Real-Time Video Stabilization for Embedded Systems

By: Aditya Pola

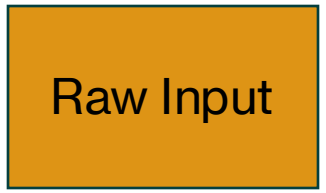
Motivation

Handheld cameras introduce unintentional jitter from small hand movements

This motion reduces visual quality and affects computer vision tasks

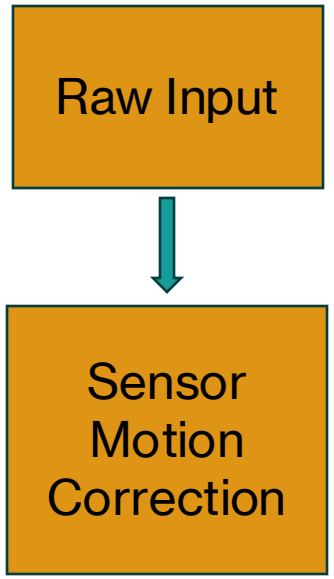
Goal: estimate camera motion and remove unintended movement

Pipeline Overview



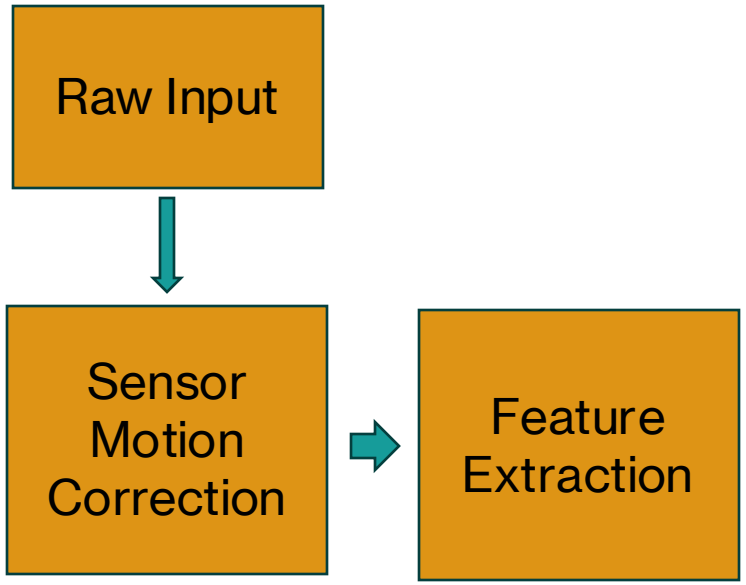
Obtain live video input from camera

Pipeline Overview



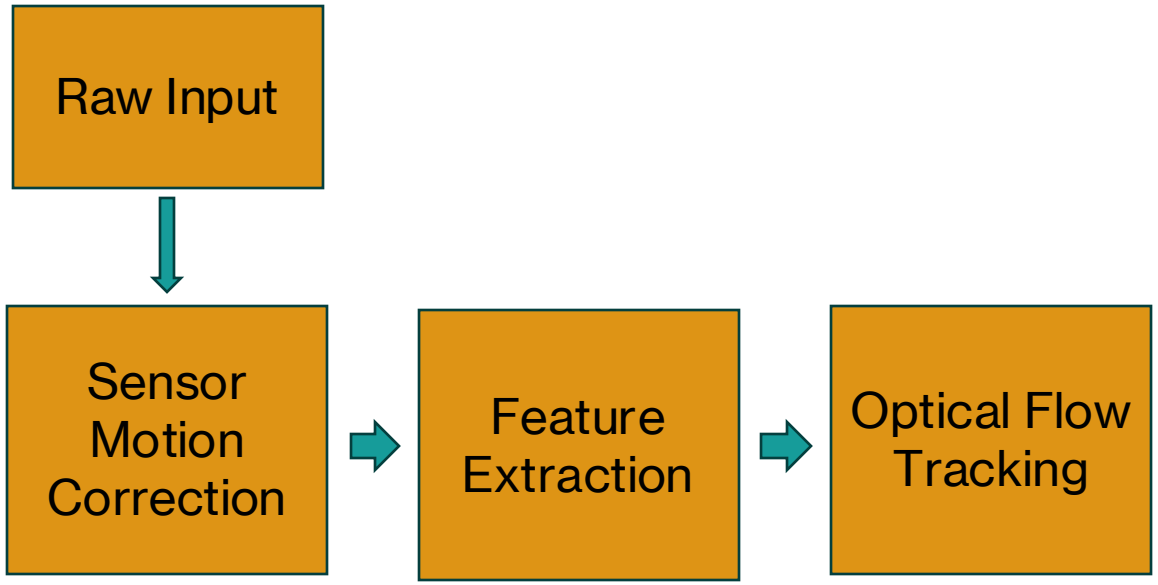
Estimate the camera’s movement using gyroscope measurements to provide an initial motion estimate.

Pipeline Overview



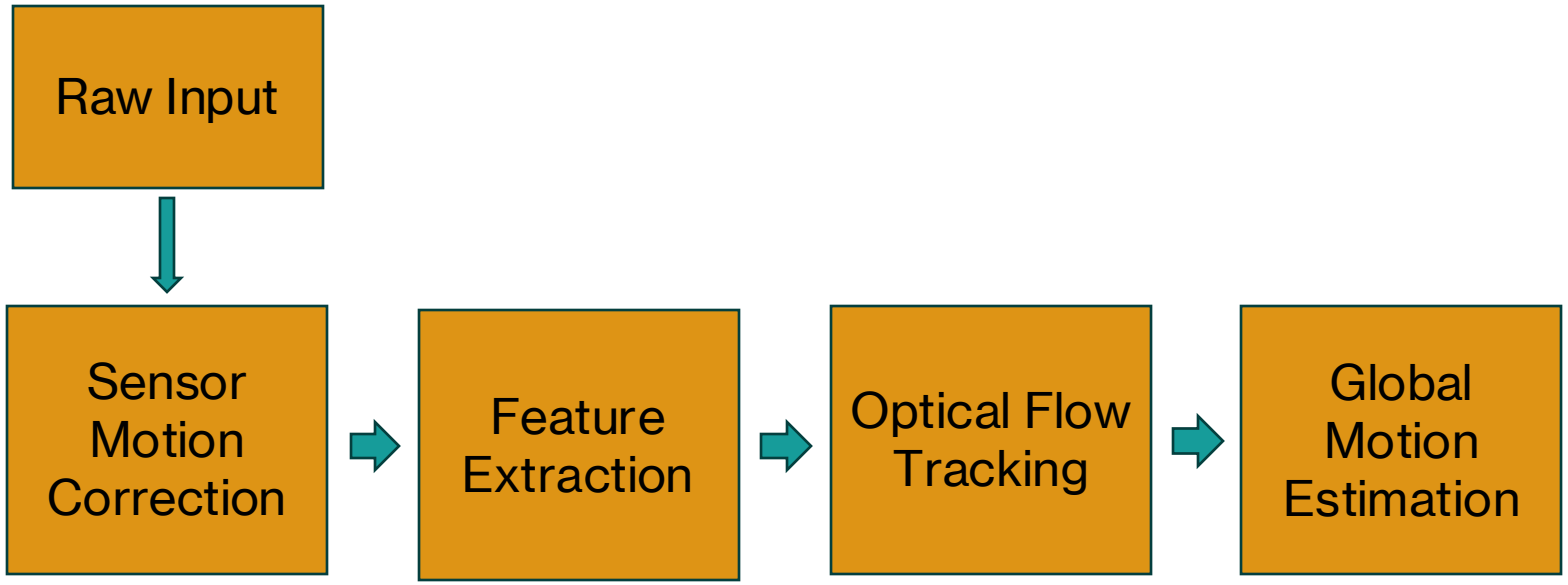
Identify stable points in the image that can be reliably tracked between frames.

