FLARE: Fisheye Lens Anchor-based Radial Enhancement for Internal Vibration Compensation in Low-cost Drones

Yassir Zardoua [1,*], Souha Seghir [1], Nouha Othman [2], Rania Othman [2], and Hadhami Garbouge [2]

[1] Caplogy, 3000 avenue Saint Louis, Fez, Morocco
[2] Caplogy, 36 Avenue de l'Europe, Vélizy-Villacoublay, France
Email: y.zardoua@caplogy.com (Y.Z.); s.seghir@caplogy.com (S.S.); n.othman@caplogy.com (N.O.); r.othman@caplogy.com (R.O.); h.garbouge@caplogy.com (H.G.)
*Corresponding author

***Abstract*—Internal micro-vibrations of fisheye lenses in low-cost drones or small Unmanned Aerial Vehicles (sUAV) are a significant problem in aerial imaging systems. With an altitude of 120 m, even a ±5-pixel shift on a 0.8 MegaPixel (MP) image may lead to more than 2 m ground displacement, affecting downstream tasks such as segmentation or object detection. There is a significant body of research addressing global vibrations, which affect the entire camera platform, e.g., sUAV. However, the problem of digital stabilization of fisheye lenses against internal jitter is under-explored, which can induce significant ground displacements, especially on low-cost fisheye cameras. Therefore, we introduce Fisheye Lens Anchor-Based Radial Enhancement (FLARE), a stabilization framework using the fisheye vignette center as a stabilization reference. Our main contributions are: (1) Gaussian Radial Mask (GRM), a spatially-varying blur technique that removes visual distractors and preserves relevant vignette boundary, and (2) Geometric Anchor Center (GAC), a coordinate prediction mechanism that anchors the stabilization reference within a reduced range, thus enabling sub-pixel stabilization accuracy. To take advantage of GAC, we formulate an anchor-aware loss that encodes the prior knowledge of the stabilization reference location. We use the predicted vignette center to estimate the vibration displacements from an arbitrary point (e.g., the image center), and subsequently stabilize the frame by applying an inverse motion through affine image transformations. We test on standardized image resolutions of 0.8 MP and report promising results, including a mean of 0.221 pixels, with 75% of center prediction errors below 0.3 pixels and 95% errors under 0.48 pixels. Additionally, we conduct ablation experiments and show that removal of GAC and GRM increases the 95-th percentile from 0.481 to 1.131 pixels.**

***Keywords*—fisheye video processing, lens stabilization, ground distance displacement, internal lens vibrations, digital video stabilization**

I. Introduction

The use of fisheye cameras in small Unmanned Aerial Vehicles (sUAV) has been largely adopted due to their high field of view, ranging from 160° up to more than 180°, allowing coverage of vast zones using a single sensor, which reduces both cost and system complexity. Nevertheless, their vulnerability to internal micro-vibrations—distinct from global body-level vibrations—remains a critical and yet neglected problem, especially for low-cost fisheye cameras that could allow a large-scale system deployment. Gurtner *et al.* [1] indicate that a one-pixel shift on a 0.8 MegaPixel (MP) image captured from a 300-meter altitude translates to a ground displacement of 2.5 m. However, such ground displacements depend not only on the sUAV altitude but also on the incident angle of the displaced ground point. These displacements will have a negative effect on downstream computer vision tasks [2–4].

The correction of these internal lens vibrations could be achieved through various approaches, such as mechanical or digital methods. Mechanical stabilizers, including gimbals and damping mounts, generally provide a better performance at the expense of more payload, cost, and sUAV design complexity [5]. For low-cost drone platforms, where budget and weight are constraints, digital stabilization represents an underexplored and yet attractive solution as it can significantly reduce the mechanical or hardware modification. Such methods rely on accurate estimation of inter-frame motion and warping techniques, which could make them promising for embedded and energy-efficient systems, given that the estimated motion is accurately computed through non-greedy algorithms [6].

Although the impact of internal lens vibrations is important on video stream quality, the current literature has been focusing on the global stabilization of the imaging system or general video correction methods. Most existing approaches process the entire camera jitter, either through a mechanical platform or digital processing pipelines, whereas the specific issue of internal jitter on low-cost fisheye cameras is almost unexplored. This gap requires specialized compensation strategies that take into

Manuscript received October 3, 2025; revised December 8, 2025; accepted January 9, 2026; published July 16, 2026.

account the unique properties of fisheye images [7]. Vibration compensation serves, in general, as a preprocessing step to aid subsequent computer vision tasks. Therefore, the stabilization methodology must be fast and resource-efficient. This requirement is crucial in sUAV and drone applications to allow low-cost deployment with a maximum flight duration. Additionally, given the micro-scale of vibrations induced by the drone's motor, the stabilization needs to meet sub-pixel stabilization accuracy [8].

Aerial platforms are becoming essential parts of next generation wireless infrastructure, making this work relevant to emerging 6G and Internet of Things (IoT) contexts. With improved real-time data processing, secure communication, and autonomous decision-making capabilities, recent advancements show that 6G-enabled drone systems are transforming surveillance and monitoring applications [9]. The commercial drone market is experiencing rapid growth, serving vital industries such as public safety, agriculture, and surveillance [10]. Because of their capacity to deliver real-time data and wide-area coverage in vital areas like public safety, agriculture, and surveillance, drones are becoming an increasingly important part of IoT and 6G ecosystems. Reliable image quality and stable vision-based performance will be crucial for dependable operation in bandwidth-intensive, latency-sensitive scenarios.

In light of this, we present Fisheye Lens Anchor-Based Radial Enhancement (FLARE), a stabilization framework intended to lessen the effect of internal micro vibrations on the quality of aerial imaging. The framework expands upon two main contributions:

- Gaussian Radial Masking (GRM): A spatially-varying blur technique that suppresses vignette-like distractors while preserving boundary information, reducing visual noise without introducing artifacts.
- Geometric Anchor Constraint (GAC): A coordinate prediction mechanism that restricts the search space to plausible displacement ranges through constrained activation functions and anchor-aware loss formulation, enabling enhanced sub-pixel stabilization accuracy.

Prior research explaining the practical effect of vibrations on ground displacements seems to be experimental without a clear explanation of the link between small internal vibrations and ground shifts. To the best of our knowledge, no previous work has explicitly formulated this link using clear mathematical equations. Therefore, we present this link as a side contribution in Section III and establish a mathematical framework computing the effect of internal micro-vibrations in fisheye lenses on ground displacements.

We organize the rest of this paper as follows: Section II presents a literature review, followed by Section III, which introduces a theoretical framework for quantifying ground displacement in meters corresponding to internal lens vibrations. Section IV details the proposed methodology (the FLARE pipeline), namely the GRM and the GAC mechanism. Before presenting the experimental results in Section VI, we provide a comprehensive explanation of the experimental setup in Section V. We then clarify the scalability of our method on embedded platforms in Section VII, provide further discussion of the results in Section VIII, and conclude in Section IX.

II. Literature Review

Digital video stabilization approaches aiming to correct fisheye lens vibrations in sUAV and drone applications are largely under explored in the literature. Recent exhaustive surveys of UAV stabilizers confirm that the available literature about sUAV vibration and image quality is very limited [11]. The same survey states that the problem of vibration is more noticeable on the sUAV, because of its small size, making it more susceptible to maneuvers and turbulence typical in small drone operations, suggesting that this literature gap is an important research area [11]. While these comprehensive surveys provide a deep coverage of mechanical and digital stabilizers in general, they lack any significant discussion or analysis of internal lens vibrations and their effect on subsequent tasks. This gap is even more notable for tailored digital solutions of this problem. We claim that accurate fisheye center estimation for lens stabilization remains virtually unexplored, with only one notable early method attempting to address it on real sUAV footage [1]. To understand this critical research gap, we review both this pioneering approach and the broader landscape of stabilization techniques.

Gurtner *et al*. [1] first attempt that explicitly aims at correcting micro-vibrations in fisheye lenses using the vignette center as an alignment reference. They use the Canny edge detector associated with a Hough circle detector to estimate the fisheye center, hence streamlining the stabilization by aligning consecutive frames to the detected center. This approach, however, relies on classical image processing technologies, and the main concern in using classical edge detectors is the blurring or illumination effect on the vignette boundary, leading to missing valuable edges and inaccurate estimation of the reference center. In such conditions, the authors report a significant precision drop from 80% down to 50%.

Similarly, a sea-skyline based stabilization technique for omnidirectional (catadioptric) cameras installed on marine buoys was proposed by Cai *et al.* [12]. They find the horizon boundary, which appears as an ellipse in omnidirectional images under camera tilt, using adaptive Canny edge detection with Hough ellipse fitting as a stabilization reference. However, the necessary edges for precise Hough ellipse fitting are weakened by blurring and dim lighting. Furthermore, there is a fundamental trade-off between resolution and accuracy for edge-based methods. While processing at high resolution (e.g., 1024×1024 at 182.3 ms) achieves better accuracy but sacrifices real-time capability, processing at low resolution (e.g., 256×256 at 12.1 ms) for real-time performance introduces scaling errors when projecting geometric parameters to high-resolution frames without special scaling methods (e.g., naive linear projection). This issue has been addressed through interpolation and gradient features applied on a low-resolution edge map [13]. Our FLARE,

on the other hand, completely avoids this trade-off by employing learned regression on 224×224 inputs, where the predicted coordinates naturally project to the original resolution without accumulated geometric errors.

Contemporary digital stabilizers have primarily focused on general video correction, without taking the specific geometrical features of fisheye lenses or the digital images they produce. Optical-flow methods demonstrated good performance in drones and UAV applications, with Lim *et al.* [6] reaching 50 Frames Per Second (FPS) thanks to the Lucas-Kanade flow estimation. Advanced techniques in trajectory optimization offer superior stabilization through simultaneous multi-trajectory optimization, but require offline processing, which is incompatible with sUAV requirements [14]. Deep learning approaches have enabled significant performance gains, including a 30× acceleration through Convolutional Neural Network (CNN)-based warping, showing a high robustness against complex movements by learning progressive mesh-grid transformation for each input image based on previous stabilized frames [8].

