{x}\in T}\left(\nabla I^T\frac{\partial W}{\partial \boldsymbol{p}}\right)^T\left(\nabla I^T\frac{\partial W}{\partial \boldsymbol{p}}\right)\qquad \boldsymbol{b}=\sum_{\boldsymbol{x}\in T}\left(\nabla I^T\frac{\partial W}{\partial \boldsymbol{p}}\right)^T\left(T(\boldsymbol{x})\ -\ I(W(\boldsymbol{x},\boldsymbol{p}))\right)$$

$$\boldsymbol{M\Delta p=b}$$

Compute update:

$$\boldsymbol{\Delta p=M^{-1}b}$$

Generalised LK tracking

- Simple update rule that iteratively refines the tracked position
- Any differentiable warp works
- Further optimization: pre-compute template gradients and warp “the other way”

- In-depth LK analysis:

    Lucas-Kanade 20 Years On: A Unifying Framework
    Simon Baker and Iain Matthews

LK Tracker Insights

- A general difficulty with trackers relying too heavily on the spatial relationships between pixels is that they are prone to break due to partial occlusion and orientation changes in the scene.

- Appearance changes can be compensated by updating the template from frame to frame

- But this can lead to “drift”:
  - The template will gradually pick up the background and eventually “stick”.
  - Can be avoided when background is simple, or with a segmentation mask.

Optical Flow – The Beginnings

- Assume a pixel at (x, y, t) with intensity $I(x, y, t)$ has moved by $\Delta x, \Delta y$ in space during a timestep Δt.
- Assume, the pixel did not change intensity, so:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

- If time and thus movement is small: Taylor expansion is a good approximation:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) \approx I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta \mathrm{y} + \frac{\partial I}{\partial t}\Delta \mathrm{t}$$

- With above assumption:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta \mathrm{y} + \frac{\partial I}{\partial t}\Delta \mathrm{t} = 0 \quad \text{or} \quad \frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} = 0$$

- Using $\nabla I = \left[\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right]^T$ and $\mu = \left[\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}\right]^T$ yields the **motion constraint equation:**

$$\mathbf{\nabla I^T}\,\boldsymbol{\mu} = -\frac{\partial I}{\partial t}$$

Horn and Schunck “Determining Optical Flow” Artificial Intelligence 17 (1981)

Optical Flow – The Beginnings

- $\nabla \mathrm{I}^{\mathrm{T}}\, \boldsymbol{\mu} = -\frac{\partial I}{\partial t}$ means we are only estimating the flow in the direction of the image gradient

- Smoothness constraint helps to regularise (weight α):

$$\min_{\boldsymbol{\mu}} \iint_R \left[\left(\nabla \mathrm{I}^{\mathrm{T}}\, \boldsymbol{\mu} - \frac{\partial I}{\partial t} \right)^2 + \alpha \left(|\nabla \mu_x|^2 + \left| \nabla \mu_y \right|^2 \right) \right] dx dy$$

- Second term makes flow of close pixels similar

Optical Flow – The Beginnings

Raw estimate

Smoothed estimate

Optical Flow – The Beginnings

- Intensity based optical flow
- Problems with uniform-coloured regions
- Difficult to regularise
  - Can be combined with LK tracking
  - Smoothness constraints

- Difficult evaluation on very few scenes
  - Synthetic with ground truth
  - Qualitative on real scenes