Pipeline Overview



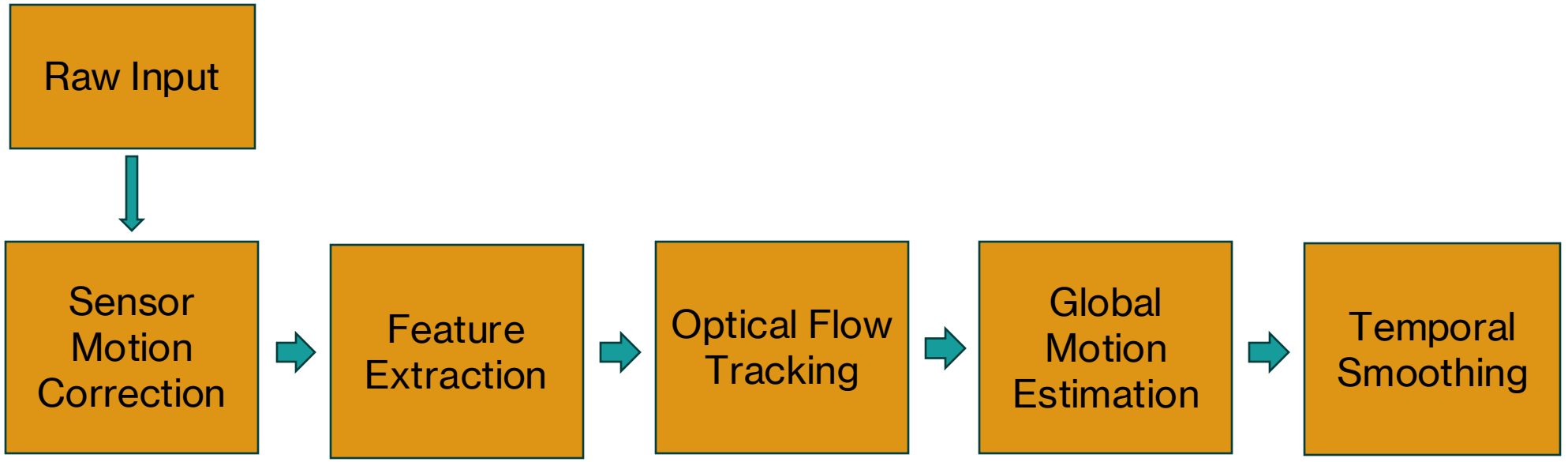
Track how detected feature points move between consecutive frames.

Pipeline Overview



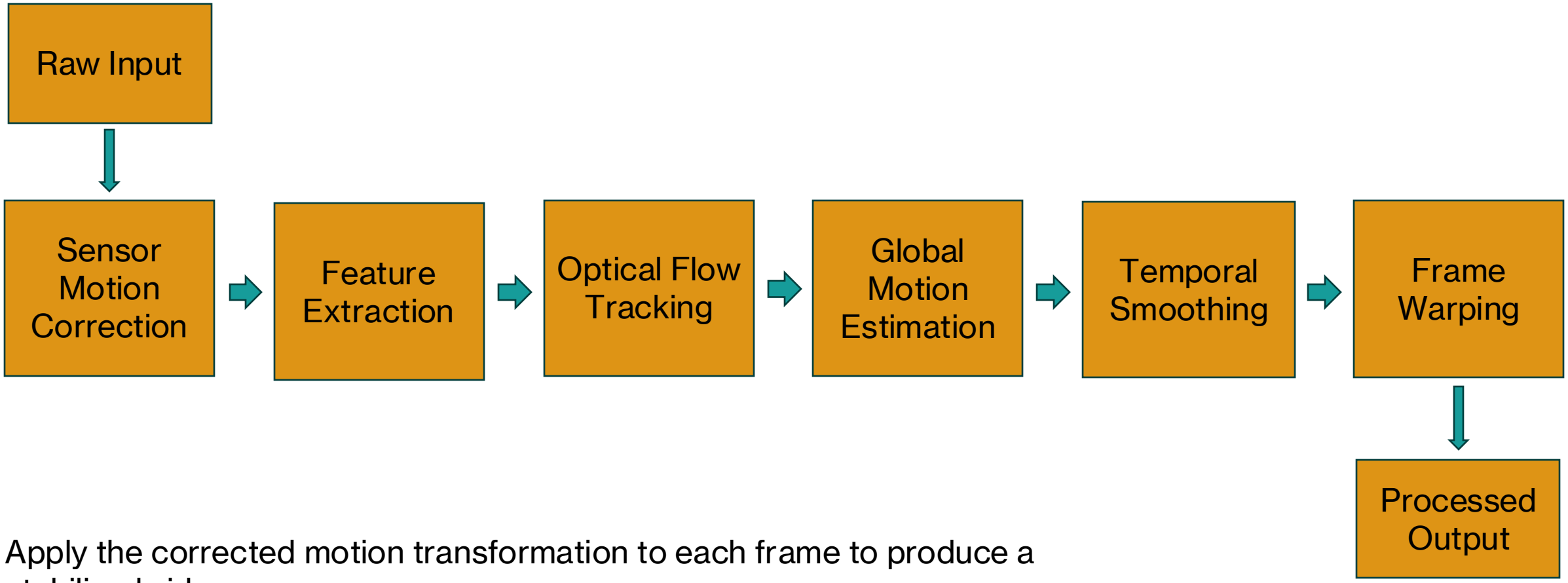
Combine feature motions to estimate the overall camera movement between frames.

Pipeline Overview



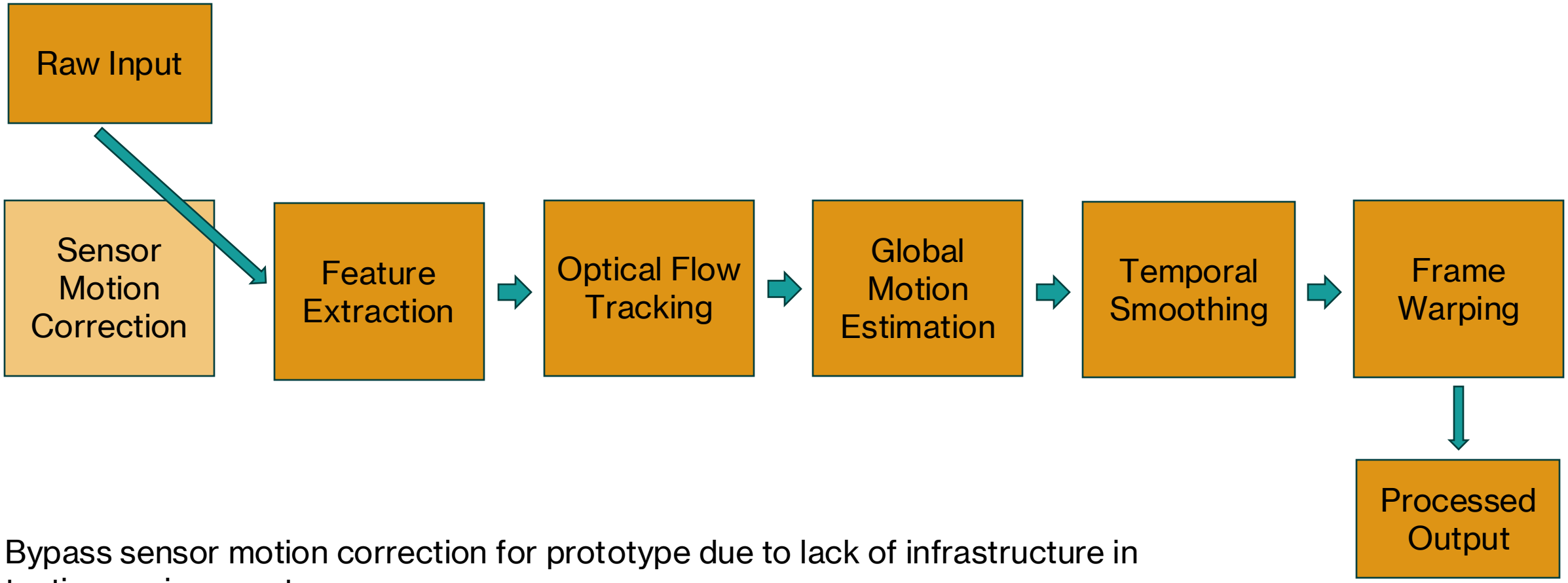
Smooth the estimated camera motion over time to remove high-frequency jitter.

