No Labels, No Look-Ahead: Unsupervised Online Video Stabilization with Classical Priors

Supplementary Material

The supplementary material additionally provides the following:

- Implementation details of the motion propagation networks;
- Implementation details of the trajectory smoothing networks;
- Details of the multi-threaded asynchronous buffering pipeline;
- Details of frame outpainting;
- Information on the multimodal UAV dataset;
- Runtime comparison on embedded platforms;
- Comprehensive description of the evaluation metrics used in this work;
- Per-scene quantitative evaluation of online methods;
- Additional visualizations of key modules and optical flow;
- A 3-minute video demonstrating comparisons with online methods;
- Limitations and future directions.

A. Details of EfficientMotionPro

A.1. Training Objectives

We propose a grid-based motion propagation network (**EfficientMotionPro**). Building upon multi-homography priors, this module explicitly encodes the relationship between sparse keypoint residuals and vertex distances. Through lightweight feature extraction, attention-weighted aggregation, and residual decoding, it achieves end-to-end propagation from sparse keypoints to a dense grid, thereby yielding a more robust global motion field. We additionally employ projection consistency and structure-preservation objectives for training.

(1) Homography Projection Constraint To enhance the geometric fidelity of the propagated motion field, we introduce a homography-based projection loss. Specifically, the estimated keypoint motion is projected via a local homography mapping generated from the predicted grid motion vectors. The Euclidean distance between this projected position and the actual ground-truth keypoint position after motion constitutes the projection error:

$$\mathcal{L}_{\text{proj}} = \frac{1}{|\Omega_{kp}|} \sum_{p \in \Omega_{kp}} \omega_{t,p} \sqrt{\left\| \mathbf{u}_{t,p} - H_t^{\text{local}}(p;\, \Delta g_t) \cdot p \right\|_2^2 + \epsilon^2}. \tag{1}$$

Here, $H_t^{\text{local}}(p;\, \Delta g_t)$ denotes the local homography transformation, obtained via bilinear sampling from the grid motion field Δg_t surrounding point p. This term effectively constrains the grid deformation predicted by the residual network to be geometrically plausible, reducing inconsistent distortions.

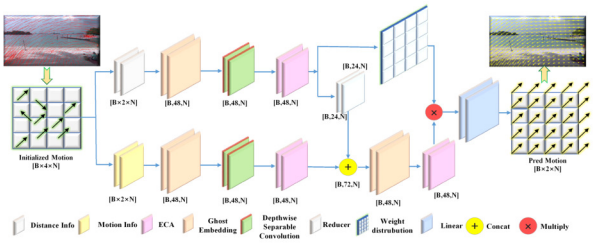

Figure 1. The architecture of our EfficientMotionPro network.

(2) Structure-Preservation Constraint In low-texture regions, the propagation estimate is prone to local distortions or discontinuous boundaries. To maintain the rigid structure of the mesh, we design a structure-preserving loss based on the angular relationships between adjacent vertices. For each grid cell defined by its four vertices $\{g_{i,j}, g_{i+1,j}, g_{i,j+1}, g_{i+1,j+1}\}$, we enforce orthogonality between adjacent edges:

$$\mathcal{L}_{\text{struct}} = \frac{1}{|G|} \sum_{i,j} \left(\frac{\langle g_{i+1,j} - g_{i,j},\ R_{90^\circ}(g_{i,j+1} - g_{i,j}) \rangle}{\|g_{i+1,j} - g_{i,j}\| \cdot \|g_{i,j+1} - g_{i,j}\|} \right)^2, \tag{2}$$

where R_{90° denotes a 90° rotation operator, $\langle \cdot, \cdot \rangle$ is the dot product, and $|G|$ is the total number of grid cells. This loss penalizes deviations from orthogonality, suppressing shear and anisotropic scaling, thereby preserving local shape and preventing distortion.

(3) Total Loss The final optimization objective for the motion-propagation module is defined as the weighted combination of all three losses:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{kp}} + \lambda_2 \mathcal{L}_{\text{proj}} + \lambda_3 \mathcal{L}_{\text{struct}}, \tag{3}$$

with default settings $(\lambda_1, \lambda_2, \lambda_3) = (10, 40, 40)$.

A.2. Network Architecture

Fig. 1 illustrates the architecture of our EfficientMotionPro network. Each frame produces an input tensor of size $[B, 4, N]$ constructed from N keypoints: two channels encode distance (or distance-derived) features and two channels encode residual motion. Distance and motion branches use

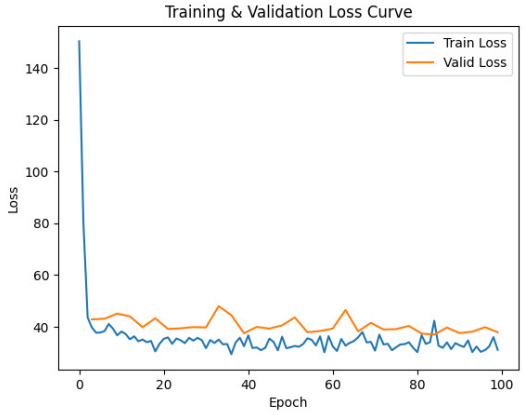

Figure 2. Training curves of **EfficientMotionPro**: the blue line shows training loss and the orange line shows validation loss.

Ghost [7] modules for lightweight embeddings and depthwise separable convolutions; motion features are refined with ECA [26] attention. Features are fused via a Lightweight Fusion Block, aggregated with attention weights, and decoded by a two-layer MLP to yield residuals $[B, 2]$. The final displacements are reshaped into a grid field $[H/P, W/P, 2]$ and smoothed by median pooling. Table 1 details the configuration.

(1) Implementation Details We train **EfficientMotionPro** on an in-house unstable video dataset [9, 33, 34] with train/val splits. Batch size is 64; optimizer is Adam $(\beta_1 = 0.9, \beta_2 = 0.999)$ with weight decay 1×10^{-5}. Learning rate follows OneCycleLR, peaking at 2×10^{-4} with 30% warm-up, then cosine decay. Training is conducted for 100 epochs on a single NVIDIA RTX 4090 GPU, taking approximately 12 hours. We save checkpoints every 3 epochs and keep the best validation loss as shown in Fig. 2.

(2) Complexity Analysis

Parameter Count Let C_e be the embedding dimension (default $C_e = 48$). Core components include Ghost modules, depthwise separable convolutions (DW), pointwise convolutions (PW, kernel 1), and ECA attention. At $C_e = 48$, the whole module has $\sim 2.3 \times 10^4$ (≈ 22.9K) parameters, significantly fewer than those in typical full-frame 2D CNN or Transformer-based propagation heads.