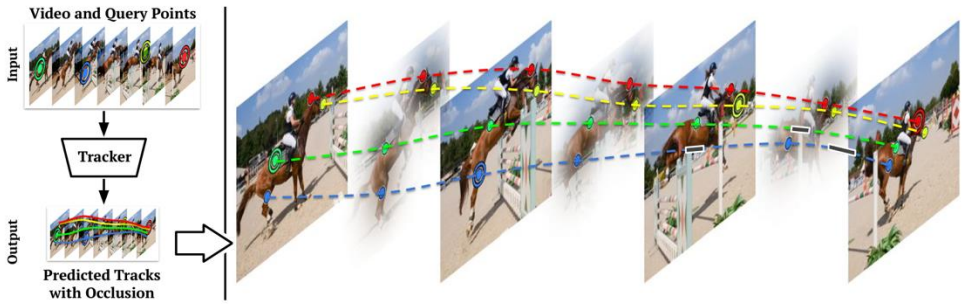
Motion Estimation

Point Tracking

Long-term tracking of individual points

PIPs

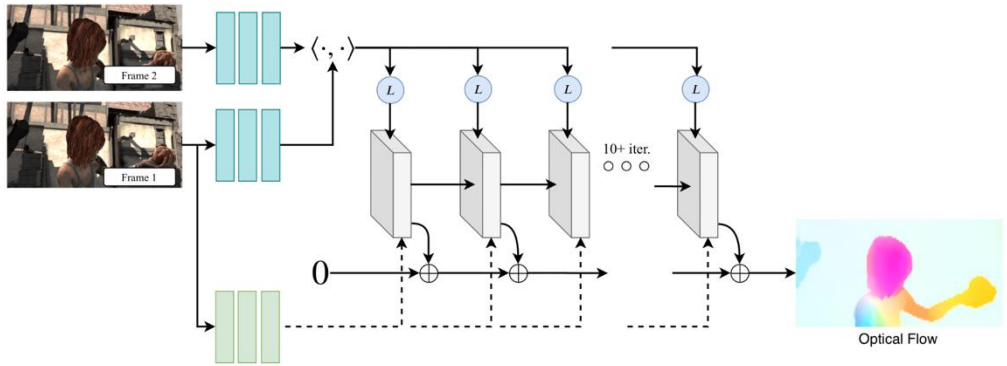
TAP-Net



Optical Flow

Dense correspondences between a pair of frames

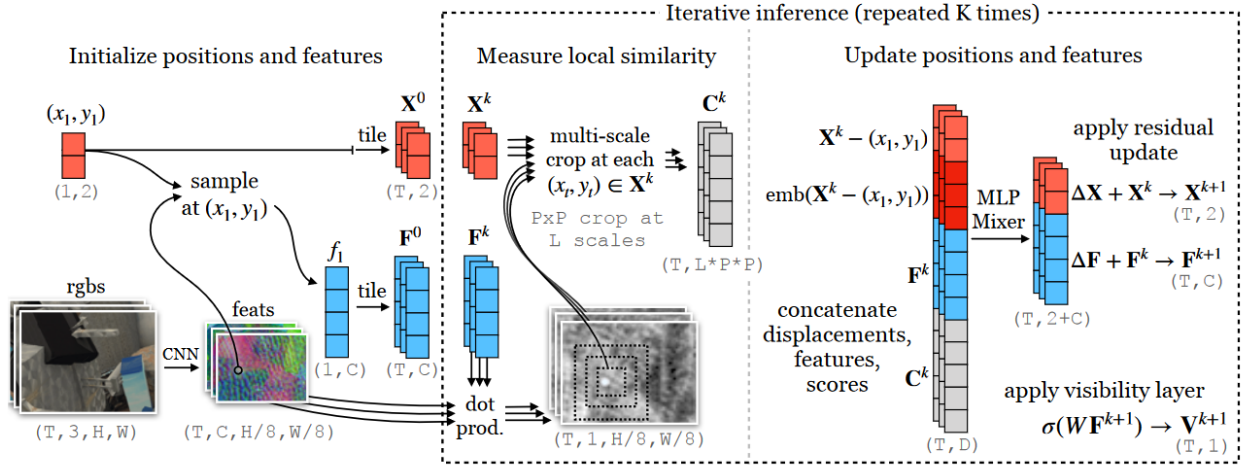
RAFT



Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022
TAP-Vid: A Benchmark for Tracking Any Point in a Video. Doersch et al., NeurIPS D&B 2022
RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. Teed et al. , ECCV 2020

Reading Computer Vision Papers

- Usually, papers have a main technical figure at the beginning of the “Methods” section.
- This figure is designed to give a holistic overview on how the method works.
- Example: Particle Video Revisited (Harley et al. ‘22).



Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited

Initialize positions and features

(x_1, y_1)

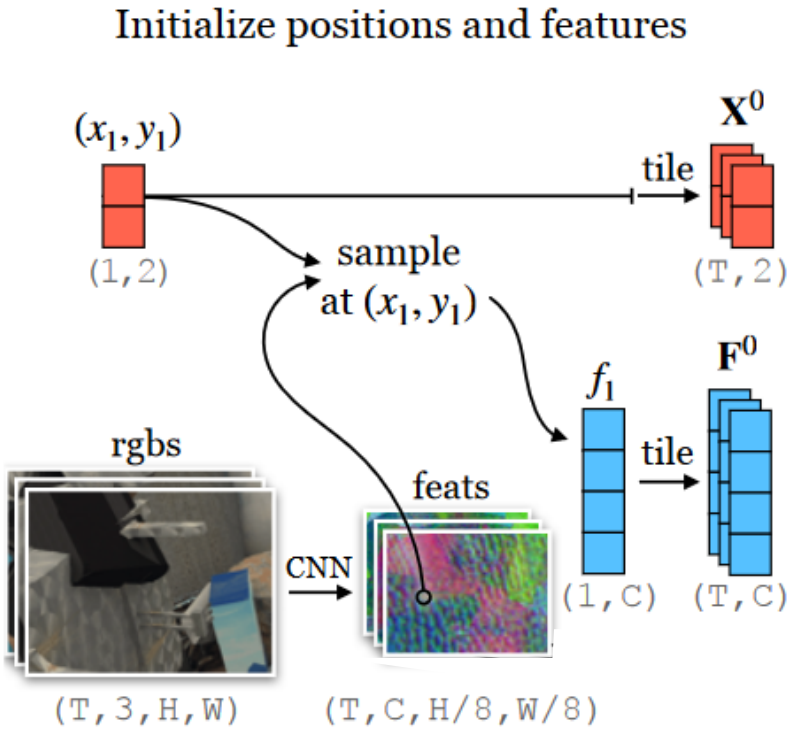
(1,2)

rgbs

(T,3,H,W)

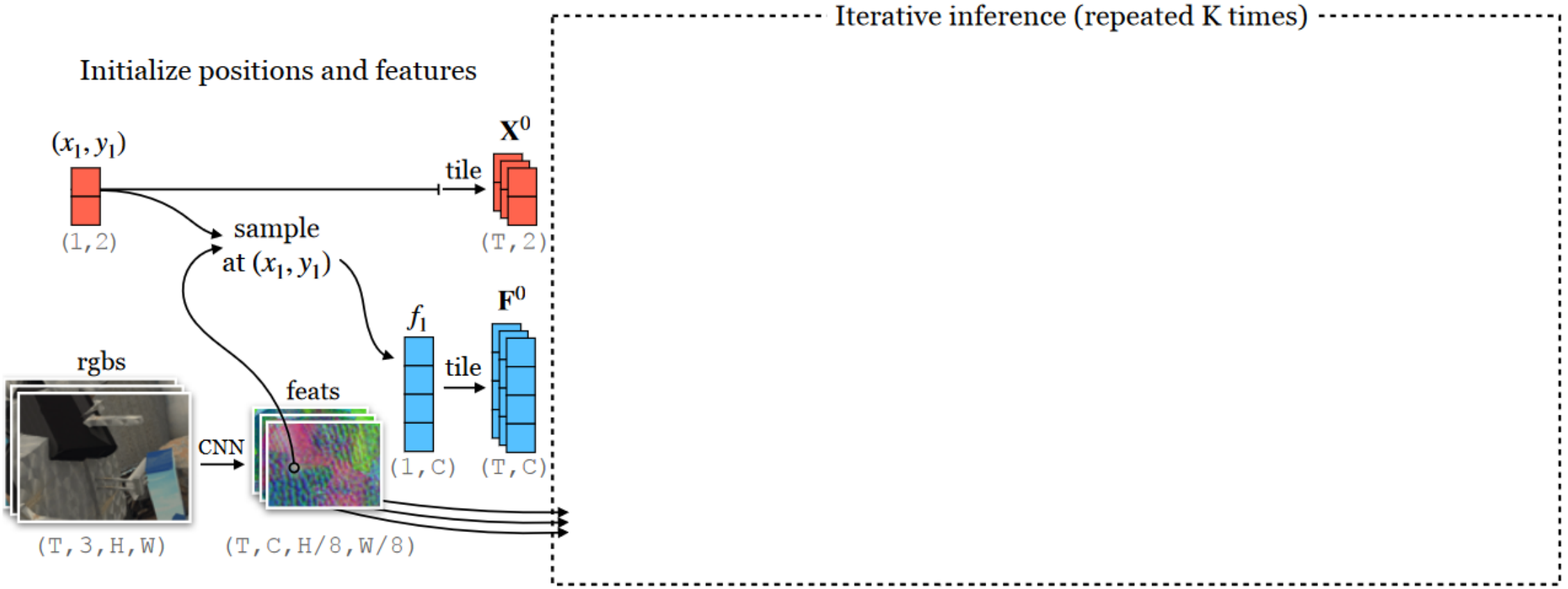
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