Pipeline Overview



Apply the corrected motion transformation to each frame to produce a stabilized video.

Prototype Pipeline Overview



Bypass sensor motion correction for prototype due to lack of infrastructure in testing environment.

Raw Video Samples [Static Case]

Raw Video Samples [Multi Subject Case]

Raw Video Samples [Dynamic Case]

Sensor Motion Correction [Gyro Motion Est.]

- Gyroscopes measure **angular velocity** of the camera over time.
- Camera orientation can be estimated by **integrating angular velocity**.

$$R(t) = \int \omega(t)\, dt$$

- Plan to eventually utilize orientation estimates to construct space-transformation matrix

$$\mathbf{x}' = \mathbf{H}_{gyro}\mathbf{x}$$

Sensor Motion Correction [Gyro Motion Est.]

- Sensor data provides a **fast coarse estimate of camera motion**.
- Does not depend on scene features, so it works even in **low-texture environments**.
- However, sensors suffer from **noise and long-term drift**.

- We will see how it goes when it gets there...

Feature Extraction [Shi-Tomasi Corners]

- Good tracking points must contain **intensity variation in multiple directions**
- Local image structure is captured by the **structure tensor**

$$M = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- I_x - Horizontal image gradient $\quad \mathrm{I}_x = \frac{\partial I}{\partial x}$
- I_y - Vertical image gradient $\quad \mathrm{I}_y = \frac{\partial I}{\partial y}$

- Eigenvalues λ_1 & λ_2 measure **directional intensity variation in orthogonal directions**

Feature Extraction [Shi-Tomasi Corners]

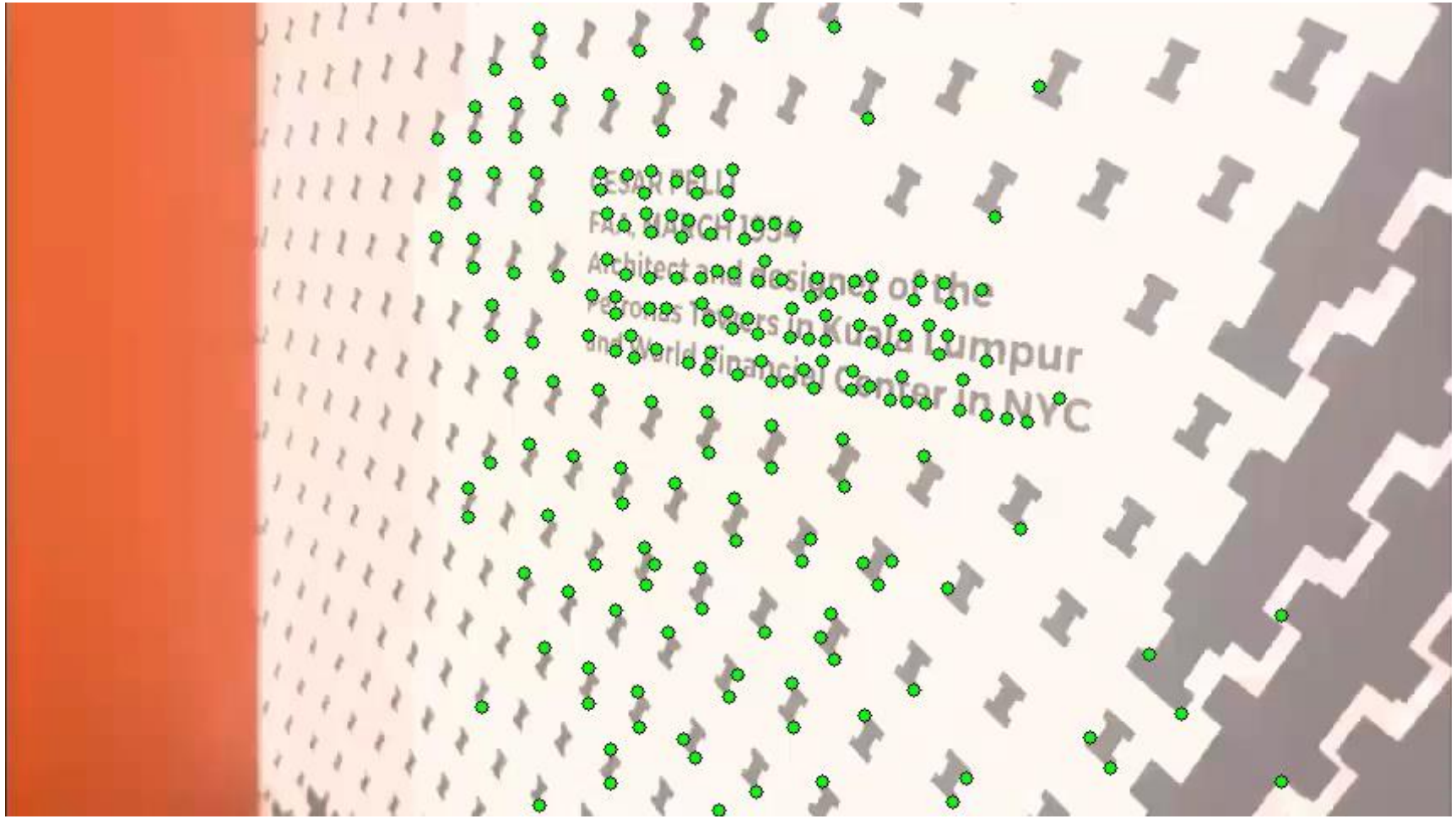
- Corners are selected using the **minimum eigenvalue test**

$$\min(\lambda_1, \lambda_2) > \tau$$

- Candidates are selected if:
  - Both eigenvalues are large -> Strong intensity changes in both directions

- Selected features are:
  - **Stable under small motion**
  - **Easy to localize across frames**

Stage 1 Video Samples [Static Case]



Stage 1 Video Samples [Multi Subject Case]

Stage 1 Video Samples [Dynamic Case]

Optical Flow Tracking [Lucas-Kanade Method]

- After detecting corners, we **track their motion between frames**
- Lucas–Kanade assumes **brightness constancy**:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

- Pixel intensity remains approximately **constant between subsequent frames**
- Linearizing the motion gives the **optical flow constraint equation**

$$I(x+u, y+v, t+1) \approx I(x, y, t) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = I_x u + I_y v + I_t = 0$$

- (u,v) - pixel displacement between frames

Optical Flow Tracking [Lucas-Kanade Method]