TABLE I. COMPARISON OF FLARE WITH TRADITIONAL STABILIZATION TECHNIQUES

Approach	Key Features	Limitations
Edge detection [1, 12]	• First fisheye-specific stabilization using vignette center as alignment reference via Canny edge and Hough circle detection [1]; • Sea-skyline horizon detection for omnidirectional cameras using adaptive Canny edge with Hough ellipse fitting [12].	• Significant precision drop under blurring or poor illumination [1]; • Blurred edges hinder Hough transformation accuracy [12]; • Resolution-accuracy trade-off: low-resolution input (higher FPS) introduces scaling errors; high-resolution input sacrifices real-time performance.
Optical flow [6, 15]	• Real-time performance (50 FPS with Lucas-Kanade); • Effective for general camera motion stabilization.	• Not designed for fisheye-specific lenses; • Struggles to distinguish internal lens vibrations from global camera motion; • Offline trajectory optimization methods [14] incompatible with real-time sUAV requirements.
Hybrid gyroscope [17, 18]	• Integrate gyroscopic data with visual correspondences (SIFT-RANSAC [17]) or electronic gyrometric signals [18].	• Require additional motion sensors (gyroscopes); • Address global camera motion without fisheye-specific lens properties; • Originally developed for mobile phones, not tailored for internal lens vibrations.
Hybrid mechanical-electronic-software [5, 22, 23]	• Active camera mounts using piezo stack actuators with sliding mode controllers [5]; • SMA actuators with low-power consumption [22]; • Passive dampers achieving 97.1% high-frequency vibration reduction [23].	• Requires hardware integration with actuators and controllers; • Adds payload weight and power consumption [5]; • Expensive precision electronics and complex hardware tuning with tailored control algorithms [22]; • Manual tuning of damper parameters (size, angle, placement) needed; • Amplifies low-frequency vibrations (~50 Hz); mitigated through additional algorithmic filtering; • Designed for onboard sensor stabilization, not camera or lens stabilization [23]; • Not tested on fisheye videos [22].
Deep learning [8, 24]	• Fast processing (30× acceleration) through CNN-based mesh-grid warping [8]; • Robust to complex camera movements via learned transformations [8]; • Self-supervised training achieving 41 FPS on desktop GPU [24].	• Designed for general videos without fisheye geometry consideration; • Requires temporal frame history (not single-frame) [8]; • Heavy architectures (U-Net + optical flow) unsuitable for embedded deployment [24]; • No method addresses fisheye center detection with sub-pixel accuracy.
FLARE (ours)	• Payload: pure software with reasonable computations and multi-task model integration supports lightweight drone development; • Speed & memory: shown to work on embedded CNN model backbone with only FP16 precision (with significant future enhancement, e.g., lighter backbone, pruning or further quantization); • Resolution-accuracy trade-off: FLARE is shown to infer the center on higher output resolution than given input through naturally learned prediction scaling; • Fisheye-tailored modules: GRM suppresses fisheye-specific distractors and enhances stabilization accuracy; GAC exploits prior reference location information during training (through a customized loss function) and inference; • Robust to blurred edges: the natural high receptive field of CNN kernels enables learning accurate center detection of blurred vignettes.	Limitations of FLARE will be discussed in Section VIII.

Recent fisheye image processing papers focus more on calibration and distortion correction rather than stabilization. For instance, Wei *et al.* [7] proposed a video correction method aiming at progressive reduction of fisheye distortions using mesh-grid warping, with the latter being guided through multiple criteria, namely line straightness, shape preservation, and temporal coherence. This requires, however, a manual intervention for each video, for example, by marking the straight lines or object zones in key frames. Regensky *et al.* [15] introduced a motion compensation framework, dividing the fisheye image into multiple viewports (perspective projections) to better model the displacements to compensate on different projection planes, allowing a much more precise alignment of movements. However, it operates on rectified coordinates and does not address internal lens vibration or dynamic center estimation. Recent calibration methods increasingly rely on learning-based technologies. In this context, Yang *et al.* [16] contributed dual diffusion models to correct real image distortions, with the main issue being the lack of undistorted image labels and the low capacity of synthetically generated labeled images to reliably reproduce the real-world distortions of fisheye images, making model generalization extremely poor.

In urban scenarios, in accordance with the Manhattan world hypothesis, most of urbane units, such as buildings and roads, get oriented along three orthogonal directions: vertical, left-right horizontal, and forward-backward horizontal. This geometric regularity can be leveraged by predicting three structured key points representing these three dominant directions. Thanks to that, the model is able to estimate the camera's orientation and correct fisheye distortions even in the absence of classical visual cues, such as the user-defined cues are scarce [7].

Hybrid approaches show a strong potential for drones and sUAV stabilization. Karpenko *et al.* [17] have been pioneers in the integration of gyroscopic data with SIFTRANSAC visual correspondences in the calibration step. While this method was initially developed for mobile phones, it shows promise if adapted to the strong fisheye distortions. Another relevant contribution is the patent by Pierre, which processes video streams affected by motor-induced internal vibrations [18]. This patent relies on electronic mechanisms driven by rapid and precise gyrometric signals.

Computational constraints of drones and sUAVs have favored the development of efficient deep learning architectures. Adaptations of ERFNet reach a fisheye segmentation speed of 83 FPS on a desktop computer and 7 FPS on embedded targets [19]. On the other hand, MobileNet variants have achieved a significant reduction in the model parameters while maintaining a practical precision for mobile deployment [20, 21]. The performance of these architectures, however, remains questionable in high-precision detection of circular vignette centers.

With different trade-offs, alternative mechanical stabilization technologies have been investigated for drone applications. A low-power substitute for piezoelectric systems, Shape Memory Alloy (SMA) actuators provide efficient image stabilization while using less energy [22]. Though they may amplify low-frequency components and require careful tuning, passive damping solutions have also shown impressive performance, with optimized damper configurations achieving up to 97.1% reduction in high-frequency vibrations [23]. Self-supervised techniques have become viable alternatives in the field of learning-based methods. They have shown promise in video stabilization tasks by achieving real-time performance of 41 FPS on desktop Graphics Processing Units (GPUs) without the need for labeled training data [24]. We summarize in Table I the key features of FLARE compared to prior stabilization techniques.

III. Quantifying Ground Displacement from Internal Optical Vibrations

Camera vibrations have long been recognized as a major source of error in aerial image processing systems [25]. In the case of wide-angle imagery, such as fisheye imaging systems, these effects are more significant; we will show that small displacements on the sUAV translate into nonlinear, exponentially exploding ground errors. Table II summarizes the mathematical notation used throughout this section.

TABLE II. Mathematical Notation Used in Section III

Symbol	Description
h	Drone altitude (height above ground)
d	Ground distance from nadir
θ	Incident angle (in radians)
r	Radial distance from image center to projected point
F	Focal length
Δp	Pixel shift on the image plane
Δr	Radial shift on the imaging sensor
p_{size}	Physical pixel size
$\Delta\theta$	Angular error
Δd	Ground distance error

Gurtner *et al.* [1] have shown some experimental effects on the ground, but the incident angle, which has a strong effect on the ground error, is overlooked. In our framework, we assume that vibrations only affect the optical system (lens assembly), with the relative movement between camera and different ground points being constant. Consider a fisheye camera mounted on a drone flying at height h above a planar surface, observing a ground point at distance d from the nadir (see Fig. 1).

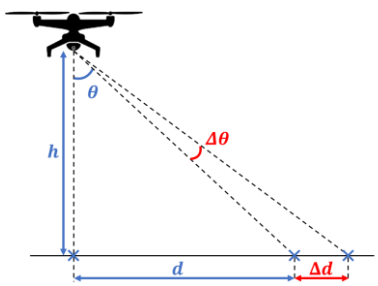


Fig. 1. Relationship between vibration amplitude and center displacement

The incident angle θ relates to ground distance through Eq. (1).

$$d = h \tan\theta \tag{1}$$

Using the equidistant fisheye projection model, the radian distance r from the image center to a projected point is given by Eq. (2):

$$r = f\theta \tag{2}$$

where f being the focal length and θ, the incident angle shown in Fig. 1, is in radians.

Internal vibrations manifest as consecutive minor pixel shifts on the image and sensor plane. We denote a particular pixel shift on the image as Δp. Δp is equivalent to a radical shift on the imaging sensor, which we denote as Δr, and express in Eq. (3):

$$\Delta r = \Delta p \cdot psize \tag{3}$$

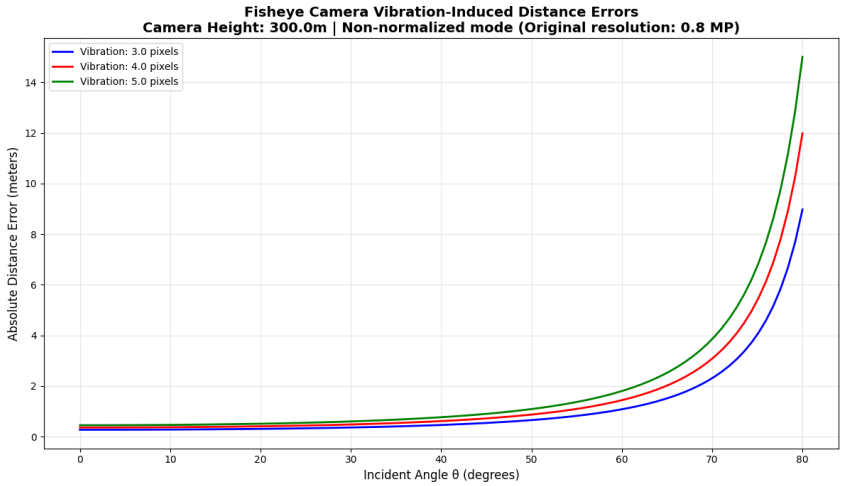
where p_{size} is the physical pixel size.