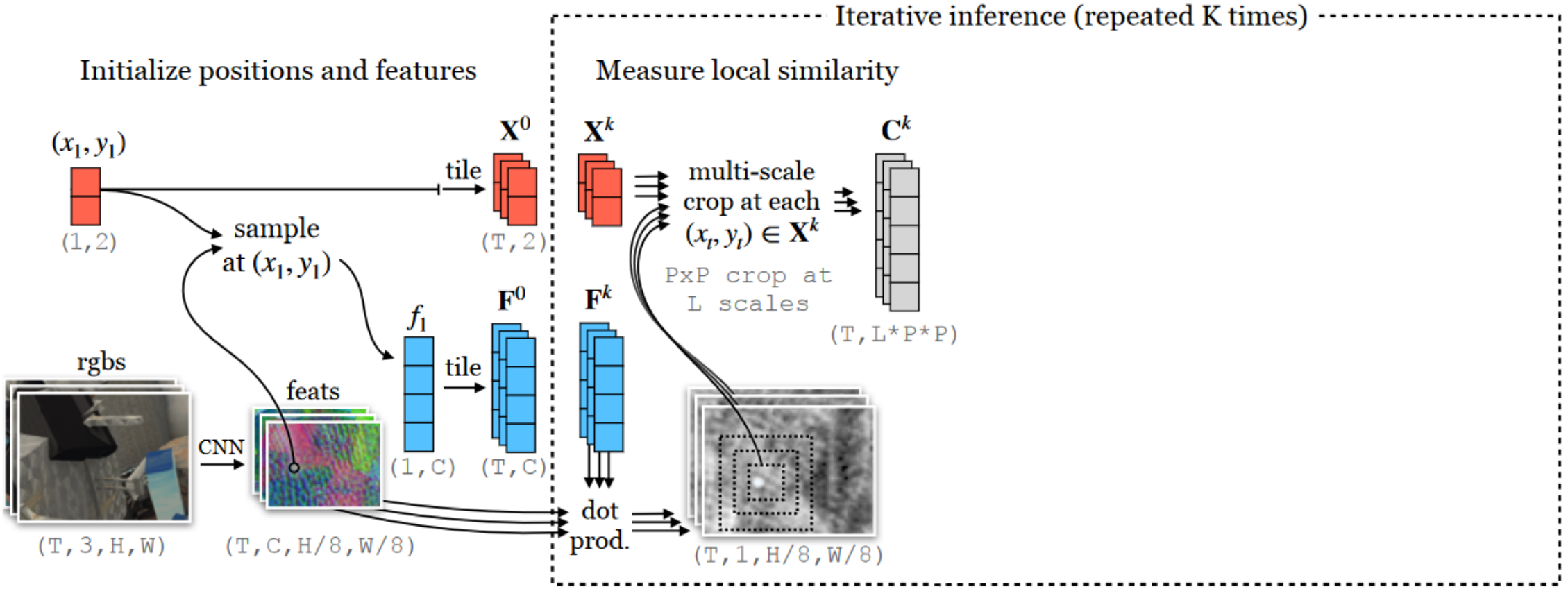
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