Time Complexity Let N be the number of keypoints per frame. Since convolutions, softmax, and weighted aggregation are along the keypoint dimension with small kernels, the dominant MACs come from PW convolutions:

$$\begin{aligned}&\mathcal{O}(NC_e^2)\ \text{[PW: dist reduction } C_e \to C_e/2]\\&+\mathcal{O}(4NC_e^2)\ \text{[PW: motion } 2C_e \to 2C_e]\\&+\mathcal{O}(NC_e^2)\ \text{[PW: motion } C_e \to C_e]\\&+\mathcal{O}(NC_ek)\ \text{[DW blocks]}\ +\ \mathcal{O}(N)\ \text{[aggregation]}\\&\approx \mathcal{O}(5NC_e^2).\end{aligned} \tag{4}$$

Thus the cost scales linearly with N and is largely decoupled from image resolution or grid size; the cost of median pooling is negligible due to its small kernel size.

Memory Complexity Intermediate tensors scale as $\mathcal{O}(BC_eN)$ (independent of HW).

Complexity Estimation Numerical MAC estimates (ignoring BN/activations/ECA) with $C_e = 48$ are shown in Table 2.

B. OnlineSmoother Details

B.1. Training Objectives

In dynamic scenes or under low-frame-rate conditions, grid vertex trajectories often exhibit local oscillations due to jitter, occlusion, or non-rigid disturbances, which compromise temporal consistency and introduce geometric distortions. To achieve online stabilization with minimal latency and without using future frames, we introduce the **OnlineSmoother** network. It leverages only the historical trajectory data from a sliding buffer, employing a Lite LS-3D [24] encoder to extract spatio-temporal features and a Star-gated [16] decoder to predict dynamic smoothing kernels for the x and y directions separately. This kernel-based iterative update mechanism progressively suppresses high-frequency oscillations while preserving motion intentionality. We additionally employ spatial consistency and projection consistency objectives for training.

(1) Spatial Distortion Constraint To maintain the local geometric structure of the mesh during temporal transformation, we introduce a distortion loss based on triangular mesh elements. The smoothed trajectories are mapped back to the grid coordinate system to construct triangular structures (eight varying configurations per quad). The loss penalizes deviations in edge length ratios and angles from their initial reference state:

$$\begin{aligned}\mathcal{L}_{\text{spatial}} = \frac{1}{|G|}\sum_{i,j}\Bigg[&\lambda_{\text{edge}}\sum_{\text{edges } e}\sqrt{\left(\frac{\|e\|}{\|e_0\|}-1\right)^2+\epsilon^2}\\&+\lambda_{\text{angle}}\sum_{\text{angles }\theta}\sqrt{\left(\frac{\theta}{\theta_0}-1\right)^2+\epsilon^2}\Bigg]\end{aligned} \tag{5}$$

Table 1. Architecture of the proposed **EfficientMotionPro** module. The input tensor $[B, 4, N]$ is split into distance and motion channels, embedded via Ghost modules, enhanced with ECA, fused, and finally decoded into grid displacements.

Component	Operation	Output Size
Input	Distance & Motion channels	$[B, 4, N]$
Distance branch	GhostModule + DWConv + PWConv	$[B, 48, N]$
	Feature reduction	$[B, 24, N]$
	Distance attention extractor	$[B, 1, N]$
Motion branch	GhostModule + DWConv + PWConv	$[B, 48, N]$
	ECA attention enhancement	$[B, 48, N]$
Fusion	Concat (Dist+Motion) $\rightarrow$ FusionBlock	$[B, 48, N]$
Aggregation	Attention-weighted sum	$[B, 48]$
Decoder	2-layer MLP	$[B, 2]$
Reshape + MedianPool	Grid displacement field	$[H/P,\ W/P,\ 2]$

Table 2. Estimated MACs per frame of EfficientMotionPro under different keypoint counts N.

Keypoints N	**Estimated MACs / frame**
128	≈ 2.14M
256	≈ 4.28M
512	≈ 8.55M

where e_0 and θ_0 represent the edge lengths and angles of the triangles in the initial, undeformed reference mesh g_{ref}, and $|G|$ is the total number of grid cells.

(2) Keypoint Projection Consistency Constraint To further ensure the alignment between the smoothed motion field and the observed keypoint trajectories, we introduce a homography-based projection supervision. Specifically, local homography transformations are constructed from the smoothed grid motion field S_t. The original keypoint positions are then projected using these homographies, and a consistency error is computed against their actual warped positions:

$$\mathcal{L}_{\text{proj}} = \frac{1}{|\Omega_{kp}|} \sum_{p \in \Omega_{kp}} \omega_{t,p}\, \rho\big(\mathcal{W}(p; O_t),\ H_t^{\text{local}}(p; S_t) \cdot p\big)\,, \tag{6}$$

where $\rho(\cdot, \cdot)$ denotes the Charbonnier penalty $\rho(a, b) = \sqrt{\|a - b\|_2^2 + \epsilon^2}$, and $H_t^{\text{local}}(p; S_t)$ represents the local homography estimated from the smoothed grid S_t at point p. This term effectively enhances the accuracy and geometric plausibility of the smoothed trajectory field at the locations of real keypoints.

(3) Total Loss The final optimization objective for the trajectory smoothing module is defined as the weighted combination of all **three** losses:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{temp}} + \lambda_{\text{spatial}} \mathcal{L}_{\text{spatial}} + \lambda_{\text{proj}} \mathcal{L}_{\text{proj}}, \tag{7}$$

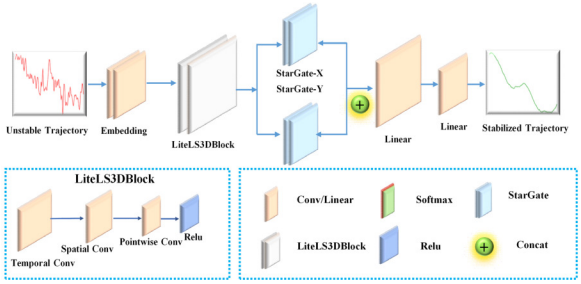

Figure 3. The architecture of our OnlineSmoother network.

where $\lambda_{\text{time}} = 20$, $\lambda_{\text{freq}} = 1$, $\lambda_{\text{spatial}} = 10$, and $\lambda_{\text{proj}} = 5$ are the default hyperparameters inside/alongside $\mathcal{L}_{\text{temp}}$ (time/freq) and the two auxiliary terms.

B.2. Network Architecture

Fig. 3 illustrates the architecture of our OnlineSmoother network. Given frame-wise displacements, we form a causal trajectory tensor $[B, 2, T, H, W]$ (two channels for x/y). **OnlineSmoother** consists of: trajectory encoder (Linear $(2 \rightarrow 64)$ + ReLU), spatiotemporal modeling (Lite LS-3D [24]: temporal large-kernel DW-Conv3d + spatial $1\times3\times3$ DW-Conv3d, optional $1\times1\times1$ PW-Conv3d for mixing), Stargated [16] dynamic-kernel decoder (two linear heads for gating/amplitude and one scale head), and kernel-guided iterative smoothing. Table 3 summarizes the design.