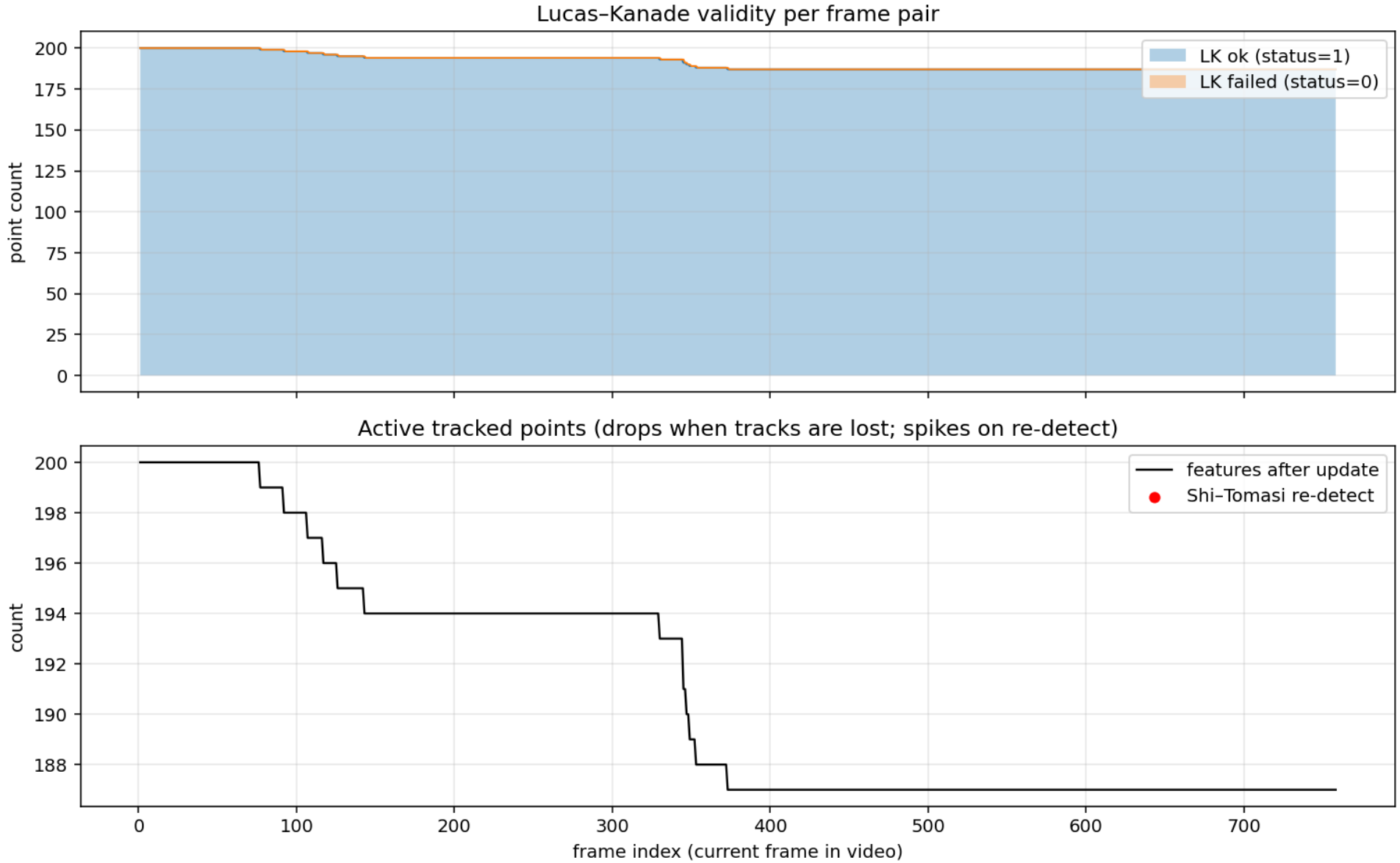
- A single pixel equation cannot determine both motion components (u,v)
- Lucas–Kanade assumes **neighboring pixels move together**
- Motion is estimated over a **small image patch**
- Solve the least-squares system

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

- Produces **feature correspondences between frames**
- These correspondences are used to estimate **global camera motion**

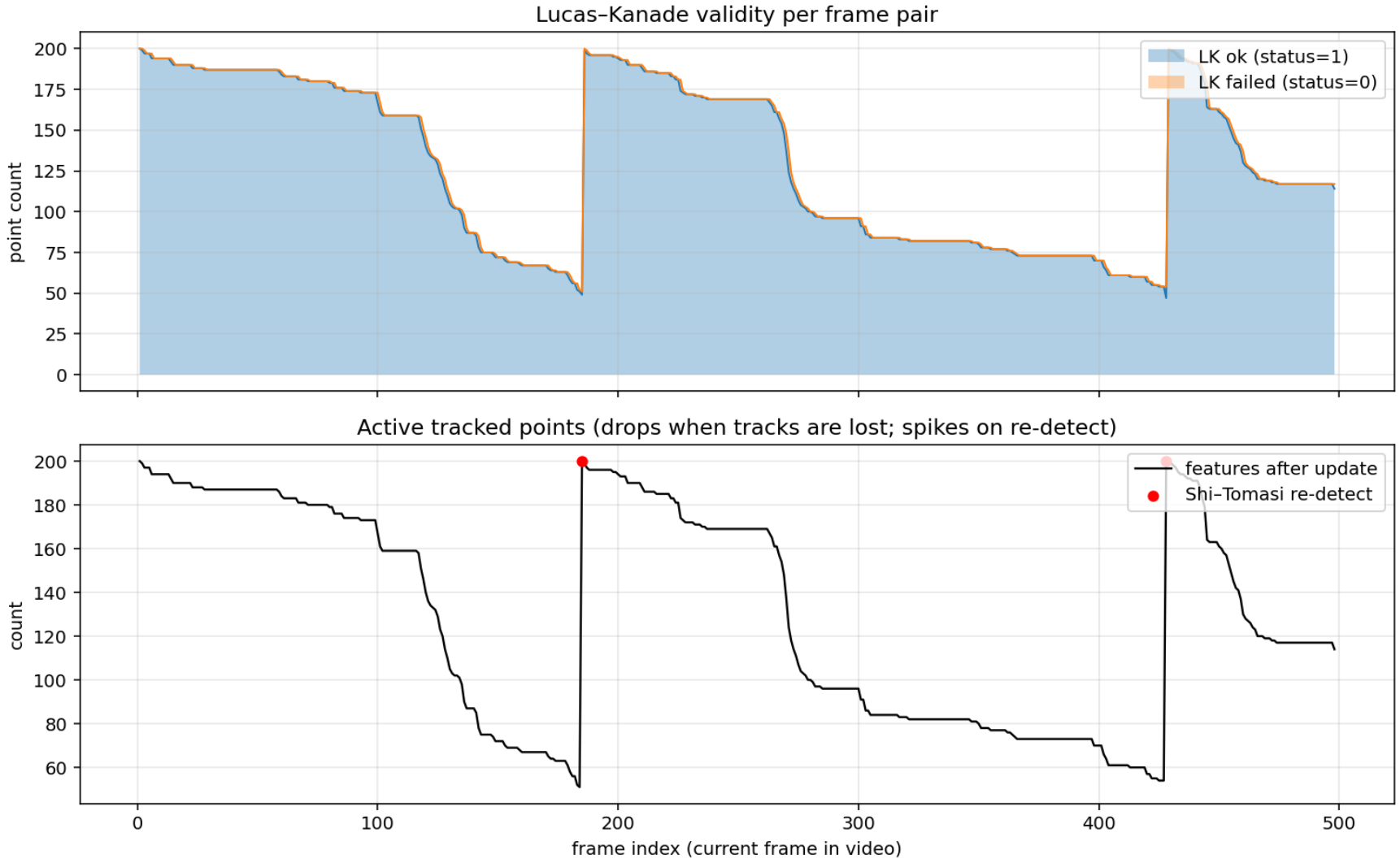
Stage 2 Video Samples [Static Case]

Point Tracking Metrics [Static Case]