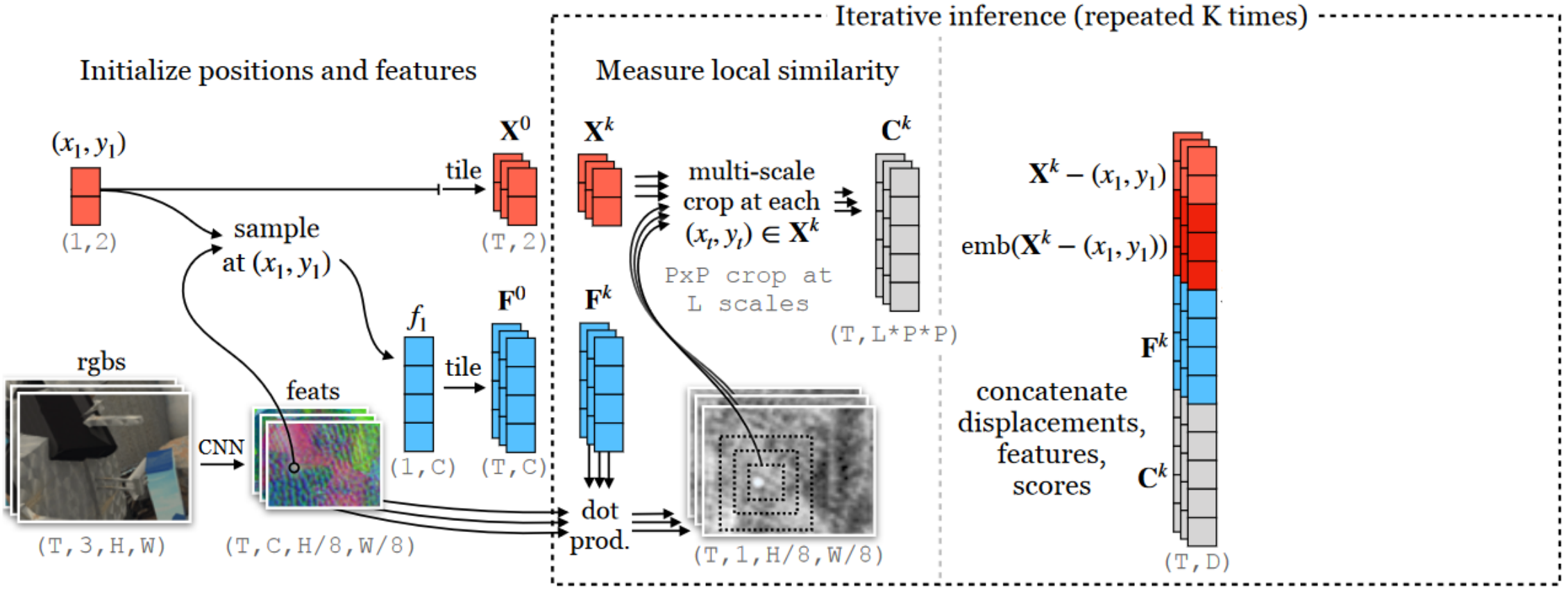
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