(1) Implementation Details We train and evaluate **OnlineSmoother** on the same self-collected unstable video dataset (with sequence-level train/val splits) that is also used for **EfficientMotionPro**.

Batch size is 1 to preserve causality. Optimizer: Adam $(\beta_1 = 0.9, \beta_2 = 0.99)$, weight decay 1×10^{-5}. Learning rate: OneCycleLR with maximum 2×10^{-4}, warm-up for the first 30% steps, then cosine annealing. The smoothing loss combines $\mathcal{L}_{\text{time}}$, $\mathcal{L}_{\text{spatial}}$, and $\mathcal{L}_{\text{proj}}$ with default weights

Table 3. Architecture of the **OnlineSmoother** module. Per-vertex processing; outputs are per-vertex kernels and smoothed trajectories.

Submodule	Tensor Shape (Input → Output)	Structural Components
Trajectory Encoder	$[B, 2, T, H, W] \to [B, 64, T, H, W]$	Linear $(2 \to 64)$ + ReLU
Lite LS-3D (Temporal)	$[B, 64, T, H, W] \to [B, 64, T, H, W]$	DW-Conv3d $(k_t \times 1 \times 1$, dilation $d_t)$ + ReLU
Lite LS-3D (Spatial)	$[B, 64, T, H, W] \to [B, 64, T, H, W]$	DW-Conv3d $(1 \times 3 \times 3)$ + ReLU
(Optional) Channel Mixing	$[B, 64, T, H, W] \to [B, 64, T, H, W]$	PW-Conv3d $(1 \times 1 \times 1$, groups=4) + ReLU
Kernel Decoder (x)	$[64] \to [6]$ (per vertex)	Linear (gate), Sigmoid; Linear (amplitude)
Kernel Decoder (y)	$[64] \to [6]$ (per vertex)	Same as x
Scale Decoder (shared)	$[64] \to [1]$ (per vertex)	Linear (scale)
Kernel Map	$[B, 12, T, H, W]$	Element-wise mult., Star-gated fusion
Kernel-guided Smoothing	$[B, 2, T, H, W] \to [B, 2, T, H, W]$	Iterative consistency refinement (6-step kernel)

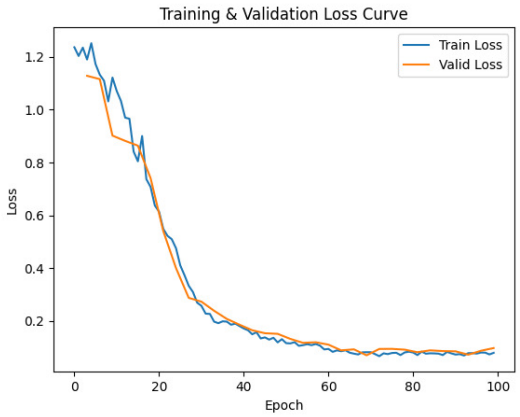

Figure 4. Training curves of **OnlineSmoother**: blue for training loss, orange for validation loss.

$(\lambda_{\text{time}}, \lambda_{\text{spatial}}, \lambda_{\text{proj}}) = (20, 10, 5)$ and internal $\lambda_{\text{freq}} = 1$. Training runs for 100 epochs; gradient clipping (threshold 5.0) ensures numerical stability.

Validation is performed every 3 epochs; the best model based on the validation loss (Fig. 4) is saved. Training is conducted on a single NVIDIA RTX 4090 GPU (about 2.5 hours).

(2) Complexity Analysis

Parameter Count. Let $C = 64$ (embedding dim), temporal kernel $k_t = 5$, and optional PW mixing controlled by `use_pointwise`. Counts (approx.): Encoder Linear $(2 \to C)$: $2C$; DW-temporal: $\mathcal{O}(Ck_t)$; DW-spatial: $\mathcal{O}(9C)$; (Opt.) PW-Conv3d (groups=4): $\approx C^2/4$; Star-gated decoder: two Linear $(C \to 6)$ per direction $(2 \cdot 6C$ each) + one Linear $(C \to 1)$ scale. Total (no PW): for $C = 64, k_t = 5$ is $\sim$ 2.6K; with PW (groups=4) adds $\sim$ 1.0K, totaling $\sim$ 3.6K.

Time Complexity. Let batch B, temporal window T, grid size $H \times W$. Dominant MACs (ignoring BN/activations/Sigmoid) come from per-vertex Linear and depthwise 3D convolutions:

$$\mathcal{O}\big(BTHW \cdot (2C + Ck_t + 9C)\big) \tag{8}$$

plus decoder heads $\approx \mathcal{O}(BTHW \cdot 25C)$ and (opt.) PW mixing $\mathcal{O}(BTHW \cdot C^2/\text{groups})$. Overall (no PW):

$$\mathcal{O}\big(BTHW \cdot (2C + Ck_t + 9C + 25C)\big). \tag{9}$$

Kernel-guided updates with R rounds and six history steps are scalar-weighted and negligible relative to the forward pass (even with $R = 20$).

Table 4. Estimated MACs of the OnlineSmoother backbone at different grid sizes ($C = 64$, $k_t = 5$, $T = 7$, $B = 1$).

Window T	Grid $H \times W$	MACs / frame
7	16×16	≈ 4.6M
7	32×32	≈ 18.4M
7	48×48	≈ 41.5M

Memory Complexity. Feature maps are $[B, C, T, H, W]$; kernel heads $[B, 6, T, H, W]$ and $[B, 1, T, H, W]$. Thus memory is $\mathcal{O}(BCTHW)$, independent of input image resolution.

Numerical Estimation (Default $C = 64, k_t = 5, T = 7$). Ignoring BN/activations/Sigmoid and with $B = 1$:

$$\text{MACs} \approx T \cdot H \cdot W \cdot C \cdot (k_t + 35) \tag{10}$$

For $H = W = 32$:

$$7 \cdot 32 \cdot 32 \cdot 64 \cdot 40 \;\approx\; 18.4\,\text{M} \tag{11}$$

We summarize per-grid costs in Table 4.

C. Multi-Threaded Asynchronous Buffering Pipeline: Additional Details

(1) Motivation and Formulation Although each individual module is computationally lightweight, executing them

in sequence leads to the accumulation of stage latencies:

$$T_{\text{serial}} = t_{\text{est}} + t_{\text{prop}} + t_{\text{smooth}}, \tag{12}$$

where t_{est}, t_{prop}, and t_{smooth} denote the average per-frame runtimes of the **Motion Estimation**, **Motion Propagation**, and **Motion Compensation** stages, respectively. This serial bottleneck prevents strict real-time performance.