This sensor shift produces an angular error $\Delta\theta$, resulting in a ground distance error Δd. Eqs. (4) and (5) formulate these two quantities, which are also illustrated visually in Fig. 2.

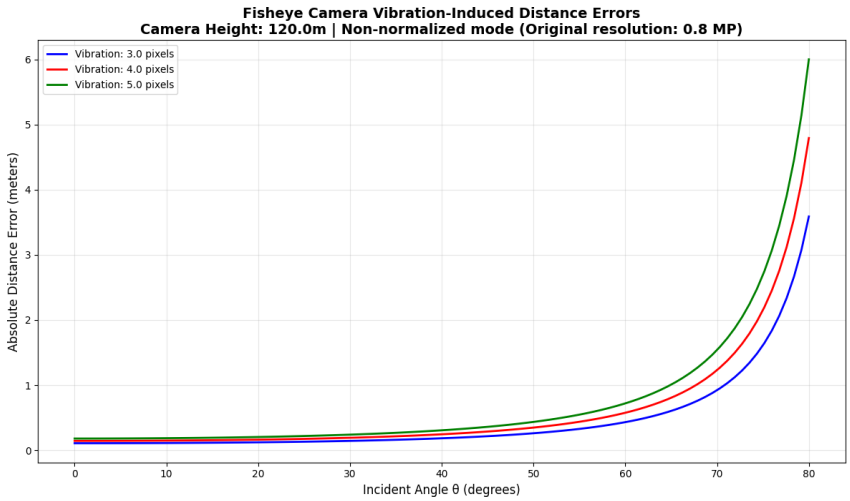
$$\Delta\theta = \frac{\Delta r}{f} \tag{4}$$

$$\Delta d = h\left[\tan(\theta + \Delta\theta) - \tan(\theta)\right] \tag{5}$$

Clearly, this error shows a strong dependence of, not only the drone's altitude h, but also the incident angle θ. According to real measurements performed, vibrations of fisheye lenses may reach a maximum pixel shift of 5 pixels on a 0.8 MP reference image [1]. Based on such an experimental observation, we show in Fig. 2(a) the ground shifts with respect to the incident angle, given three specific vibration ranges: 5, 4, and 3 pixels. While Gurtner *et al*. [1] mention an altitude of 300 m (1000 feet), our research aims at stabilizing vibrations onboard low-cost drones or sUAVs, where typical altitudes range from 10 to 85 m, with 120 m as the maximum legal altitude in many countries, such as France. The errors corresponding to the maximum legal altitude are plotted in Fig. 2(b), revealing that eventual ground displacements could reach 1.5 m at θ = 70°, and even surpassing the 3–6 m threshold past θ = 70°.



(a)



(b)

Fig. 2. Absolute distance error versus incident angle in fish-eye cameras under varying vibration levels (±3.0, ±4.0, and ±5.0 pixels). (a) Ground errors at 300 m altitude. (b) Ground errors at 120 m altitude.

IV. Methodology

Our method's pipeline is illustrated in Fig. 3. Before delving into a detailed explanation of each stage, we provide a brief overview of the overall processing workflow.

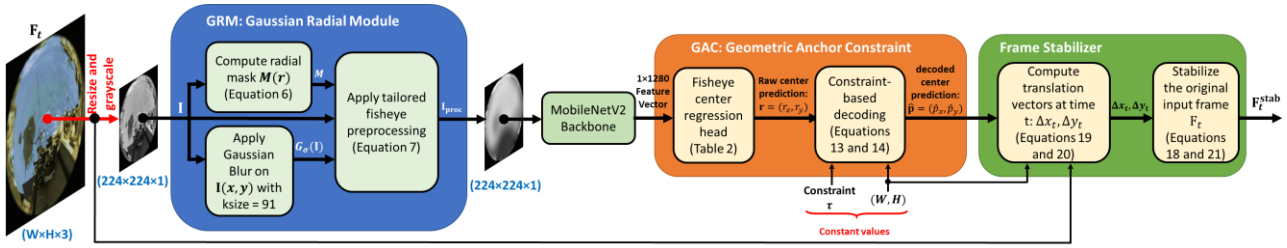


Fig. 3. FLARE pipeline overview.

Given an input frame F_t at time t, we apply image resizing and grayscale transformation, thus producing image I of shape (224×224) [1]. The GRM module preprocesses the frame I to produce I_{proc}, effectively eliminating semantic distractors while preserving vignette boundary features. The preprocessed image I_{proc} is then fed to a CNN backbone for feature extraction, and the (1×1280) feature vector is sent to a center prediction head, which outputs $r = [r_x, r_y] \in [-1, 1]$ representing the vignette center. Subsequently, the GAC module decodes r into predicted pixel coordinates $\left(\hat{p}_{x,t}, \hat{p}_{y,t}\right)$ on the original input frame F_t. Finally, the Stabilization module aligns the detected vignette center with the frame center by applying an affine transformation to F_t, producing the stabilized output F^{stab}.

A. GRM Preprocessing

A core component of our methodology is a tailored pre-processing aiming at reducing the semantic distractors while preserving information related to vignette edges. The GRM block in Fig. 3 takes a grayscale input I at resolution (224×224) and generates both a radial mask $M(r)$ and a Gaussian-blurred version $G_\sigma(I)$, which are then combined with the original input I through Eq. (7) to produce I_{proc}. The radial mask $M(r)$ modulates blur intensity along the radial direction (i.e., the direction normal to a virtual circle centered at the image center), varying from maximum blur intensity at the image center to zero at the image periphery, as shown in Fig. 4 and expressed in Eq. (6):

$$M(r) = \begin{cases} 1.0 & \text{if } r \le r_{start} \\ 1.0 - \dfrac{r - r_{start}}{r_{end} - r_{start}} & \text{if } r_{start} < r \le r_{end} \\ 0.0 & \text{if } r > r_{end} \end{cases} \quad (6)$$

where $r = \sqrt{(x - c_x)^2 + (y - c_y)^2}$ is the distance from the center (c_x, c_y), $r_{start} = r_{start}^{\%} \cdot d_{half}$ defining the entire blur zone radius, and $r_{end} = r_{end}^{\%} \cdot d_{half}$ defines the transition boundary, where $r_{start}^{\%}$ and $r_{end}^{\%}$ represent scalar ratios within the range [0, 1], with $d_{half} = \frac{\min(W,H)}{2}$ corresponding to half the smaller image dimension.

In our case, $W = H$, but when $H \neq W$, the minimum operation enables visual interpretability of the blurring radius proportion, as it simplifies comparison between min(W, H) and the fisheye vignette radius. The preprocessed image I_{proc} is computed according to Eq. (7):

$$I_{proc}(x, y) = M(r) \cdot G_\sigma(I)(x, y) + (1 - M(r)) \cdot I(x, y) \quad (7)$$

where $G_\sigma(I)$ represents the blurred image using a big kernel of size k and a standard deviation σ governed by k.

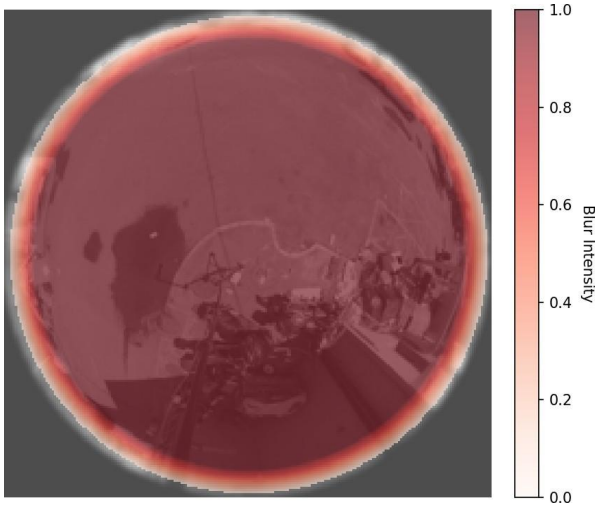


Fig. 4. GRM's blurring mask $M(r)$ applied to a fisheye image to suppress semantic distractors while preserving vignette boundaries. The visualization corresponds to the preprocessing pipeline defined in Eqs. (7) and (6).

The approach presented in Eq. (7) allows effective suppression of distractors without affecting relevant features or introducing new artifacts, which is achieved thanks to the radial mask defined in Eq. (6), as the blurring gets closer to relevant features, i.e., vignette boundary, the blurring effect is attenuated. This attenuation is gradual and smooth, thus preventing the introduction of abrupt

[1] We can also denote this image as I_t to specify the t-th frame. Throughout this paper, the subscript t is optional and simply refers to the t-th frame.

intensity changesin the preprocessed image, thus avoiding the introduction ofharmful artifacts. We provide a detailed visual illustration of that in Fig. 4.

B. CNN Architecture and Output Prediction

Classical edge detection methods, including the Sobel-Canny combination, present various limitations in the detection of blurred vignette contours. Their high dependence on local gradients and sensitivity to artifacts make them less reliable. Blurred vignette effects, combined with the illumination changes, such as black shadows affecting the transition to black vignette color, pose significant issues and require better methods. CNNs, by structuring small kernels into multiple layers, provide a wider receptive field and much more robust feature extractions. Even though CNNs typically rely on small kernel sizes (3×3), their composition into multiple layers enables large receptive fields, thus capturing complex features exceeding the capacity of any single handcrafted kernel.