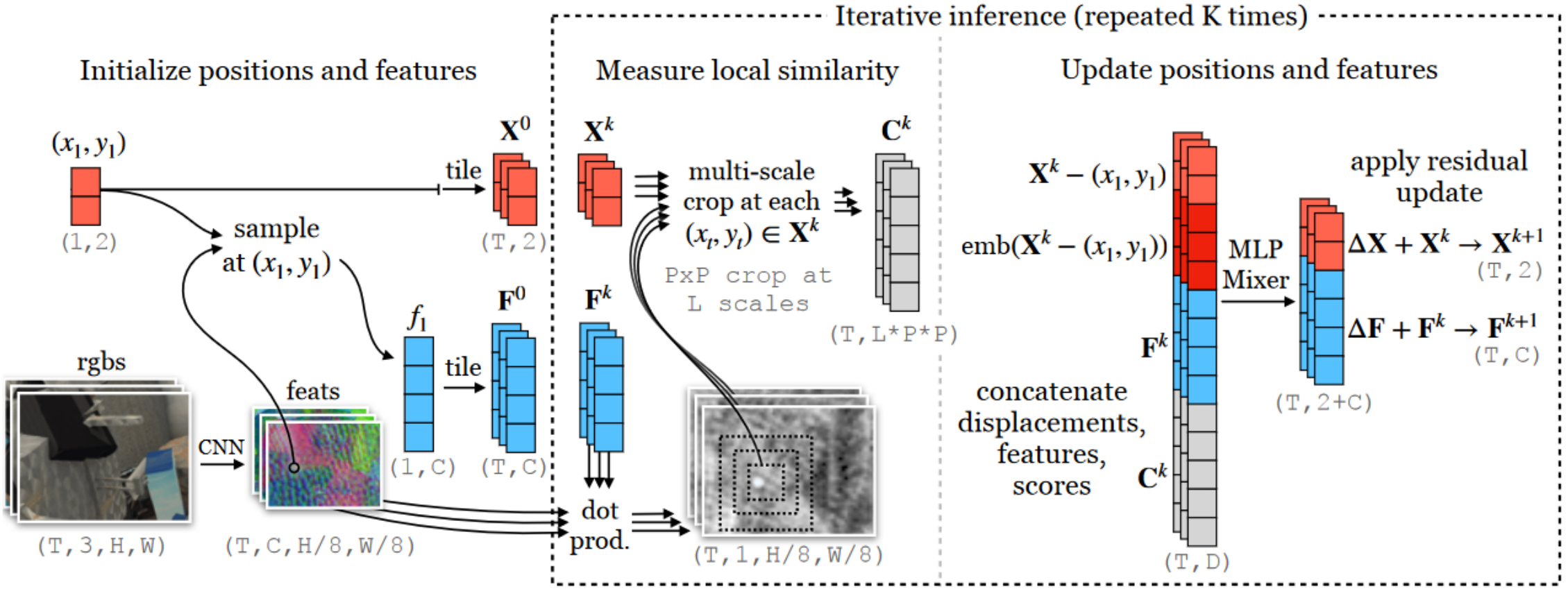
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



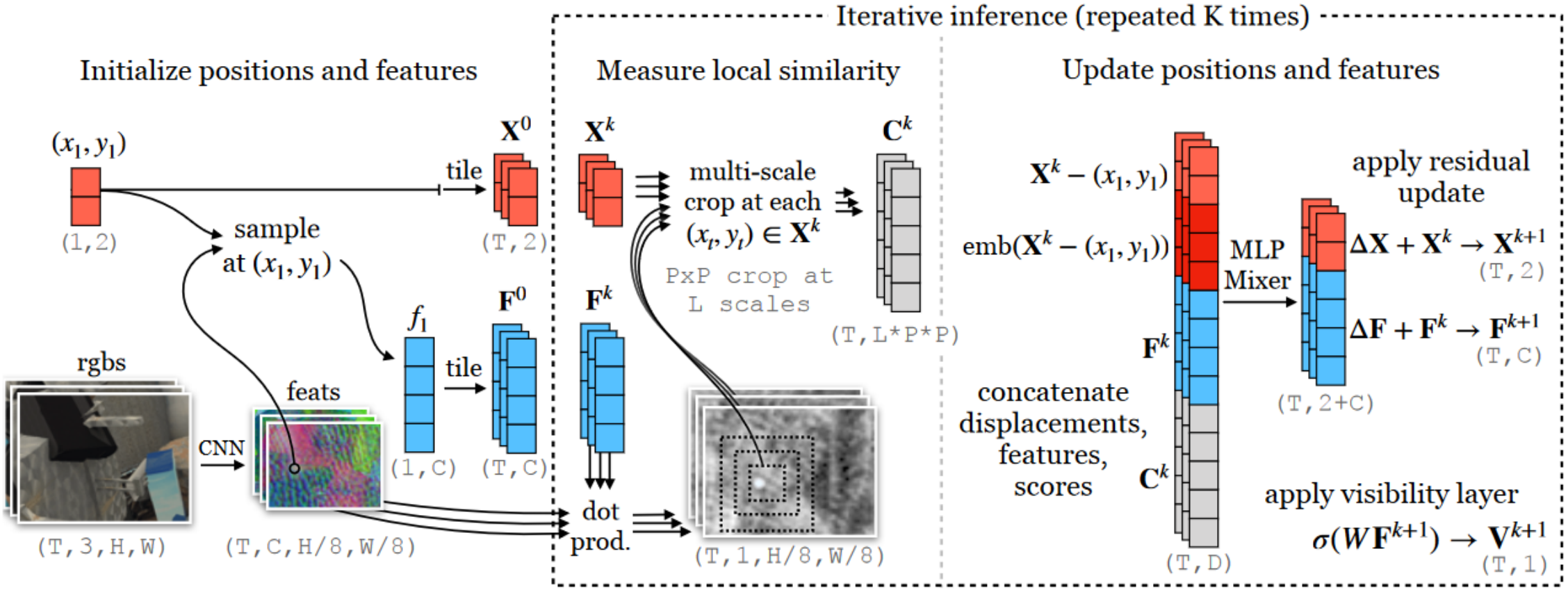
Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Particle Video Revisited