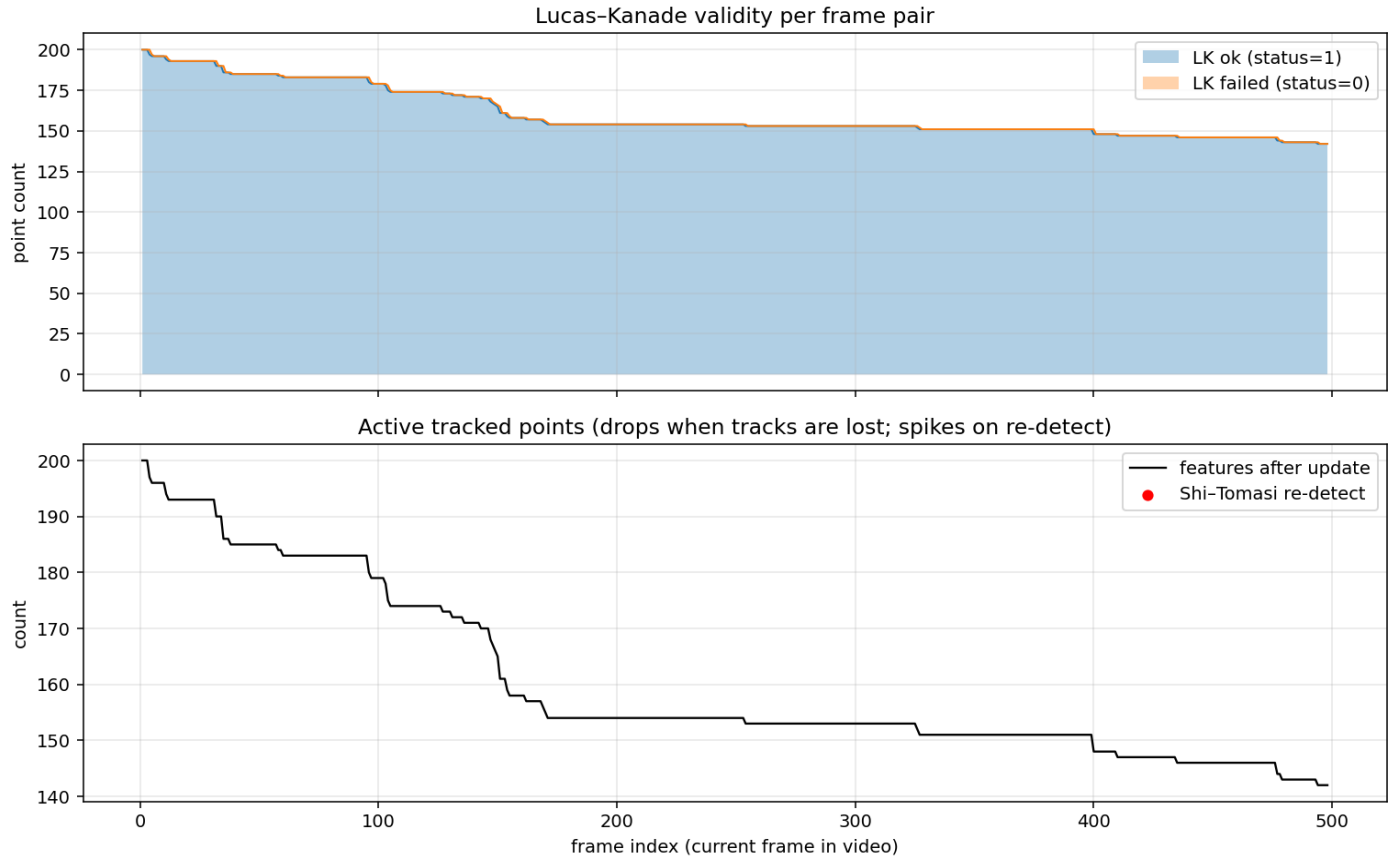
Stage 2 Video Samples [Multi Subject Case]

Point Tracking Metrics [Multi Subject Case]



Stage 2 Video Samples [Dynamic Case]

Point Tracking Metrics [Dynamic Case]



Global Motion Estimation [Affine Transform]

- Optical flow produces **many individual feature displacements**
- Not all motion corresponds to **camera motion**
  - moving objects
  - tracking noise
  - mismatched features
- Goal: estimate the **dominant camera motion** between frames
- We model camera motion using an **affine transformation**

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Global Motion Estimation [Affine Transform]

- The affine transform models several geometric functions to estimate motion

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

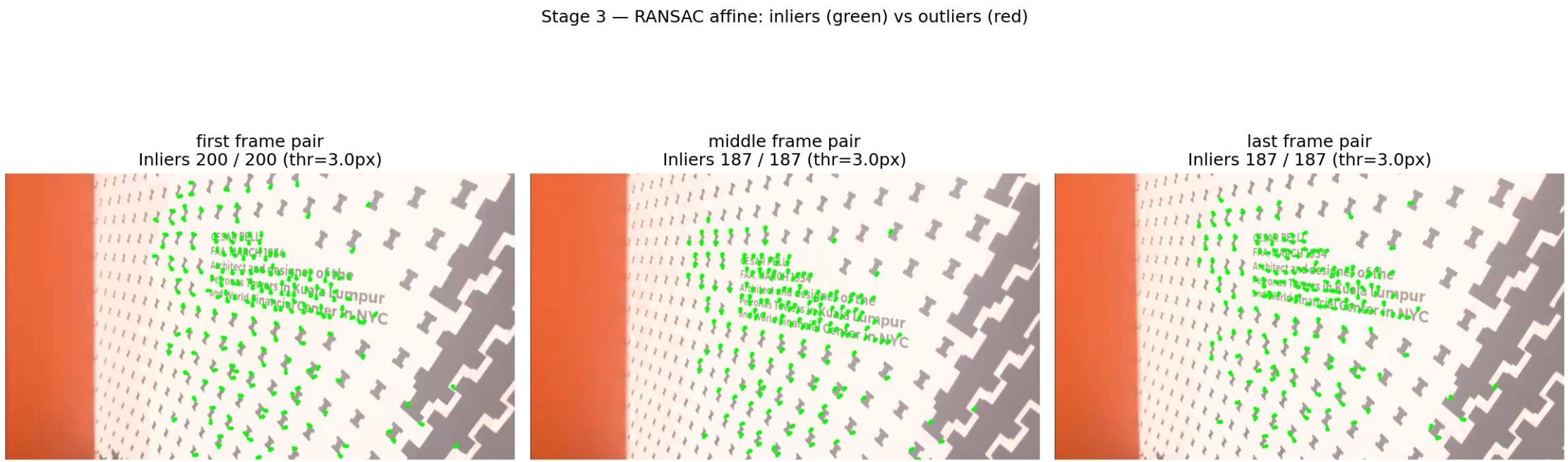
Shear

Global Motion Estimation [RANSAC]

- Some feature matches correspond to **independent object motion**
- Direct fitting would be **sensitive to outliers**
- RANSAC estimates the affine transform **robustly**
- **Procedure**
  - Randomly select a **small subset of feature matches**
  - Compute candidate **affine transform**
  - Apply transform to all matches
  - Identify **inliers** (points consistent with motion)
  - Repeat for finite interval and choose model with **largest inlier set**

Stage 3 Video Samples [Static Case]

RANSAC Inliers & Outliers [Static Case]



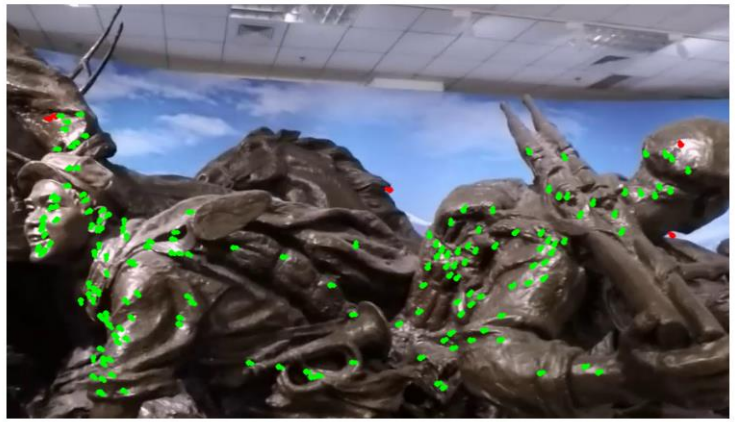
Stage 3 Video Samples [Multi Subject Case]

RANSAC Inliers & Outliers [Multi Subject Case]

Stage 3 — RANSAC affine: inliers (green) vs outliers (red)

first frame pair
Inliers 199 / 200 (thr=3.0px)

middle frame pair
Inliers 164 / 169 (thr=3.0px)



last frame pair
Inliers 85 / 114 (thr=3.0px)

Stage 3 Video Samples [Dynamic Case]

RANSAC Inliers & Outliers [Dynamic Case]

Stage 3 — RANSAC affine: inliers (green) vs outliers (red)

first frame pair
Inliers 178 / 200 (thr=3.0px)

middle frame pair
Inliers 154 / 154 (thr=3.0px)

last frame pair
Inliers 112 / 142 (thr=3.0px)

Temporal Smoothing [Kalman Filter]

- Frame-to-frame affine estimation produces a **sequence of motion parameters**
- These parameters form a **camera motion trajectory over time**
- Raw motion estimates often contain **high-frequency jitter**
- Directly applying them would produce **unstable video**
- Goal: smooth the motion trajectory while **preserving intentional movement**

Temporal Smoothing [Kalman Filter]