We use an ImageNet-1K pretrained MobileNetV2 as the main feature extractor, selected for its balance between computational speed and feature robustness [21]. Table III concisely describes the complete architecture, with the output layer composed of two scalars referring to normalized prediction of the vignette center coordinates, denoted as $r = [r_x, r_y]^T$. The normalized range of r will be mapped onto Cartesian coordinates of the original image using the GAC block, which we describe in Section IV.C.

TABLE III. MODIFIED MOBILENETV2 ARCHITECTURE FOR VIGNETTE CENTER DETECTION

Architecture Stage	Input	Operator	*t*	*c*	*n*	*s*
Mobilenetv2 backbone	$224^2 \times 3$	conv2d	-	32	1	2
	$112^2 \times 32$	bottleneck	1	16	1	1
	$112^2 \times 16$	bottleneck	6	24	2	2
	$56^2 \times 24$	bottleneck	6	32	3	2
	$28^2 \times 32$	bottleneck	6	64	4	2
	$14^2 \times 64$	bottleneck	6	96	3	1
	$14^2 \times 96$	bottleneck	6	160	3	2
	$7^2 \times 160$	bottleneck	6	320	1	1
	$7^2 \times 320$	conv2d 1×1	-	1280	1	1
Fisheye Center Regression Head	1×1280	Linear + ReLU + Dropout	-	512	1	-
	1×512	Linear + ReLU + Dropout	-	128	1	-
	1×128	Linear + Tanh	-	2	1	-

The upper section details the Mobilenetv2 backbone (original head removed), lower section describes our custom regression head for 2D coordinate prediction, with each row defined by: *t* (expansion factor applied to the input channels), *c* (number of output channels), *n* (number of operator repetitions), and *s* (effective operation stride).

C. GAC: A Constrained Coordinate Prediction Approach

We experimentally found that unconstrained coordinate regression of the vignette center leads to significant errors, specifically when mapping the model prediction to the original image vignette center (x_c, y_c). Such free predictions lead to exploring the entire image space instead of the relevant regions. The GAC block (Fig. 3) decodes the normalized prediction r into pixel coordinates $\left(\hat{p}_{x,t}, \hat{p}_{y,t}\right)$ using the fixed image dimensions (W, H) and the tolerance constant τ, which remain constant during both training and inference. We increase the model's prediction precision by incorporating simple prior knowledge of the vibrations' range through τ, specifically by constraining the mapping range of (x_c, y_c), as shown in Fig. 5.

The range restriction in Fig. 5 concentrates the representational power capacity of the model, i.e., its capacity to distinguish different coordinate positions, into a bounded search space. Consider the model output $r = [r_x, r_y]^T \in \mathbb{R}^2$ with a floating-point precision of b bits. We theoretically analyze the quantification resolution of the coordinate prediction along a single spatial dimension, such as x, given that we work with square fisheye images ($W = H$) and that both axes, x and y, share the same range constraint.

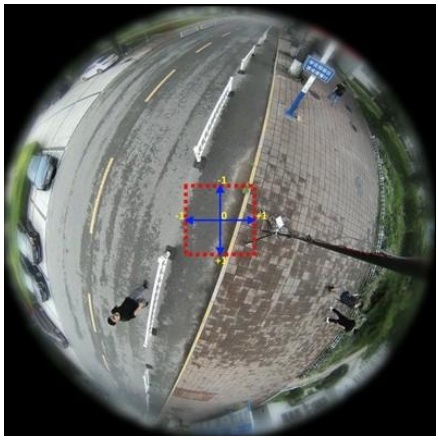


Fig. 5. GAC: Constraining the output tanh activation range within a smaller area.

Examining the x-coordinate case for unconstrained prediction mapping to the full image space, Eqs. (8) and (9) describe an unconstrained mapping and x-coordinate resolution, respectively:

$$r \in [-1,1]^2 \rightarrow p \in [c_x \pm \tau W, c_y \pm \tau H] \tag{8}$$

$$R_{u,x} = \frac{W}{2^b} \tag{9}$$

where $p = [p_x, p_y]$ denotes the decoded pixel coordinates, and the symbol → indicates the mapping from the model's output to the reference image $I(x, y)$ with spatial dimensions (W, H). We note that the range [−1, 1] in Eq. (8) results from the tanh output activation. The constrained approach we implement restricts the search space to a tolerance-bounded region centered around the image center $(c_x, c_y) = (W/2, H/2)$. Accordingly, the revised mapping and x-coordinate resolution are expressed in Eqs. (10) and (11), respectively:

$$r \in [-1,1]^2 \rightarrow p \in [c_x \pm \tau W, c_y \pm \tau H] \tag{10}$$

$$R_{c,x} = \frac{2\tau W}{2^b} \tag{11}$$

where τ = 0.0078 represents the tolerance parameter defining the maximum expected displacement from the image center.

The maximum theoretical precision enhancement factor for the x-coordinate is defined in Eq. (12):

$$\xi_x = \frac{R_{u,x}}{R_{c,x}} = \frac{W/2^b}{2\tau W/2^b} = \frac{1}{2\tau} \approx 64.1 \tag{12}$$

Identical analysis applies to the y coordinate, leading to $\xi_y = \frac{1}{2\tau} \approx 64.1$. This 64-fold precision enhancement translates to a sub-pixel localization accuracy, given that the network allocates its entire representational capacity within narrower and relevant regions. The tolerance constraint τ in GAC encodes domain-specific knowledge, similar to the anchor box mechanisms in early You Only Look Once (YOLO) models [26–28], with its value derived from real-world vibration measurements conducted by Gurtner *et al.* [1]. The choice of activation function critically determines the coordinate decoding mechanism and subsequent precision characteristics. For the tanh activation function producing outputs $r \in [-1, 1]^2$, the decoding equations are given in Eqs. (13) and (14):

$$p_x = W \cdot \left(\frac{1}{2} + r_x \cdot \tau \right) \tag{13}$$

$$p_y = H \cdot \left(\frac{1}{2} + r_y \cdot \tau \right) \tag{14}$$

where (p_x, p_y) representing the decoded pixel coordinates in the original image. We confirm experimentally that the tanh activation function presents superior prediction characteristics when compared to the sigmoid function. The output range [−1, 1] is symmetric, providing a natural alignment with the bidirectional vibrations that could be both positive and negative when measured with respect to the image center. Another property that makes tanh more suitable is its higher gradient value close to the origin, with $\left(\frac{d \tanh(x)}{dx}\Big|_{x=0} = 1 \right)$ compared to $\left(\frac{d\sigma(x)}{dx}\Big|_{x=0} = 0.25 \right)$, thus aiding the convergence of the model and better leverage of GAC.

D. Loss Function Design

We employed a square root loss since it has a high derivative at small errors, enabling a higher emphasis on small prediction errors. We compute two loss functions: the raw pixel loss $\mathcal{L}_{\text{pix}}$, used for visualization and interpretation on a reference image of 0.8 MP [1], and the normalized loss $\mathcal{L}_{\text{norm}}$, used during backpropagation to ensure invariance with respect to input image size. We provide $\mathcal{L}_{\text{pix}}$ in Eq. (15), which is simply the raw Euclidean distance on a reference image I_{ref} of configurable width and height (W_{ref}, H_{ref}).

The term $\|p - t\|_2^2$ in Eq. (15) is detailed in Eq. (16).

$$\mathcal{L}_{\text{pix}} = \sqrt{\| p - t \|_2^2} \tag{15}$$

$$\begin{aligned} p - t &= \begin{bmatrix} p_x \\ p_y \end{bmatrix} - \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} p_x - t_x \\ p_y - t_y \end{bmatrix} \\ \|p - t\|_2^2 &= (p_x - t_x)^2 + (p_y - t_y)^2 \end{aligned} \tag{16}$$

where t denotes the ground truth coordinates in pixel space for the reference image I_{ref}, and p denotes the decoded predictions using the coordinate decoding procedure described in Section IV C, in which the width and height parameters are W_{ref} and H_{ref}, respectively. We express the actual loss used in backpropagation in Eq. (17):

$$\mathcal{L}_{\text{norm}} = \lambda_{\text{pen}} \sqrt{\frac{\mathcal{L}_{\text{pix}}}{\tau \sqrt{W_{\text{ref}} H_{\text{ref}}}} + \epsilon} \tag{17}$$

where the expression $\tau\sqrt{W_{ref} H_{ref}}$ refers to the maximal pixel error on the reference image I_{ref}, thus normalizing the pixel loss $\mathcal{L}_{pix}$ within the interval [0, 1]. The scalar λ_{pen} is a penalty term, and ε introduces a small positive nudge to enable numerical stability. The normalized loss $\mathcal{L}_{\text{norm}}$ includes a square root operator, which increases the loss sensitivity to small errors, a crucial adaptation given the high precision requirement for compensating video jitter.

E. Video Stabilization through Vignette Center Alignment

The final Stabilization block (Fig. 3) receives both the original frame F_t and the predicted vignette center coordinates $(\hat{p}_{x,t}, \hat{p}_{y,t})$ from GAC, then computes the translation matrix T_t to produce the stabilized output F_t^{stab}. Given an input frame F_t at instant t, we predict the coordinates of the vignette center $(\hat{p}_{x,t}, \hat{p}_{y,t})$ using the decoding procedure we previously described in Eqs. (13) and (14), then we calculate the translation vectors that would align the vignette center to the image center according to Eq. (18):

$$T_t = \begin{bmatrix} 1 & 0 & \Delta x_t \\ 0 & 1 & \Delta y_t \\ 0 & 0 & 1 \end{bmatrix} \tag{18}$$

where: $\Delta x_t = \frac{W}{2} - \hat{p}_{x,t}$, $\Delta y_t = \frac{H}{2} - \hat{p}_{y,t}$.