(2) Pipeline Architecture To mitigate this, we decompose the stabilization pipeline into three concurrent daemon threads, decoupled by bounded FIFO (First-In-First-Out) shared queues with back-pressure (threads block on empty/full states). This design ensures safe resource handling even under exceptional conditions:

- **$\mathbf{T_{ME}}$ (Motion Estimation):** Acquires input frames I_t, performs keypoint collaboration detection $\mathcal{D}$, homogenization via SSC, and sparse keypoint-guided causal flow fusion to produce $\mathbf{m}_t = [x_{kp}; y_{kp}; u; v]$ and the reweighted flow field $\hat{\mathbf{f}}_{t \leftarrow t-1}$.
- **$\mathbf{T_{MP}}$ (Motion Propagation):** Consumes outputs from T_{ME}. Executes the EfficientMotionPro network, which computes the multi-homography prior, predicts per-vertex residuals $\Delta g_{\text{res},t}$ via the lightweight backbone, and outputs the full-frame grid motion field Δg_t.
- **$\mathbf{T_{MC}}$ (Motion Compensation):** Consumes grid motions Δg_t (or integrated trajectories O_t) from T_{MP}. Applies the **OnlineSmoother** network for online causal smoothing, yielding the stabilized trajectory S_t. Finally, it computes the compensation field $M_t = S_t - O_t$, performs frame warping via $\mathcal{W}(x; M_t)$, and outputs the stabilized frame $\tilde{I}_t$.

Threads communicate exclusively through the following shared queues:

$$\begin{aligned} &Q_{\text{ME}\to\text{MP}}: \\ &\quad \text{Transmits } \mathbf{m}_t \text{ (keypoint motion features),} \\ &\quad \text{plus metadata,} \\ &Q_{\text{MP}\to\text{MC}}: \\ &\quad \text{Transmits } \Delta g_t \text{ (grid motion field),} \\ &\quad \text{plus metadata.} \end{aligned} \tag{13}$$

(3) Theoretical Analysis Let the steady-state per-frame service times be $\mathbf{t} = (t_{\text{est}}, t_{\text{prop}}, t_{\text{smooth}})$ and the capacity of each queue be C:

- **Throughput (Steady-State):** The pipeline's frame rate is bounded by the slowest stage.

$$\begin{aligned} \eta_{\text{pipe}} &\approx \frac{1}{\max\{t_{\text{est}}, t_{\text{prop}}, t_{\text{smooth}}\}}, \\ \text{FPS}_{\max} &\approx \frac{1}{\max\{t_{\text{est}}, t_{\text{prop}}, t_{\text{smooth}}\}}. \end{aligned} \tag{14}$$

- **Theoretical Speedup (vs. Serial Execution):**

$$S = \frac{t_{\text{est}} + t_{\text{prop}} + t_{\text{smooth}}}{\max\{t_{\text{est}}, t_{\text{prop}}, t_{\text{smooth}}\}}. \tag{15}$$

- **End-to-End Latency:** The delay for a frame to traverse the entire pipeline is reduced from the serial sum to the duration of the longest stage plus queuing delays, which are minimal for adequately sized buffers.
- **Memory Overhead:** The memory footprint is dominated by the bounded queues.

$$\mathcal{M} = \mathcal{O}\big(C_{\text{ME}} \cdot |\mathbf{m}_t| + C_{\text{MP}} \cdot |\Delta g_t|\big), \tag{16}$$

where $|\mathbf{m}_t|$ and $|\Delta g_t|$ represent the size of the data packets transmitted between threads, and $C_{\text{ME}}, C_{\text{MP}}$ are the capacities of the respective queues.

(4) Complexity and Real-Time Properties

- **Computational Complexity:** Pipeline parallelism does not change the asymptotic complexity $\mathcal{O}(\cdot)$ of each individual stage; it primarily reduces the wall-clock time from the sum $t_{\text{est}} + t_{\text{prop}} + t_{\text{smooth}}$ to the maximum $\max\{t_{\text{est}}, t_{\text{prop}}, t_{\text{smooth}}\}$.
- **Memory/Bandwidth:** The memory and inter-thread communication bandwidth scale linearly with the chosen queue capacities. This overhead is typically negligible compared to the memory required for processing batches of frames or implementing global buffering and reordering strategies.
- **Real-Time Performance:** If the service times are balanced ($t_{\text{est}} \approx t_{\text{prop}} \approx t_{\text{smooth}}$), the throughput approaches a $3\times$ speedup over serial execution. If one stage is the dominant bottleneck, the overall throughput is determined by that stage's processing rate, while the latency for an individual frame is still reduced to approximately the duration of that longest stage.

D. Frame Outpainting Details

When the stabilized video is cropped, we first compute the cropping ratio and then determine the scaling required for outpainting. The scaling factors are derived from the cropped dimensions defined by the frame borders.

Cropping Ratio and Cropped Size Let $W_{\text{orig}}, H_{\text{orig}}$ be the original width and height. Let B_{hor} and B_{ver} denote the per-side horizontal (left/right) and vertical (top/bottom) border thickness (in pixels), respectively. Then

$$\begin{aligned} W_{\text{cropped}} &= W_{\text{orig}} - 2\,B_{\text{hor}}, \\ H_{\text{cropped}} &= H_{\text{orig}} - 2\,B_{\text{ver}}. \end{aligned} \tag{17}$$

The cropping ratio (content preservation) is

$$C \;=\; \frac{W_{\text{cropped}}\,H_{\text{cropped}}}{W_{\text{orig}}\,H_{\text{orig}}} \in (0, 1]. \tag{18}$$

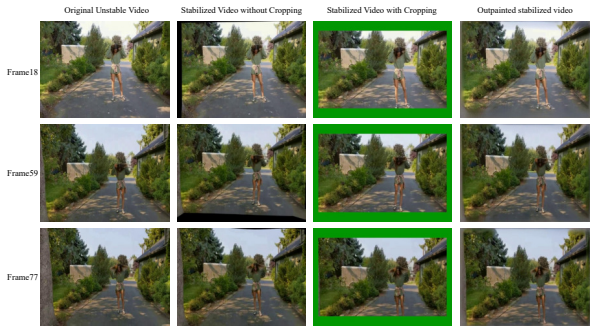

Figure 5. Left to right: Original unstable video; Stabilized (uncropped); Stabilized (cropped); Stabilized (with frame outpainting).