- We model camera motion as a **hidden state evolving over time**
- Predicted Step

$$\hat{x}_k = x_{k-1}$$

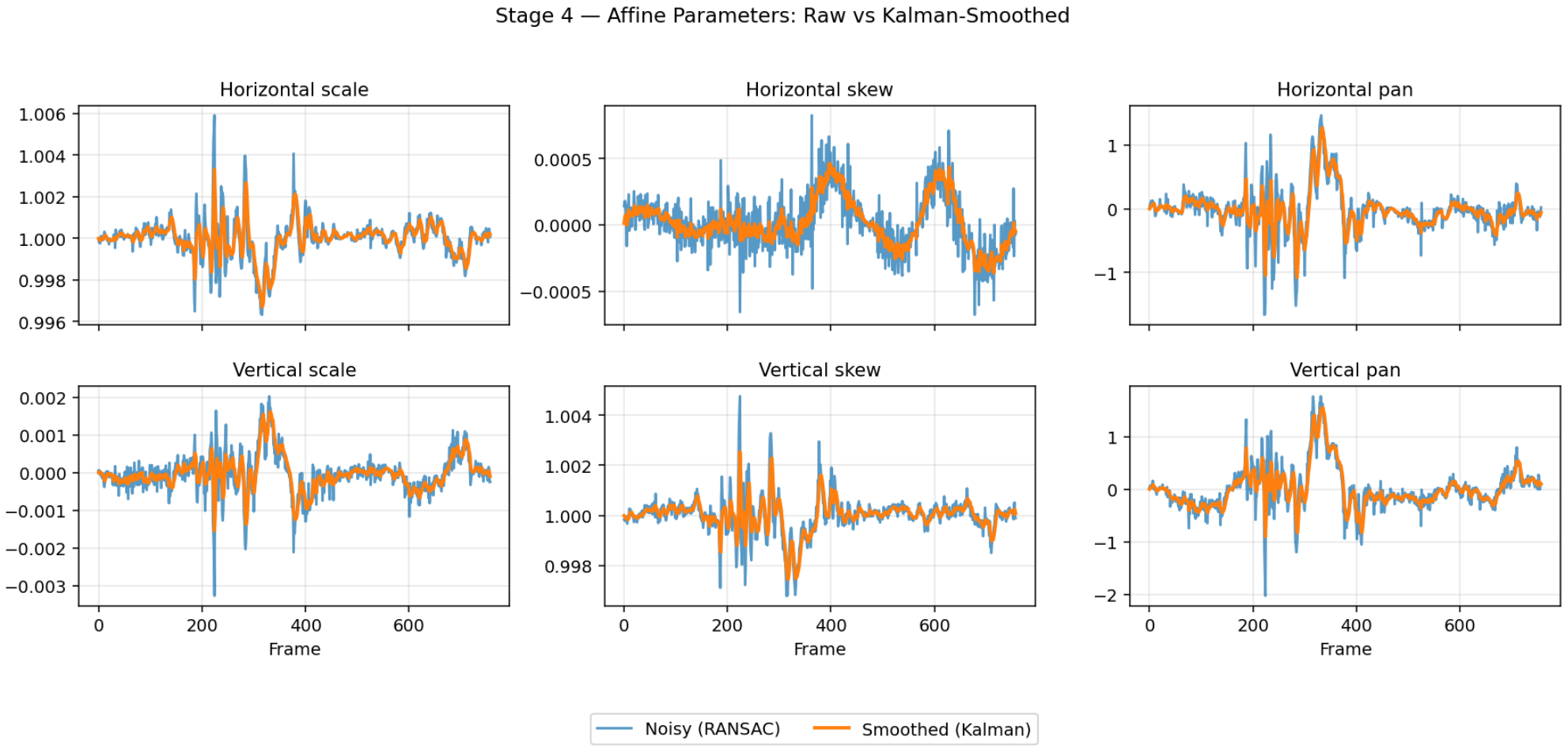
- Correction Step:

$$x_k = x_{k-1} + K\left(z_k - x_{k-1}\right)$$

- System state is computed from previous system state and current measured states
- Removes outliers by applying scalar 'K' to measurement temporal error

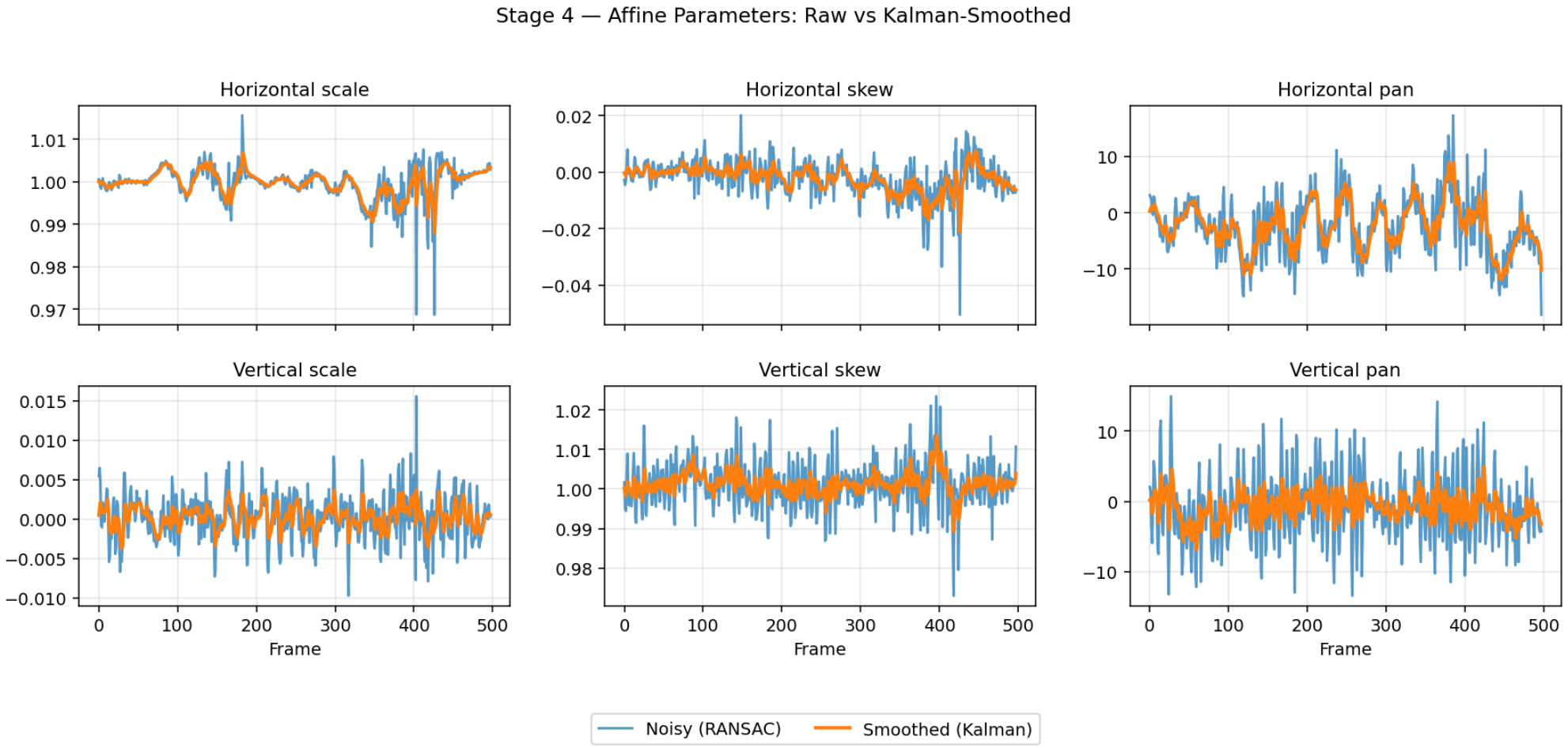
Stage 4 Video Samples [Static Case]

Affine Parameters [Static Case]