We denote the stabilized frame at instant t by F_t^{stab}, and compute it through an affine transformation using Eq. (19):

$$F_t^{stab} = T_t \cdot F_t \tag{19}$$

Stabilizing a video stream repeats the same process described at instant t to all subsequent frames.

V. Experimental Setup

A. Dataset Description

1) Dataset structure

The experimental dataset comprises fisheye video sequences sourced from the Fisheye8k dataset [29], originally captured with stationary fisheye cameras. To address the absence of annotated sUAV vibration data, synthetic vibrations are applied to simulate realistic lens micro-jitter effects commonly observed during aerial operations, as summarized in Table IV.

Ground truth annotations are provided as frame-level coordinate labels stored in Comma-Separated Value (CSV) format. Each annotation file corresponds to a single video sequence and contains frame indices paired with precise vignette center coordinates (x_c, y_c) in pixel space. To facilitate model training and evaluation across varying image resolutions, the annotation pipeline automatically generates normalized coordinate representations, where vignette centers are expressed as fractional positions $(\hat{x}_c, \hat{y}_c) \in [0,1]^2$ relative to image dimensions.

TABLE IV. Simulation Parameters for the Synthetic Fisheye Vibration Dataset, Based on Real-World Vibration Footage in Ref. [1]

Parameter	Value	Description
Vibration Type	Translational	Random pixel shifts along the x and y axes.
Magnitude Range	±5 pixels	Maximum displacement applied to the image plane
Axes of Motion	x, y	Axes along which the image is translated, with displacement values constrained by the magnitude range, to simulate vibrations.

2) Data balancing strategy

Inter-video frame count disparities necessitate explicit data balancing to prevent model bias toward overrepresented sequences during training. Preliminary analysis revealed significant variation in video sequence lengths, with frame counts ranging from hundreds to several thousand frames per video. Without intervention, this imbalance would lead to disproportionate influence from longer sequences during model training. The balancing strategy employs uniform frame subsampling constrained by the minimum sequence length across the training partition. Specifically, all training videos are limited to $N_{\min}$ frames, where $N_{\min}$ represents the length of the shortest sequence in the dataset. Rather than selecting consecutive initial frames, which would introduce temporal bias toward video beginnings, frames are sampled uniformly across the entire temporal span of each sequence. For a video sequence of length N_{orig}, the sampling indices are computed as Eq. (20):

$$\mathcal{I}_{\text{sample}} = \left\{ \left\lfloor i \cdot \left(\frac{N_{\text{orig}}-1}{N_{\min}-1} \right) \right\rfloor : i \in \{0,1,\ldots,N_{\min}-1\} \right\} \tag{20}$$

where the term $\left(\frac{N_{\text{orig}}-1}{N_{\min}-1}\right)$ denotes the frame-wise increment, and $\lfloor . \rfloor$ indicates the floor operation, ensuring integer-valued indices.

3) Validation set sampling

To ensure computational efficiency during iterative model evaluation while maintaining statistical validity, the validation set employs fixed deterministic subsampling. Each validation video contributes exactly K frames, where $K = 64$ is chosen as a multiple of the training batch size to optimize GPU memory utilization and eliminate partial batch effects during validation.

The sampling methodology mirrors the training balancing approach, utilizing the uniform temporal distribution described in Eq. (20), with deterministic seed initialization to guarantee reproducible validation metrics across experimental sessions. The validation set does not contain any videos shorter than K frames. However, if such a case arises, all available frames are retained without subsampling.

To ensure robust generalization evaluation, validation videos are sourced from completely different scene types compared to the training set. Specifically, training data consists exclusively of outdoor street scenes, while validation sequences comprise indoor environments. This deliberate scene separation prevents overfitting to specific visual content and provides reliable assessment of the model's ability to detect vignette centers across diverse semantic contexts.

B. Preprocessing Pipeline

The preprocessing pipeline implements the GRM technique described in Section IV A, applying spatially-varying blur intensity modulated by the radial mask $M(r)$ as formalized in Eq. (6). The key preprocessing parameters are configured as follows: Gaussian kernel size $k = 91$ pixels, full blur zone radius $r_{start} = 0.88 \cdot d_{half}$, and transition boundary $r_{end} = 0.99 \times d_{half}$, where $d_{half} = \min(W, H)/2$ represents half the smaller image dimension.

The preprocessing is applied post-resize to the model input resolution of 224 × 224 pixels for computational efficiency. This technique effectively corrupts semantic content that could create vignette-like distractors while maintaining the spatial frequency characteristics essential

for accurate vignette boundary detection. Alternative preprocessing approaches, including twist effects, color quantization, and circular motion blur, were evaluated but demonstrated inferior stabilization performance.

Identical preprocessing operations are applied during both training and inference phases to ensure consistency between learned feature representations and test-time inputs, with the preprocessing configuration automatically preserved in model checkpoints to eliminate domain shift between training and deployment scenarios.

C. Training Configuration and Hyperparameter Values

The training optimization employs the Adam optimizer with an initial learning rate of $\eta_0 = 0.001$ and weight decay regularization $\lambda_{decay} = 10^{-4}$. Learning rate scheduling utilizes the ReduceLROnPlateau strategy, monitoring validation loss with a patience of 5 epochs and a reduction factor $\gamma = 0.1$.

The square root normalized penalty loss function described in Eq. (17) is employed with penalty multiplier $\lambda_{pen} = 4.0$ and tolerance constraint $\tau = 0.0078$. Training incorporates mixed precision computation using PyTorch's Automatic Mixed Precision (AMP) with gradient scaling to accelerate convergence while maintaining numerical stability. Gradient clipping with maximum norm $\|\nabla\|_{\max} = 1.0$ prevents gradient explosion during optimization.

The model training exhibits stable convergence behavior without overfitting, with validation and training losses maintaining consistent patterns due to the high structural similarity of vignette geometries across datasets. Early stopping mechanisms were disabled in favor of fixed-epoch training (50 epochs), empirically determined through extended training analysis demonstrating reliable loss stabilization within this timeframe for batch size 64. Smaller batch configurations (e.g., batch size 32) require approximately double the training duration to achieve comparable convergence stability. The complete hyperparameter configuration employed for model training and evaluation is presented in Table V.

TABLE V. COMPLETE HYPERPARAMETER CONFIGURATION FOR FISHEYE VIGNETTE CENTER DETECTION TRAINING

Category	Parameter	Value	Description
Training Parameters	Epochs	50	Maximum training iterations
	Batch Size	64	Samples per training batch
	Learning Rate	0.001	Initial optimizer learning rate
	Weight Decay	10^{-4}	L2 regularization coefficient
	Optimizer	Adam	Gradient-based optimization
	LR Scheduler	ReduceLROnPlateau	Adaptive learning rate strategy
	LR Reduction Factor	0.1	Learning rate decay multiplier
	LR Patience	5	Epochs before LR reduction
Loss Function & Coordinate Prediction	Penalty Multiplier (λpen)	4.0	Square root loss scaling factor
	Tolerance (τ)	0.0078	Coordinate constraint parameter
	Reference Resolution	894×894	0.8 MP error reporting standard
	Numerical Stability (ε)	10^{-8}	Gradient stabilization constant
Data Configuration & Preprocessing	Training Workers	4	Parallel data loading processes
	Validation Samples	64	Fixed samples per validation video
	Data Balancing	Uniform	Temporal frame sampling strategy
	Random Seed	42	Reproducibility initialization
	Gaussian Kernel Size	91	Radial blur kernel dimensions
	Blur Start Ratio ($r^{\%}_{start}$)	0.88	Start of the full blur zone boundary
	Blur End Ratio ($r^{\%}_{end}$)	0.99	Transition to the no blur zone boundary
Regularization & Training Control	Gradient Clipping	1.0	Maximum gradient norm threshold
	Mixed Precision	Enabled	Mixed floating-point precision for training acceleration

D. Hardware and Training Time

All experiments were conducted on a desktop workstation equipped with an NVIDIA GeForce RTX 4060 GPU (8 GB VRAM) and an Intel Core i5-12400F processor (3.6 GHz) with 16 GB system memory. The system operates on Windows 11 Pro with NVIDIA driver version 560.94 supporting CUDA 12.6. Training typically converges within 25–30 epochs, requiring approximately 4–6 h depending on dataset size and preprocessing configuration.

E. Evaluation Metrics

The evaluation framework employs a comprehensive set of metrics designed to assess both pixel-level accuracy and statistical robustness of vignette center predictions, with particular emphasis on their practical implications for sUAV stabilization applications. Our primary evaluation metric utilizes the pixel distance error $\mathcal{L}_{\text{pix}}$ as previously defined in Eq. (15), computed on a standardized 0.8 MP reference resolution (894×894 pixels) following the established benchmark from Gurtner *et al.* [1]. The metric is systematically applied to both training and validation datasets to monitor convergence behavior and assess generalization performance across different scene contexts.

Beyond simple mean error values, we employ percentile-based statistical analysis on the standardized pixel errors to characterize the complete error distribution and assess model reliability. Specifically, we report the 25th percentile (Q25), median (Q50), 75th percentile (Q75), and 95th percentile (Q95) of pixel distance errors across validation sets. The metric Q95 is crucial, as it reveals a reliable picture of the worst error case of our

stabilizer. However, its interpretation is contextualized by the observed distribution: if errors above Q95 are rare and isolated, their practical impact may be limited; if they cluster or persist across scenes, they may signal robustness issues. Hence, percentile metrics are complemented with full error distribution plots.

F. Ablation Study Methodology

To assess the individual contributions of our proposed components, we conduct systematic ablation experiments by selectively disabling the GAC and GRM preprocessing pipeline. The baseline configuration incorporates both components as described in Sections IV.A and C.