Particle Video Revisited: Tracking Through Occlusions Using Point Trajectories. Harley et. al. ECCV 2022

Single Point Tracking

- Background points
- Object points

- Tracking single points lacks global consistency
- Points drift relative to each other

Tracking with Optical Flow

- Background points
- Object points

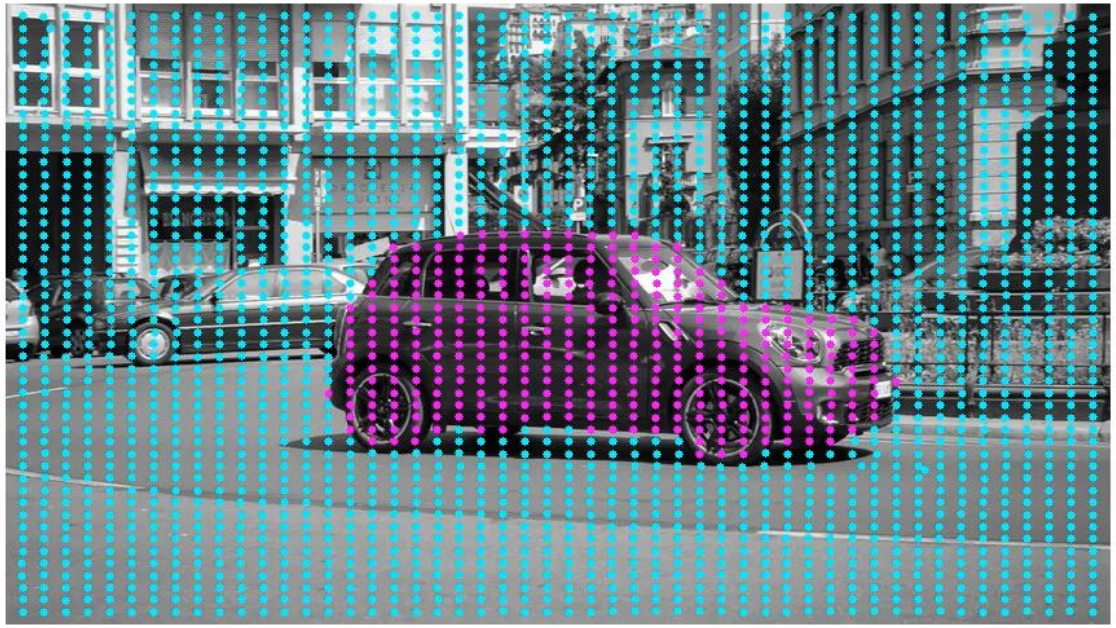
- Global consistency
- Occlusion problem: background accumulates on the object

Tracking points together

- Global consistency from tracking many points
- Occlusion handling from long-term tracking
- Train on synthetic data only
- Main mechanism:
  - Transformer architecture
  - Factorised attention: space, time, and group (across tracks)

CoTracker: It is Better to Track Together, N Karaev et al., ECCV 2024

Tracking Points Together



- Background points
- Object points

- Better global consistency
- Better occlusion handling

Tracking Points Together

Tracking Points Together

Problem: Long Occlusions

CoTracker3

Same new architecture, trained in two flavours:

- Offline model: look at the whole video at once
- Online model: windowed forward tracking

- Simpler architecture: 50% less parameters, 0.2ms/(frame*point)