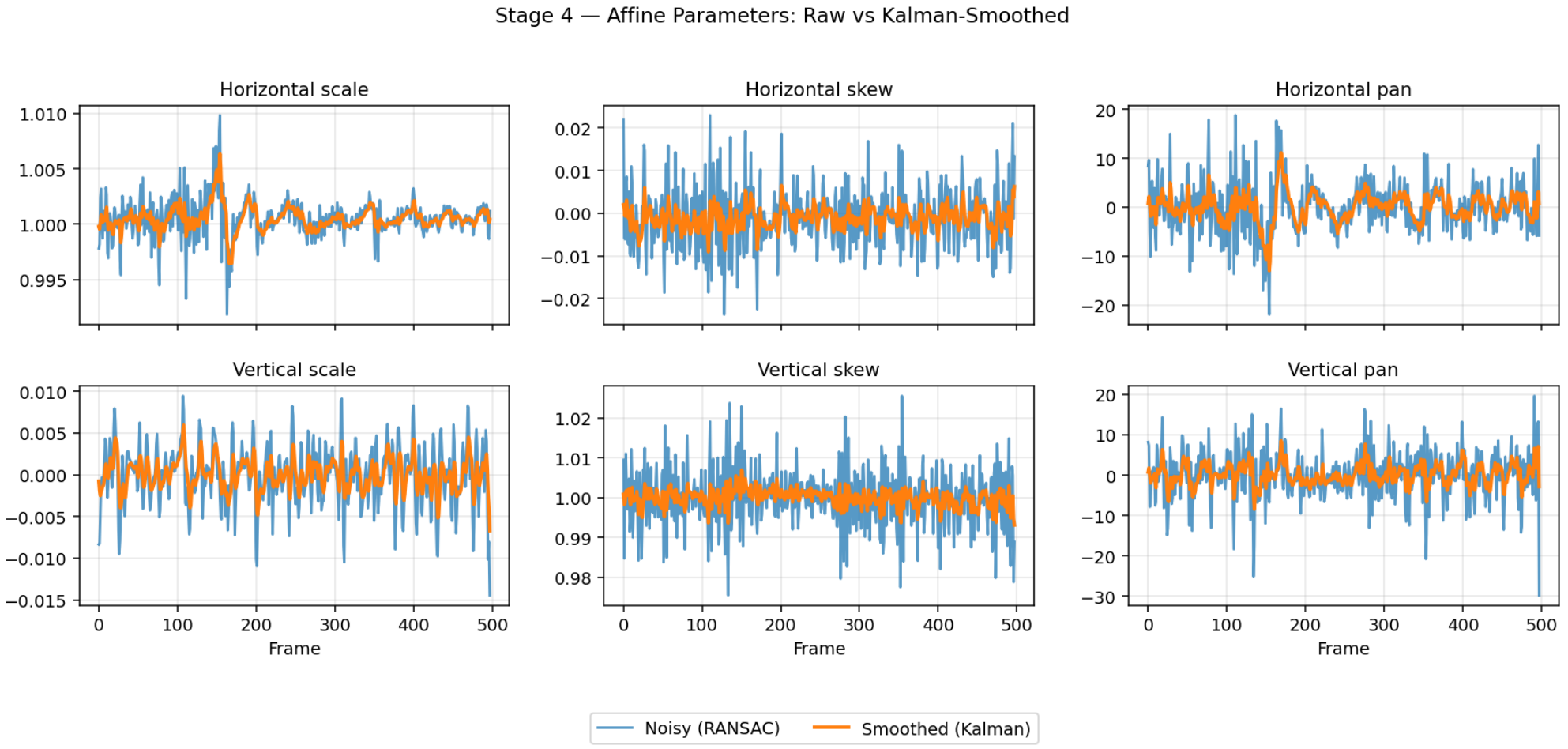
Stage 4 Video Samples [Multi Subject Case]

Affine Parameters [Multi Subject Case]



Stage 4 Video Samples [Dynamic Case]

Affine Parameters [Dynamic Case]



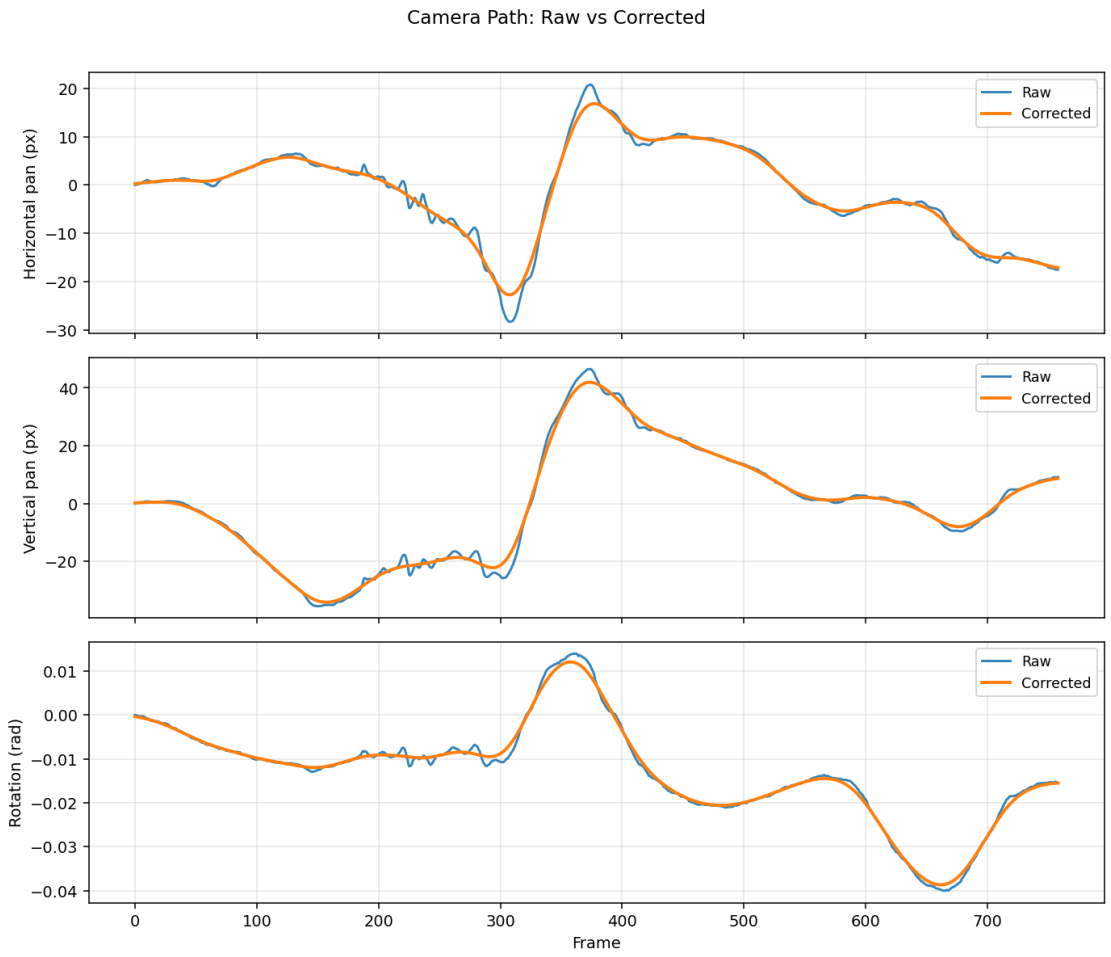
Frame Warping [Inverse Motion Transform]