Four distinct experimental configurations are evaluated: (1) baseline with both GAC and GRM enabled, (2) GAC disabled while maintaining GRM preprocessing, (3) GRM disabled while maintaining GAC, and (4) both components disabled. When GAC is disabled, the model operates without coordinate prediction constraints, reverting to standard regression output in the $[0, 1]^2$ space. When GRM preprocessing is disabled, input images bypass radial blur processing and are fed directly to the model after standard resizing operations.

Each ablation configuration undergoes complete model training using identical hyperparameters, dataset partitions, and evaluation protocols as the baseline to ensure fair comparison. All models are trained for the full 50-epoch duration to eliminate convergence-related bias, with final model evaluation conducted using the trained checkpoint from the last epoch.

G. Timing Analysis Methodology

Real-time performance evaluation employs frame-by-frame processing analysis to accurately characterize computational requirements without batch processing optimizations that could skew timing measurements. A representative test video containing 600 frames is selected to provide sufficient statistical samples while maintaining computational feasibility for detailed timing analysis.

The timing evaluation isolates three distinct processing stages: (1) GRM preprocessing implementation using CPU-based Gaussian convolution operations, (2) MobileNetV2 model inference executed on GPU hardware, and (3) final image stabilization transformation based on detected vignette center coordinates. Each frame undergoes sequential processing through all stages with individual timing measurements recorded using high-precision system timers.

Frame processing operates in single-frame mode rather than batch processing to reflect realistic deployment scenarios where frames arrive sequentially from camera hardware. Statistical timing analysis includes mean processing time, quartile distribution (Q25, Q50, Q75), and 95th percentile measurements to characterize both typical performance and worst-case latency scenarios. FPS calculations are derived as the reciprocal of total per-frame processing time.

H. Error Distribution Analysis

Comprehensive error characterization employs statistical distribution analysis to assess model reliability beyond simple mean error metrics. The complete validation dataset provides the sample population for distribution analysis, with pixel distance errors computed using Eq. (15) on the standardized 0.8 MP reference resolution. Error distribution visualization utilizes both frequency histograms and probability density estimation to characterize the complete error profile. Percentile analysis provides robust statistical summaries less sensitive to outliers compared to mean-based metrics. The analysis specifically examines distribution shape, skewness toward low-error regions, and tail behavior to identify potential failure modes or systematic biases in prediction accuracy.

Consistency validation examines whether increased validation sampling (beyond the standard 64 frames per video) significantly alters distribution characteristics, ensuring that reported metrics provide stable and representative performance assessment across different sampling strategies.

VI. Experimental Results

A. Training Performance and Convergence

We present detailed experimental results to evaluate the effectiveness of our proposed approach. We trained the model for 50 epochs using the configuration in Table V, which took 7 h and 30 min to complete. Fig. 6 shows the training and validation loss computed using Eq. (17). We show the corresponding learning rate values in Fig. 7. We can observe significant spikes in the validation error before epoch 20, with an important decline after the start of epoch 19, corresponding to the decrease of the learning rate from 10^{-3} to 10^{-4}. We noticed among different training sessions that there is a consistent correlation between the reduction of the learning rate and major validation loss reductions. The validation loss stabilizes after epoch 20, and reaches the lowest value, 0.570, at epoch 45. After stabilization of the curves, we observe a consistent gap between the training and validation curves, indicating a well-controlled generalization without noticeable overfitting. These loss values, although low, are not really interpretable in terms pixel-level jittery. Therefore, we present in Section VI B further quantitative results.

B. Quantitative Performance Analysis

Fig. 8 shows the training and validation errors using interpretable pixel-level metrics. The curves in this figure have the same shape as those in Fig. 6. The final pixel error at epoch 50 is smaller than the quarter of a pixel. Fig. 9 shows four percentiles of the validation errors over the training epochs. At the final epoch (epoch 50), these percentiles show promising results, with 75% of predictions below 0.3 pixel and 95% of predictions below 0.48 pixels.

Fig. 10 shows the complete distribution of the errors on the validation set using the final model (saved at epoch 50). The distribution is normal and skewed to the right (where the small errors lie).

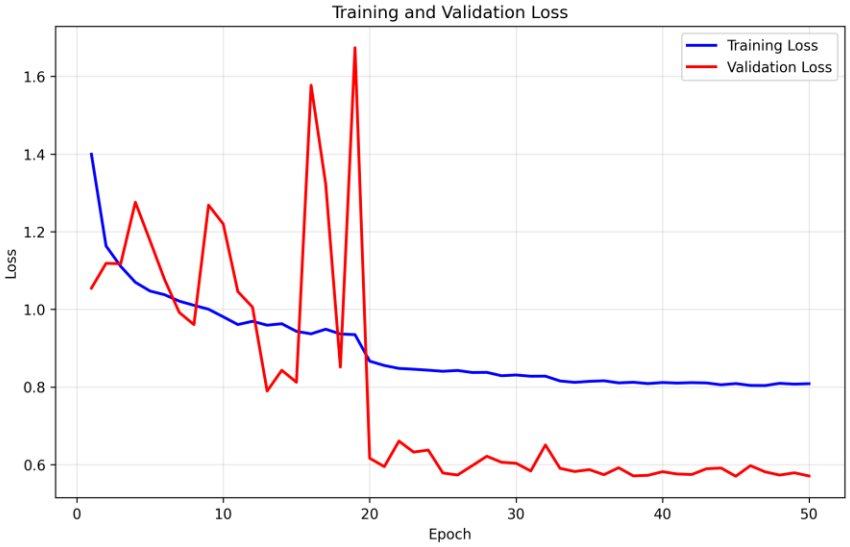


Fig. 6. Training and validation loss curves over 50 epochs.

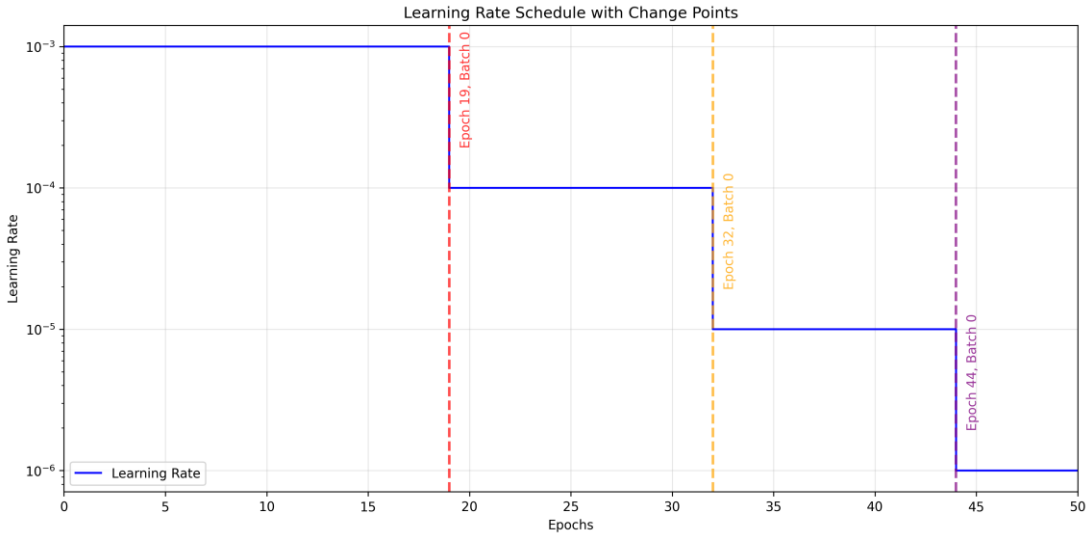


Fig. 7. Learning rate values computed by the scheduler.

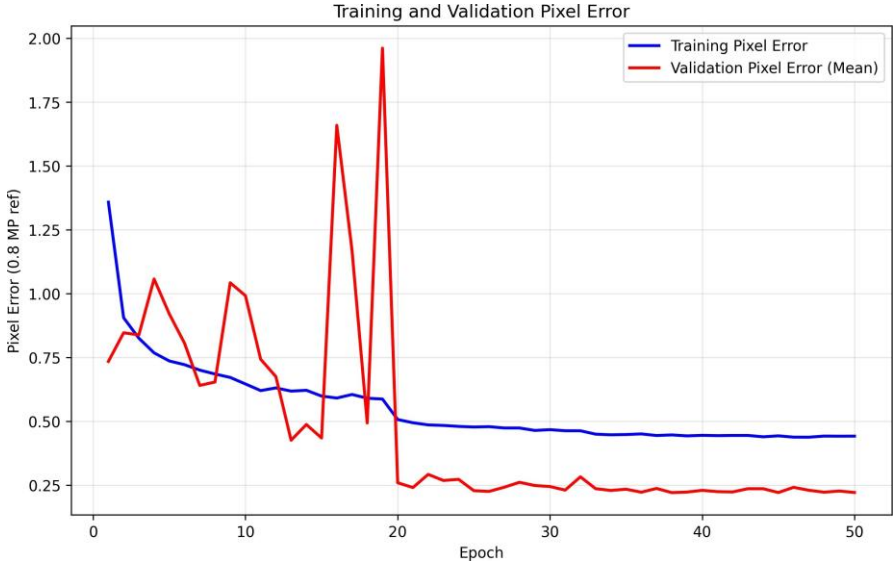


Fig. 8. Training and validation errors in pixels over 50 epochs.