Scaling Factors for Outpainting To compensate for the cropped regions, we define the anisotropic scaling factors

$$\text{scale}_w = \frac{W_{\text{orig}}}{W_{\text{cropped}}} = \frac{W_{\text{orig}}}{W_{\text{orig}} - 2\,B_{\text{hor}}}, \tag{19}$$

$$\text{scale}_h = \frac{H_{\text{orig}}}{H_{\text{cropped}}} = \frac{H_{\text{orig}}}{H_{\text{orig}} - 2\,B_{\text{ver}}}. \tag{20}$$

If aspect ratio must be preserved (recommended), use an isotropic factor

$$s = \max\big(\text{scale}_w,\ \text{scale}_h\big) \quad (s \geq 1). \tag{21}$$

Outpainting with the ProPainter Model ProPainter [32] is then applied to the missing boundary regions induced by the target scaling. In practice, we first warp each stabilized frame to the outpainted canvas using the chosen factor (scale_w, scale_h or s), and then invoke ProPainter to fill the uncovered areas near the borders. This preserves the original content while minimizing visible seams at frame boundaries. Figures 5 illustrates the difference between the pre- and post-stabilization video with outpainting.

E. Additional Details for UAV-Test Dataset

To comprehensively evaluate robustness and generalization in complex real-world conditions, we introduce a new multimodal UAV aerial video dataset (UAV-Test). This dataset captures instability patterns common in aerial operations, such as high-frequency vibrations, aggressive maneuvers, and occlusions. Figures 6 and 7 present sample visible-light and infrared scenes.

Data Collection Protocol Data were captured using a DJI Matrice 300 RTK UAV equipped with a DJI Zenmuse H20N gimbal camera, featuring both a wide-angle visible-light sensor and a long-wave infrared sensor. The visible-light lens supports optical zoom from 4.5–360 mm, while the infrared lens (8–14 μm) has a fixed focal length of 150 mm.

Table 5. Per-frame runtime comparison on Jetson AGX Orin. We report the average per-frame runtime (ms) and FPS. Our method is significantly faster than MeshFlow, StabNet, and NNDVS, and second only to the highly lightweight Liu et al. [14].

Method	Runtime (ms)	FPS
MeshFlow [13]	168.95	5.92
StabNet [25]	179.88	5.56
NNDVS [28]	340.02	2.94
Liu et al. [14]	36.41	27.47
Ours	78.94	12.67

The shutter speed was set to automatic, and ISO was set within 100–1600. Videos were recorded at 1280×720 and 1920×1080 resolutions, with frame rates of 25–30 FPS, simulating varying transmission bandwidths and computational loads.

Scene Distribution UAV-Test comprises 92 video sequences totaling ∼26 minutes, covering five categories:

- **Urban Areas** (19 sequences, 4.5 min): Building circumnavigation and traversal under wind levels 3–4.
- **Highways** (22 sequences, 6.3 min): High-speed parallel tracking under wind levels 2–3.
- **Forested Mountains** (18 sequences, 5.9 min): Canopy penetration and terrain tracking under wind levels 3–5.
- **Waterfronts** (20 sequences, 6.0 min): Disturbance from water-surface reflections under wind levels 3–4.
- **Industrial Sites** (13 sequences, 3.2 min): Occlusions from machinery under wind level 3.

Annotation and Cleaning Corrupted frames caused by transmission errors were automatically detected and removed using `ffmpeg`, ensuring clean sequences for evaluation.

Partitioning and Licensing UAV-Test is designated as a pure testing set. A stratified sampling strategy based on scene type, weather (sunny/rainy/foggy), and time of day (day/night) ensures diversity. The dataset will be released with source code; access can be requested via email. All flights complied with airspace regulations, and privacy-sensitive information such as human faces and license plates was blurred.

F. Runtime Analysis

While many stabilization algorithms achieve real-time performance on high-end GPUs, their efficiency often drops sharply on embedded platforms. To assess practical efficiency, we benchmarked our method on the Jetson AGX Orin and compared it with MeshFlow [13], StabNet [25], NNDVS [28], and Liu et al. [14]. Experimental results

Figure 6. Representative examples of unstable UAV visible-light videos from diverse scenarios, including urban areas, highways, forested mountains, waterfronts, and industrial sites.

show that, thanks to its lightweight architecture and multi-threaded buffering, our method runs at ∼13 FPS (78.94 ms per frame). This is markedly faster than MeshFlow, StabNet, and NNDVS, and second only to the extremely lightweight approach of Liu et al., demonstrating a favorable trade-off among accuracy, real-time performance, and deployment adaptability. *Note:* all timings exclude the optional frame-outpainting post-processing step; only the time to produce a stabilized frame is measured for every method.

G. Evaluation Metrics and Implementation

For reproducibility, we explicitly define the three core metrics used in the main paper, following [12], together with implementation details.

Cropping Ratio (C) For each frame, a global homography between the input and the stabilized output is estimated, and the scaling components are used to compute the cropping ratio. The overall cropping ratio C is defined as the average across all frames. Let the effective field of view after stabilization be Ω_t and the original field of view be Ω, then:

$$C = \frac{1}{T}\sum_{t=1}^{T}\frac{|\Omega_t|}{|\Omega|}. \tag{22}$$

Implementation details. The cropping ratio is estimated from the scaling factors s_x, s_y of the frame homography H_t. To obtain a stricter measure, the four image corners are projected onto the stabilized view, and the preserved ratios along the width and height are computed; the minimum of the two defines the FOV metric, capturing the worst-case preserved region.

Distortion Value (D) The distortion value measures anisotropic scaling introduced by the affine part of the homography. Specifically, for each frame t, let $\sigma_{\max}(H_t)$ and $\sigma_{\min}(H_t)$ be the singular values of the estimated homography H_t. The frame-wise distortion (higher is better) is