- The smoothed trajectory represents the **desired camera motion**
- Stabilization is achieved by applying the **inverse of the smoothed motion**
- Each frame is transformed to **cancel the estimated camera movement**

$$\mathbf{x}_{stab} = \mathbf{H}_{smooth}^{-1}\mathbf{x}$$

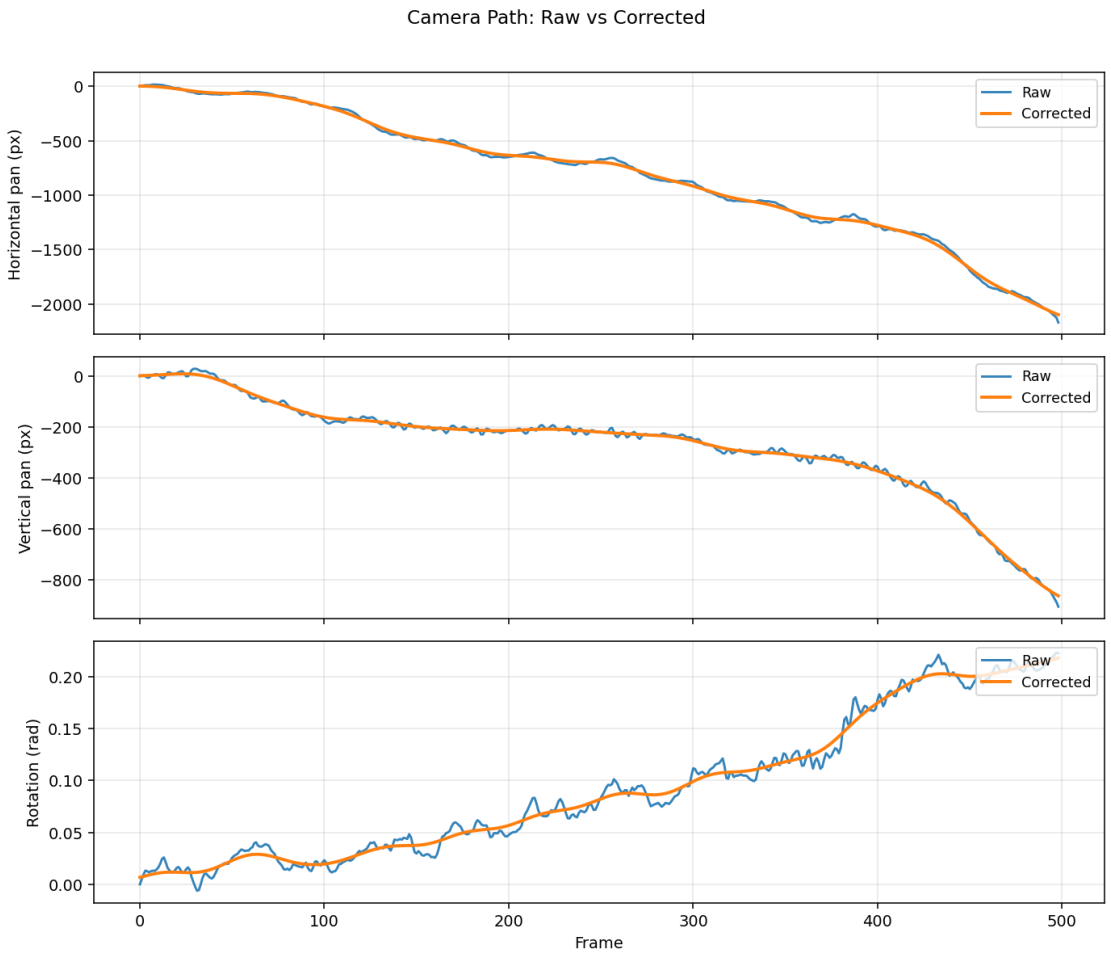
Stabilized Video Samples [Static Case]

Stabilized Path [Static Case]



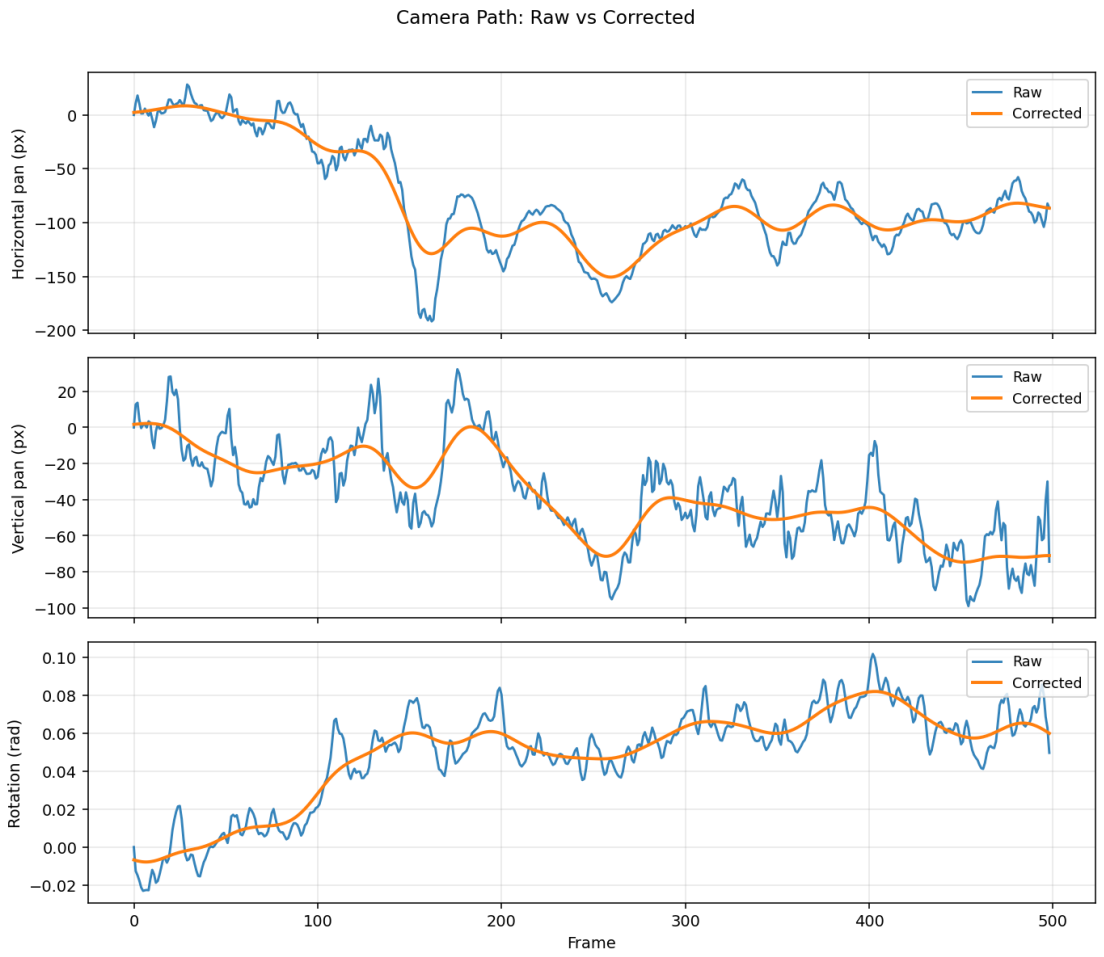
Stabilized Video Samples [Multi Subject Case]

Stabilized Path [Multi Subject Case]



Stabilized Video Samples [Dynamic Case]

Stabilized Path [Dynamic Case]



Limitations

- **Moving objects / multiple subjects**
  Difficult to distinguish subject motion from camera motion; large moving objects can bias the estimated transform.
- **Low-texture scenes**
  Feature detection and tracking degrade when there are few corners or gradients (e.g., blank walls, sky).
- **Large or rapid camera motion**
  Lucas–Kanade assumes small inter-frame motion; fast motion can break tracking.
- **Parallax and depth variation**
  A single affine transform assumes roughly planar motion; scenes with strong depth changes may produce distortion.
- **Accumulated drift over long sequences**
  Small estimation errors can accumulate over time, affecting trajectory stability.

References

- Lim, Anli & Ramesh, Bharath & Yang, Yue & Xiang, Cheng & Gao, Zhi & Lin, Feng. (2019). Real-Time Optical flow-based Video Stabilization for Unmanned Aerial Vehicles. Journal of Real-Time Image Processing. 16. 10.1007/s11554-017-0699-y.
- Deniz, O.; Bueno, G.; Bermejo, E.; Sukthankar, R. (2011). Fast and accurate global motion compensation. Pattern Recognition. Volume 44, Issue 12. Pages 2887–2901. ISSN 0031-3203. https://doi.org/10.1016/j.patcog.2010.10.019.
- https://zhangleibit.github.io/download/stabfr/dataset.html