- Also predict confidence for each track

- Fine-tuned by pseudo-labelling 10k real videos

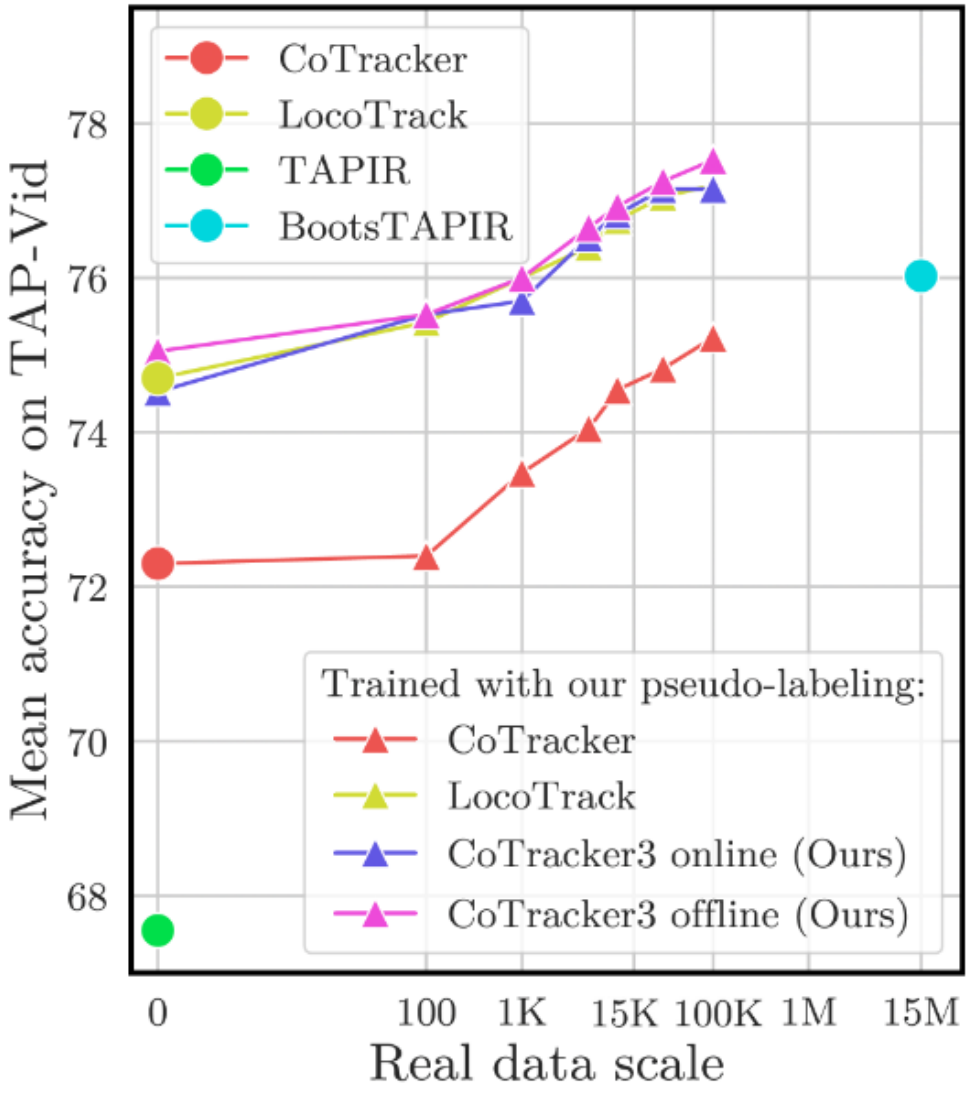
CoTracker3: Simpler and Better Point Tracking by Pseudo-Labelling Real Videos, N Karaev et al., arxiv 2024, cotracker3.github.io

Long Occlusions: Offline Tracker

Scaling on real videos

- Train model on synthetic data
- Pseudo-label real videos with it
- Train on pseudo-labelled videos + simple augmentations

Scaling Trackers



A note on correspondences

- Many computer vision problems are correspondence problems “at heart”.
- Optical flow: pixel correspondences through time
- Stereo/disparity estimation
- Tracking
- Retrieval
- Multi-view reconstruction
- Etc.

A note on correspondences

- Many of the algorithms developed for one problem, can be used for the others as well.
- Lucas-Kanade was originally used for optical flow, then templates, then extended to more complex warps.
- LK can be combined with learning/features instead of intensities.
- Models now tend to learn features that generalise for many tasks.
- “Old” ideas are complementary to learned models.