$$D_t = \frac{\sigma_{\min}(H_t)}{\sigma_{\max}(H_t)} \in (0, 1]. \tag{23}$$

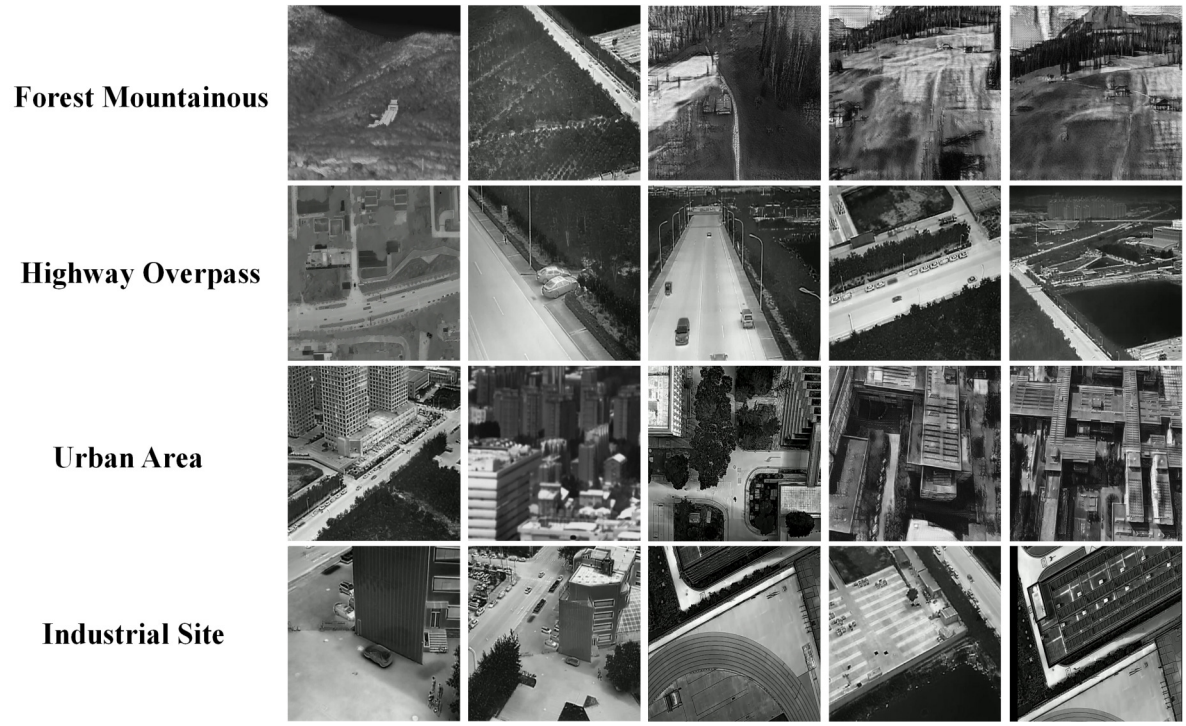

Figure 7. Representative examples of unstable UAV infrared videos from diverse scenarios, including urban areas, highways, forested mountains, and industrial sites.

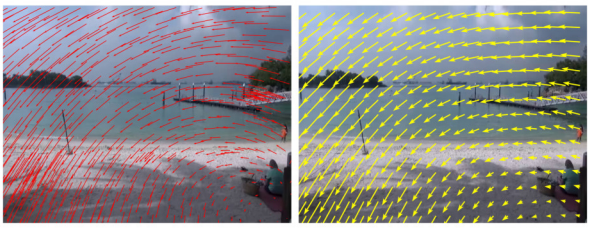

Figure 8. Visualization of motion propagation. The sparse optical flow guided by initial keypoints is contrasted with the grid-wise motion after propagation by our EfficientMotionPro module. The results show that our propagation yields more consistent and structured motion, robust to disturbances caused by dynamic objects and noise.

The global distortion metric D is defined as the minimum value across all frames, ensuring robustness against outliers:

$$D = \min_t D_t. \tag{24}$$

Stability Score (S) The stability score is defined by frequency-domain analysis of the estimated camera trajectory. The translation and rotation components are extracted as 1D temporal signals. The stability score is defined as the proportion of low-frequency energy (2nd–6th order frequencies, excluding the DC component). Formally, let $X(f)$ denote the Fourier spectrum of the trajectory signal and B_{low} the low-frequency band:

$$S = \frac{\sum_{f \in B_{\text{low}}} |X(f)|^2}{\sum_f |X(f)|^2}. \tag{25}$$

Implementation details. Trajectories are accumulated frame by frame using relative homographies, from which translation and rotation (approximated by $\arctan(s_x/s_y)$) are extracted. FFT is applied to both translation and rotation sequences. After removing the DC component, the energy ratio of the first five non-zero frequencies (2nd–6th) is computed. The final stability score is the minimum of the translation-based and rotation-based ratios, yielding a strict evaluation.

PSNR Peak Signal-to-Noise Ratio (PSNR) is used to measure reconstruction quality in videos and images:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}_I^2}{\text{MSE}}\right), \tag{26}$$

where MAX_I is the maximum possible pixel value (255 for 8-bit images). The mean squared error (MSE) between a reference frame I_i and a test frame K_i with N total pixels is

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(I_i - K_i)^2. \tag{27}$$

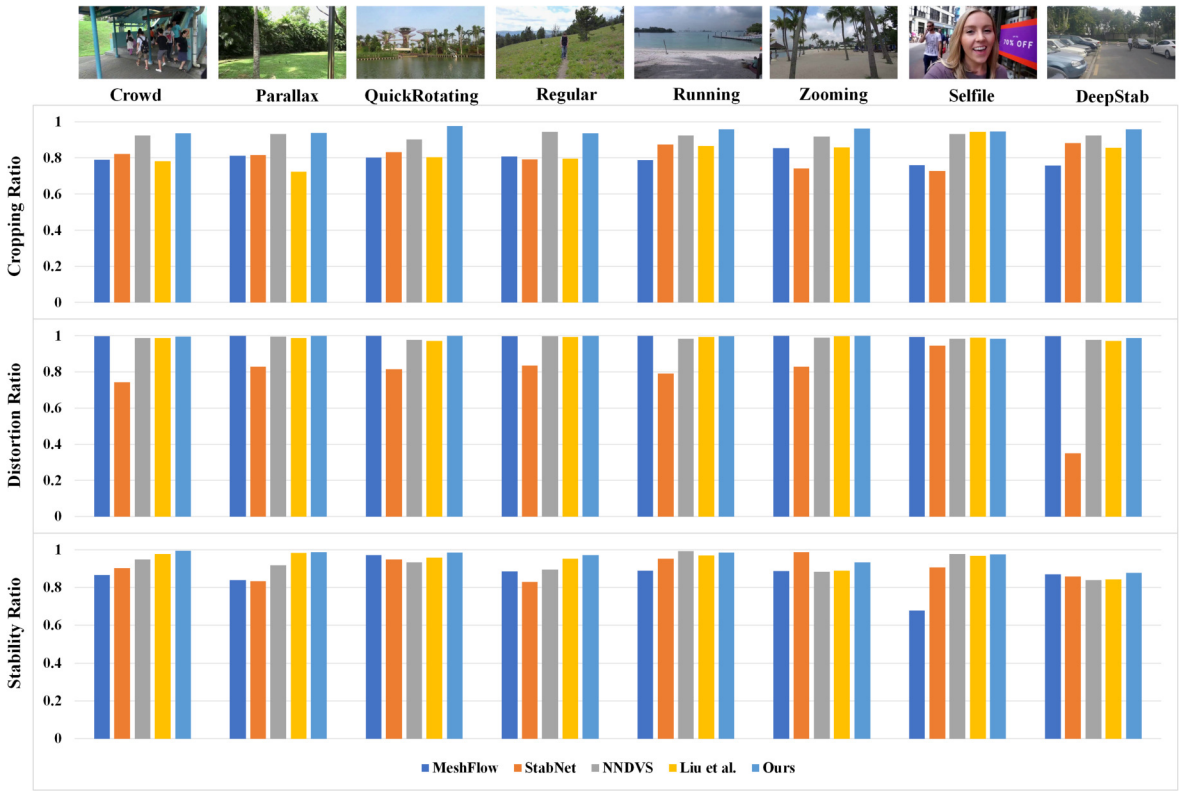

Figure 9. Per-scene quantitative evaluation.