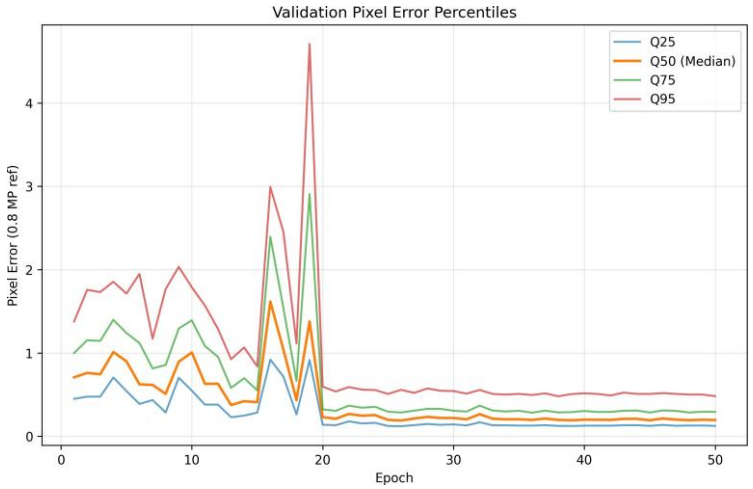
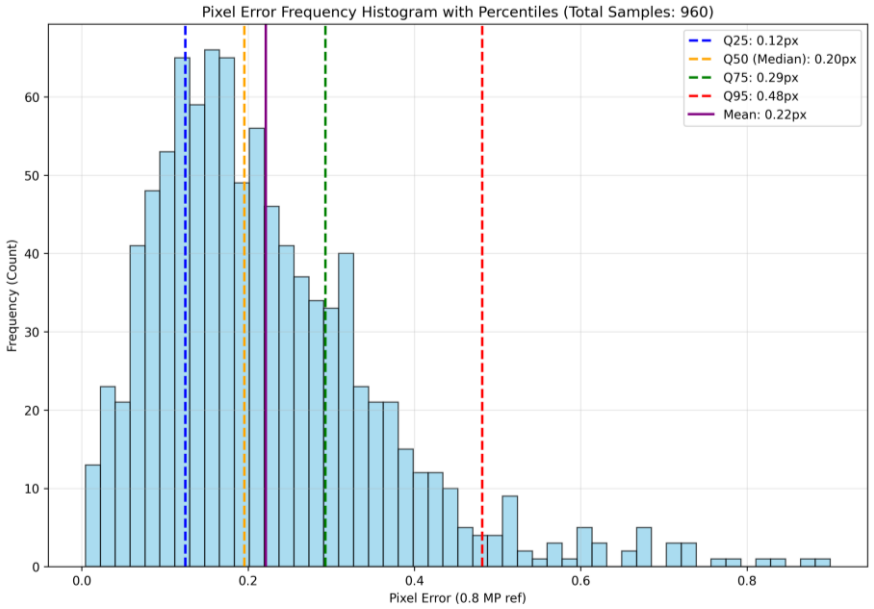
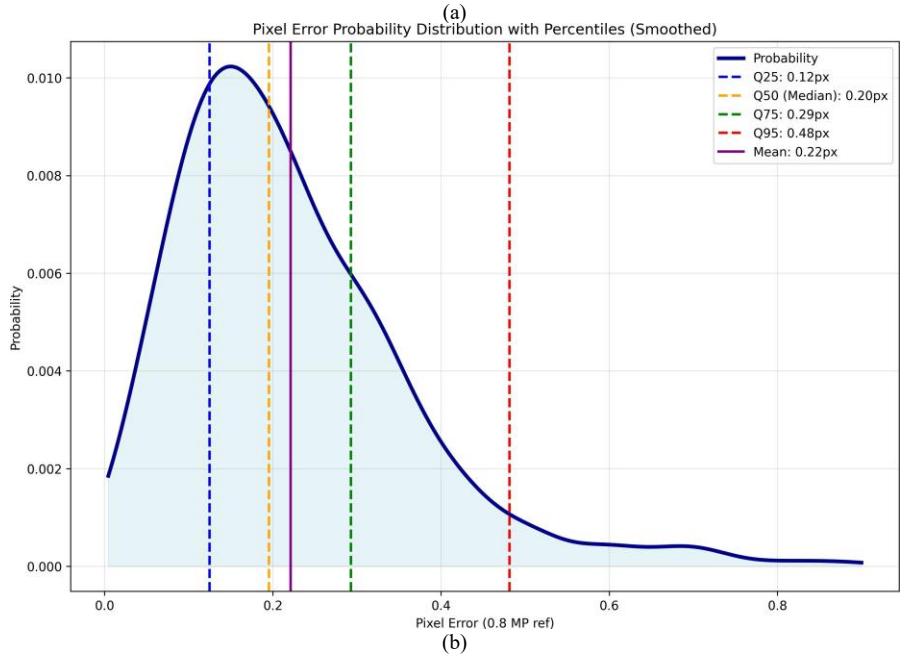


Fig. 9. Percentiles (25th, 50th, 75th, 95th) of validation errors in pixels over the training epochs.



(a)



(b)

Fig. 10. Error distribution of validation errors: (a) Frequency histogram of validation errors with percentiles, (b) Probability distribution of validation errors with percentiles.

C. Time Analysis

Table VI summarizes the statistical timing results for our video stabilizer. The results show a satisfying FPS. The Gaussian preprocessing is performed using the Central Processing Unit (CPU) and takes twice the computational time required by the lightweight model. Nevertheless, we show in Section VI.D, that the Gaussian preprocessing pipeline effectively increases the stabilization performance.

TABLE VI. TIMING STATISTICS PER FRAME (MS) AND FPS OF A 600-FRAME VIDEO

Stage	Mean	Q25	Q50	Q75	Q95
Preprocessing	11.79	11.43	11.73	12.04	12.77
Model	6.16	4.90	6.31	6.34	7.13
Stabilization	1.14	1.09	1.14	1.18	1.26
Total	19.09	17.42	19.18	19.56	21.16
FPS	52.38	57.41	52.14	51.12	47.26

D. Ablations

Table VII reveals that the best performance metrics (in bold) are consistently scored by the baseline method. Disabling our preprocessing pipeline doubles the Q95 error (0.481 vs 0.991), and leads to 2.6× worse Q75 error, while disabling GAC seems consistently less serious across all metrics. The most significant performance drop is shown in the last row of Table VII, which correspond to disabling both GAC and GRM preprocessing.

TABLE VII. VALIDATION PIXEL ERROR METRICS COMPARISON AFTER ENABLING AND DISABLING DIFFERENT COMPONENTS

Disabled Components	Mean	Q25	Q50	Q75	Q95
None (Baseline)	0.221	0.125	0.196	0.293	0.481
GAC	0.341	0.195	0.304	0.447	0.716
GRM	0.530	0.314	0.519	0.722	0.991
Both (GAC and GRM)	0.576	0.326	0.550	0.781	1.131

VII. SCALABILITY OF COMPUTATIONAL PERFORMANCE

While Section VI presented detailed experimental results pertaining to accuracy and latency on desktop hardware, thus, scalability to embedded platforms arises. Thus, we aim to present in this section a theoretical discussion to clarify computational performance on embedded platforms. Since the focus of our study is on low-cost drone systems, we will analyze scalability on the NVIDIA Jetson Nano (4 GB), which costs approximately 100 USD, aligning thus with low-cost drone development requirements. The Jetson Nano has a 921-MHz Maxwell GPU with 128 cores, a quadcore ARM Cortex-A57 CPU operating at 1.43 GHz, 4 GB LPDDR4 memory, and 5–10 W power consumption. We remind the reader that preprocessing and stabilization stage in Table VI use a desktop CPU, whereas the model is run on a GPU. Hence, we will approximately project the computational time of both CPU and GPU from our hardware, which we previously described in Section V D, to that of Nano Jetson. We will apply different projection strategies for CPU and GPU due to their inherent characteristics and downstream tasks.

A. Scaling Approach

1) CPU

For CPU preprocessing, we will apply a linear scaling of FPS using the CPU's clock ratio[2]. This requires a high computational throughput, i.e., latency is mostly spent on performing multiply-adds rather than memory transfer operations. We argue that FLARE's CPU operations, namely preprocessing and stabilization stages, exhibit the computational throughput required to roughly scale the latency using CPU clocks. We base our argument on three points:

(1) Small image size;
(2) Regular memory access;
(3) Vectorized computations.

We detail our three-point argument in the following, which all pertain to different aspects of memory latency reduction:

The small image size indicates its susceptibility to fitting into the CPU's cache, thus reducing the memory transfer time thanks to the higher speed of cache compared to RAM. The CNN we employed requires only an image shape of 224×224, where each pixel is only 1 byte. Additionally, the GRM blurs only one grayscale image, which means that only one channel requires memory transfer, thereby avoiding the need to transfer two additional 224×224 channels. The required input image to the model, 224×224×3, is achieved through stacking three replicas. Hence, the individual image data is only 49 kB, fitting easily into the L2 cache of the target CPU. Regular memory access is a key property of the Gaussian filtering we integrated into the GRM. In other words, the memory access pattern is predictable, allowing quicker prefetch of required data and lower CPU stalls. Additionally, the Gaussian filtering is easily vectorized, enabling the CPU's SIMD (Single Instruction Multiple Data). SIMD increases the number of operations per CPU clock, thereby reducing the memory access count. We emphasize that SIMD doesn't reduce the amount of transferred data but rather affects the number of times the memory is accessed to fetch this data. These three factors contribute collectively to intense computational throughput, thus making CPU clock scaling a reasonable approach.

2) GPU

Since CNN inference on embedded GPUs is memory bandwidth limited rather than compute-bound, naive clock-based scaling would be deceptive for GPU-bound inference (model stage in Table VI). We rely on recent published benchmarks that directly measure MobileNetV2 inference time on Jetson Nano instead of estimating performance through theoretical projections.

The mean inference time for MobileNetV2 (224×224 input, FP16 precision, feature extractor configuration without classification head) on Jetson Nano is 12.08 ms, according to a recent benchmark conducted by Tobiasz *et al.* [30]. Since our model had an FP32 precision

[2] Ratio between our CPU clock and the Jetson Nano CPU clock.

during training, we reduced the precision down to FP16 prior to reporting the timing statistics in Table VI, thus making the time scaling more reliable. We note that the reduced FP16 precision did not decrease any validation pixel error metric, indicating that FP32 was an overkill. Our desktop GPU achieves a mean inference time of 6.16 ms, confirming a 1.96× slowdown on Jetson Nano. Although our implementation adds a lightweight regression head (512 → 128 → 2 neurons) to the benchmark's standard MobileNetV2 feature extraction, the additional computational overhead is minimal (<0.5 ms based on typical dense layer performance).