H. Quantitative Evaluation

Fig. 9 presents the per-scene evaluation results for our method and other online approaches on the NUS [11], Selfie [27], and DeepStab [25] datasets. Both our method and representative 2D-based full-frame methods [3, 10, 30] achieve the maximum cropping ratio of 1, indicating that the stabilized videos retain the full field of view without cropping. In terms of distortion and stability, our method achieves performance comparable to the state-of-the-art 3D-based method [17], while outperforming 2D-based methods [3, 10, 13, 30]. In summary, the proposed method demonstrates both effectiveness and robustness across diverse scenarios.

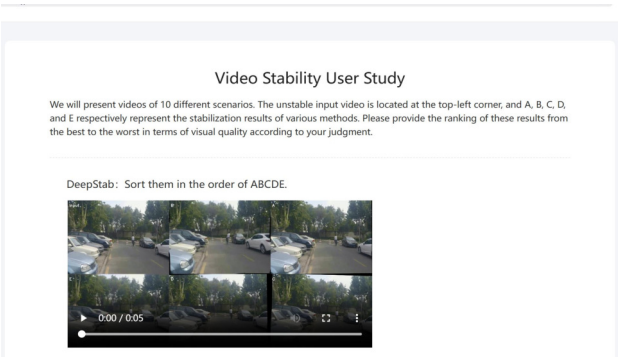

Figure 10. Example question from the user study survey.

I. User Study

We conducted a user study via an online SurveyPlus questionnaire. From the five datasets described in the paper, we randomly selected two video sequences from each, resulting in ten short video clips in total. The original unstable videos were synchronized and displayed side-by-side with stabilized outputs generated by four state-of-the-art online methods and our approach. The stabilized videos were arranged in randomized order, with each video potentially appearing in either central or peripheral positions. Participants could replay the videos to ensure a fair comparison. Finally, they were asked to rank the visual quality from best to worst, with the best video assigned 5 points, followed by 4, 3, 2, and 1 point for the worst. Figure 10 illustrates an example of our questionnaire design.

J. Additional Visualization Results

Motion Visualization Fig. 8 illustrates the sparse optical-flow motion guided by the initial keypoints and the grid-wise motion obtained after propagation with our **EfficientMo-**

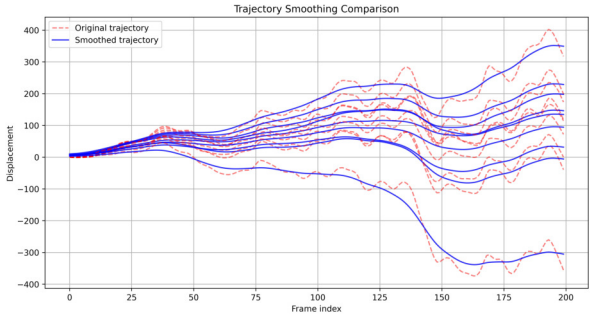

Figure 11. Visualization of mesh vertex trajectories during online smoothing. Six vertices are randomly selected, showing comparisons between unsmoothed trajectories after motion propagation and those refined by our **OnlineSmoother**.

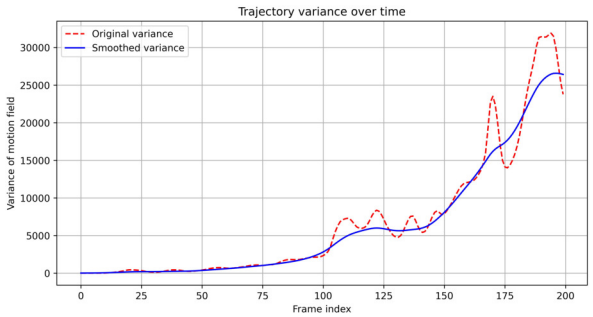

Figure 12. Comparison of trajectory variance before and after applying **OnlineSmoother**, demonstrating its effectiveness in suppressing fluctuations.

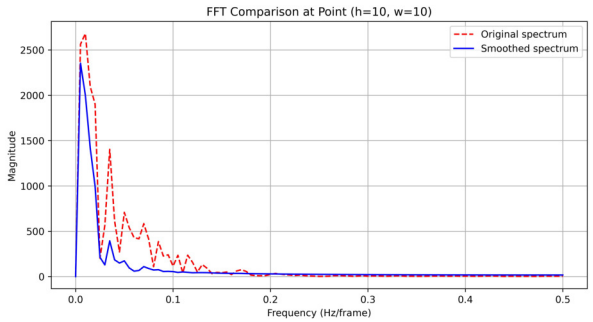

Figure 13. Frequency-domain analysis of trajectories via FFT, highlighting that **OnlineSmoother** effectively attenuates high-frequency noise.

tionPro module. After propagation, the estimated motion remains consistently structured and is less affected by dynamic objects and noise.

Trajectory Visualization As illustrated in Figs. 11, 12, and 13, we visualize mesh-vertex trajectories during online smoothing, comparing trajectories obtained after motion propagation without **OnlineSmoother** to those refined by **OnlineSmoother**. For clarity, we randomly select six mesh vertices to display their smoothed trajectories. We also plot trajectory variance and FFT spectra to further highlight the effectiveness of our OnlineSmoother module.

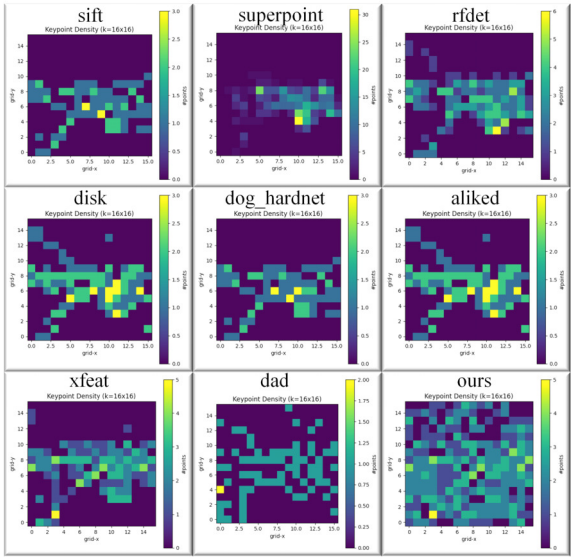

Figure 14. Keypoint density heatmaps of representative detectors (SIFT [15], SuperPoint [4], RFdet [19], DISK [23], DoG-HardNet [2], ALIKED [31], XFeat [18], DAD [6], and Ours) on a sample scene. Our collaborative detector achieves more uniform coverage across the image grid.

Keypoint Density Visualization To further illustrate the advantage of our collaborative detector, we visualize keypoint density maps of different detectors in Fig. 14. The heatmaps are computed over a 16×16 grid. As shown, traditional and learning-based detectors often produce highly clustered distributions (e.g., SIFT [15] and SuperPoint [4]), whereas our collaborative method yields more uniform spatial coverage, providing a stronger foundation for downstream geometric estimation.

Optical Flow Visualization Fig. 15 presents the results of several representative and state-of-the-art optical-flow algorithms, including [1, 5, 8, 20–22, 29]. In our experiments, we adopt MemFlow [5], which achieves high accuracy while being memory-efficient.

K. Video Results

In the supplementary material, we provide a 3-minute comparison video of our online stabilization method against other online methods. This video offers an intuitive view of our method's behavior across diverse scenarios.

Figure 15. Comparison of representative optical-flow algorithms on diverse examples. We report results from several state-of-the-art methods [1, 5, 8, 20–22, 29], among which MemFlow [5] is adopted in our experiments for its balance of accuracy and memory efficiency.

L. Limitations

Our method achieves unsupervised online video stabilization through explicit motion estimation and trajectory smoothing. However, there are two main limitations. First, it relies on existing optical-flow estimators, which may be less accurate in complex scenes. Second, to improve visual quality, we currently rely on a post-processing frame outpainting step to handle black borders; due to its computational cost, it is not integrated into the online pipeline. To address these issues, we plan to (i) explore more accurate yet efficient optical-flow models, and (ii) develop lighter, online-friendly outpainting techniques that improve edge quality without sacrificing real-time performance.

References

[1] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004. 10, 11

[2] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. Hardnet: A low memory traffic network. In *ICCV*, pages 3552–3561, 2019. 10

[3] Jinsoo Choi and In So Kweon. Deep iterative frame interpolation for full-frame video stabilization. *ACM TOG*, 39(1):1–9, 2020. 9

[4] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, pages 224–236, 2018. 10

[5] Qiaole Dong and Yanwei Fu. Memflow: Optical flow estimation and prediction with memory. In *CVPR*, pages 19068–19078, 2024. 10, 11

[6] Johan Edstedt, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. DaD: Distilled reinforcement learning for diverse keypoint detection. *arXiv preprint arXiv:2503.07347*, 2025. 10

[7] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *CVPR*, pages 1580–1589, 2020. 2

[8] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *CVPR*, pages 8981–8989, 2018. 10, 11

[9] Pierre-Marc Jodoin, Lucia Maddalena, Alfredo Petrosino, and Yi Wang. Extensive benchmark and survey of modeling methods for scene background initialization. *IEEE TIP*, 26 (11):5244–5256, 2017. 2

[10] Feng Liu, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM TOG*, 30(1): 1–10, 2011. 9

[11] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Bundled camera paths for video stabilization. *ACM TOG*, 32(4):1–10, 2013. 9

[12] Shuaicheng Liu, Lu Yuan, Ping Tan, and Jian Sun. Steadyflow:

Spatially smooth optical flow for video stabilization. In *CVPR*, pages 4209–4216, 2014. 7

[13] Shuaicheng Liu, Ping Tan, Lu Yuan, Jian Sun, and Bing Zeng. Meshflow: Minimum latency online video stabilization. In *ECCV*, pages 800–815, 2016. 6, 9

[14] Tao Liu, Gang Wan, Hongyang Bai, Xiaofang Kong, Bo Tang, and Fangyi Wang. Real-time video stabilization algorithm based on superpoint. *IEEE TIM*, 73:1–13, 2024. 6

[15] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 10

[16] Xu Ma, Xiyang Dai, Yue Bai, Yizhou Wang, and Yun Fu. Rewrite the stars. In *CVPR*, pages 5694–5703, 2024. 2, 3

[17] Zhan Peng, Xinyi Ye, Weiyue Zhao, Tianqi Liu, Huiqiang Sun, Baopu Li, and Zhiguo Cao. 3d multi-frame fusion for video stabilization. In *CVPR*, pages 7507–7516, 2024. 9

[18] Guilherme Potje, Felipe Cadar, André Araujo, Renato Martins, and Erickson R Nascimento. Xfeat: Accelerated features for lightweight image matching. In *CVPR*, pages 2682–2691, 2024. 10

[19] Xuelun Shen, Cheng Wang, Xin Li, Zenglei Yu, Jonathan Li, Chenglu Wen, Ming Cheng, and Zijian He. Rf-net: An end-to-end image matching network based on receptive field. In *CVPR*, pages 8132–8140, 2019. 10

[20] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical flow estimation. In *CVPR*, pages 1599–1610, 2023. 10, 11

[21] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *CVPR*, pages 8934–8943, 2018.

[22] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *ECCV*, pages 402–419, 2020. 10, 11

[23] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. Disk: Learning local features with policy gradient. *NeurIPS*, 33: 14254–14265, 2020. 10

[24] Ao Wang, Hui Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Lsnet: See large, focus small. In *CVPR*, pages 9718–9729, 2025. 2, 3

[25] Miao Wang, Guo-Ye Yang, Jin-Kun Lin, Song-Hai Zhang, Ariel Shamir, Shao-Ping Lu, and Shi-Min Hu. Deep online video stabilization with multi-grid warping transformation learning. *IEEE TIP*, 28(5):2283–2292, 2019. 6, 9

[26] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. In *CVPR*, pages 11534–11542, 2020. 2

[27] Jiyang Yu, Ravi Ramamoorthi, Keli Cheng, Michel Sarkis, and Ning Bi. Real-time selfie video stabilization. In *CVPR*, pages 12036–12044, 2021. 9

[28] Zhuofan Zhang, Zhen Liu, Ping Tan, Bing Zeng, and Shuaicheng Liu. Minimum latency deep online video stabilization. In *ICCV*, pages 23030–23039, 2023. 6

[29] Zhiyong Zhang, Huaizu Jiang, and Hanumant Singh. Neuflow: Real-time, high-accuracy optical flow estimation on robots using edge devices. In *IEEE IROS*, pages 5048–5055, 2024. 10, 11

[30] Minda Zhao and Qiang Ling. Pwstablenet: Learning pixel-wise warping maps for video stabilization. *IEEE TIP*, 29: 3582–3595, 2020. 9

[31] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter C. Y. Chen, Qingsong Xu, and Zhengguo Li. Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. *IEEE TIM*, 72:1–16, 2023. 10

[32] Shangchen Zhou, Chongyi Li, Kelvin C.K Chan, and Chen Change Loy. Propainter: Improving propagation and transformer for video inpainting. In *ICCV*, pages 10477–10486, 2023. 6

[33] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Heng Fan, Qinghua Hu, and Haibin Ling. Detection and tracking meet drones challenge. *IEEE TPAMI*, 44(11):7380–7399, 2021. 2

[34] Yabin Zhu, Qianwu Wang, Chenglong Li, Jin Tang, Chengjie Gu, and Zhixiang Huang. Visible–thermal multiple object tracking: Large-scale video dataset and progressive fusion approach. *PR*, 161:111330, 2025. 2