B. Scaling Estimation on Jetson Nano

We use the projection factors we established in Section VII A and present comprehensive scaling performance in Table VIII. We apply scaling factor 3–3.5× for CPU-bound preprocessing, 1.96× for GPU-bound inference, and 1× for negligible stabilization overhead. The projections indicate mean throughput of 18–21 FPS with quartile range spanning 17–22 FPS, representing viable real-time operation for UAV applications with moderate frame rate requirements. The preprocessing stage contributes 35–41 ms of 48–54 ms mean total (73–76%), confirming it as the primary optimization target for embedded deployment. We show in Section VIII that our method FLARE exhibits a significant room for latency enhancements.

TABLE VIII. PROJECTED TIMING STATISTICS ON JETSON NANO (MS AND FPS).

Stage	Mean	Q25	Q50	Q75	Q95
Preprocessing	[35, 41]	[34, 40]	[35, 41]	[36, 42]	[38, 45]
Inference	12.1	9.6	12.4	12.4	14.0
Stabilization	1.14	1.09	1.14	1.18	1.26
Total	[48, 54]	[45, 51]	[49, 55]	[50, 56]	[53, 60]
FPS	[18, 21]	[20, 22]	[18, 20]	[18, 20]	[17, 19]

Brakets [,] Indicate the CPU's Scaling Factor Range 3–3.5× and its Effect on Total Estimations

VIII. DISCUSSION AND FUTURE WORK

A. Performance and Optimization Opportunities

With real-time desktop performance (52.38 FPS) and feasible embedded throughput (18–21 FPS projected on Jetson Nano), FLARE achieves sub-pixel stabilization accuracy. Significant improvement potential exists for both speed and memory efficiency because our study did not optimize model architecture (layer count, width), and FP32 to FP16 precision reduction showed zero accuracy degradation. Our layer activation analysis suggests significant compression may result from structured pruning that targets dead neurons and redundant filters.

There should be more research done on the GRM preprocessing bottleneck. The ablation results show a promising research direction: suppressing RGB semantics inside the fisheye vignette significantly improves center detection, even though our tests confirm that smaller kernel sizes degrade accuracy. A worthwhile line of inquiry is the creation of substitute preprocessing methods that maintain this semantic suppression while lowering computational overhead.

Promising real-time performance gains are possible with multitask processing. Stabilization does not have to be a blocking step for vision tasks that do not require temporal constraints (e.g., instance segmentation, depth estimation). Instead of pre-processing the input, the stabilization transform calculated from vignette center detection can be applied post-processing to task outputs (such as segmented masks), allowing parallel execution that increases throughput.

B. Dataset Limitations and Validation Strategy

Based on real-world measurements from Gurtner *et al.* [1], we applied artificial vibrations on Fisheye8K dataset [29]. This method is in line with established fisheye computer vision research practices: for instance, SynWoodScape uses synthetic data for autonomous driving applications [31]. Practical benefits of synthetic generation include: large-scale dataset creation becomes feasible where manual frame-by-frame annotation would be impractical; annotations are automatically derived from known transformations with perfect accuracy; and the fixed-camera source allows replicating one annotation across hundreds of frames in the same video. Since vignette center detection relies on lens geometry rather than scene content, the fixed-camera source footage is suitable for this task. In order to isolate the vignette boundary, which stays constant regardless of camera motion through space, our GRM preprocessing eliminates most of the RGB semantic information. Regardless of the drone's motion, the internal lens vibrations take place inside the optical assembly. This is corroborated by cross-domain validation, which shows that our model trained on outdoor scenes and tested on indoor environments retains accuracy.

To extend this work to real-world sUAV vibration footage, while bearing in mind the required annotation labor, a practical solution balances data scalability with domain-specific adaptation by combining synthetic pre-training on large-scale data with fine-tuning on manually annotated real vibration sequences.

In summary, FLARE provides an effective method to stabilize internal fisheye lens vibrations in inexpensive drones, demonstrating that fisheye-specific preprocessing outperforms generic methods.

IX. CONCLUSION

Internal micro-vibrations in fisheye lenses on inexpensive drones pose a significant challenge. In order to fill a gap in digital stabilization techniques that ignore lens vibrations, we developed FLARE, a stabilization framework that uses the vignette center for sub-pixel jittery compensation. FLARE combines two major contributions: GRM, which suppresses distractions while maintaining vignette boundaries, and GAC, which leverages prior center position information during both training and inference. Our experimental results reveal sub-pixel accuracy, with a mean error of 0.221 pixels even when boundaries of vignette circle are blurred, which

misleads the prior edge-based fisheye stabilizer relying on hand-crafted edge features. Real-time viability for resource-constrained sUAVs is considered by stage-wise latency analysis, which shows 52.38 FPS on our desktop and scales to 18–21 FPS on Jetson Nano. FLARE lays the groundwork for effective, accurate stabilization in embedded aerial imaging systems and provides a digital substitute for expensive mechanical stabilizers.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Y.Z. conceptualized the research topic, proposed the core methodological innovations (Gaussian Radial Mask and Geometric Anchor Constraint), derived the mathematical framework quantifying ground displacement from internal optical vibrations, designed and developed the annotation software, performed model training and testing, conducted ablation studies, and drafted the initial manuscript. S.S. conducted the comprehensive literature review, identified the critical research gap in fisheye lens stabilization, designed and developed the annotation software, reviewed the synthetic vibration dataset, performed model training and testing, analyzed and interpreted the numerical results, and wrote the final paper. N.O. reviewed and validated the synthetic dataset, conducted ablation studies, analyzed computational scalability on the NVIDIA Jetson Nano embedded platform, and contributed to the analysis and interpretation of numerical results. R.O. reviewed the synthetic dataset, analyzed computational scalability on the NVIDIA Jetson Nano embedded platform, and supported data analysis and experimental validation. H.G. proposed experiments that enhanced model performance, provided technical guidance, supervised the research methodology, and contributed to manuscript review and editing. All authors reviewed and approved the final version of the manuscript.

REFERENCES

[1] A. Gurtner, R. Walker, and W. Boles, "Vibration compensation for fisheye lenses in UAV applications," in *Proc. Digital Image Computing: Techniques and Applications (DICTA)*, 2007, pp. 218–225.

[2] N. Purwono and A. Syetiawana, "Application of UAV with fish-eye lenses camera for 3D surface model reconstruction," *Geoplanning: Journal of Geomatics and Planning*, vol. 5, no. 1, pp. 115–130, 2018.

[3] G. Ronchetti, D. Pagliari, and G. Sona, "DTM generation through UAV survey with a fisheye camera on a vineyard," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42, pp. 983–989, 2018.

[4] H. Kim, J. Jung, and J. Paik, "Fisheye lens camera-based surveillance system for wide field of view monitoring," *Optik*, vol. 127, no. 14, pp. 5636–5646, 2016.

[5] J. S. Oh, Y. M. Han, and S. B. Choi, "Vibration control of a camera mount system for an unmanned aerial vehicle using piezostack actuators," *Smart Materials and Structures*, vol. 20, no. 8, 085020 2011.

[6] A. Lim, B. Ramesh, Y. Yang *et al.*, "Real-time optical flow-based video stabilization for unmanned aerial vehicles," *Journal of Real-Time Image Processing*, vol. 16, no. 6, pp. 1975–1985, 2019.

[7] J. Wei, C. F. Li, S. M. Hu *et al*., "Fisheye video correction," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 10, pp. 1771–1783, 2011.

[8] M. Wang, G. Y. Yang, J. K. Lin *et al*., "Deep online video stabilization with multi-grid warping transformation learning," *IEEE Transactions on Image Processing*, vol. 28, no. 5, pp. 2283–2292, 2018.

[9] S. Islam, A. Karam, A. Boualem *et al*., "6G-enabled network security in drone technology: Revolutionizing communications and enhancing security applications," in *Proc. Nordic e-Infrastructure Collaboration Conference*, 2024, pp. 28–44.

[10] T. Ahmad, A. Morel, N. Cheng *et al*., "Future UAV/Drone systems for intelligent active surveillance and monitoring," *ACM Computing Surveys*, vol. 58, no. 2, pp. 1–37, 2025.

[11] C. D. Rodin, F. A. D. Alcantara-Andrade, A. R. Hovenburg *et al*., "A survey of practical design considerations of optical imaging stabilization systems for small unmanned aerial systems," *Sensors*, vol. 19, no. 21, 4800, 2019.

[12] C. Cai, X. Weng, and Q. Zhu, "Sea-skyline-based image stabilization of a buoy-mounted catadioptric omnidirectional vision system," *EURASIP Journal on Image and Video Processing*, vol. 2018, no. 1, p. 1, 2018.

[13] Y. Zardoua, M. Boulaala, M. E. Mrabet, and A. Astito, "A fast horizon detector and a new annotated dataset for maritime video processing," arXiv Preprint, arXiv:2110.13694, 2024.

[14] S. Liu, L. Yuan, P. Tan, and J. Sun, "Bundled camera paths for video stabilization," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, pp. 1–10, 2013.

[15] A. Regensky, C. Herglotz, and A. Kaup, "A novel viewport adaptive motion compensation technique for fisheye video," in *Proc. ICASSP 2021–2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 1565–1569.

[16] S. Yang, C. Lin, K. Liao, and Y. Zhao, "Innovating real fisheye image correction with dual diffusion architecture," in *Proc. IEEE/CVF International Conference on Computer Vision*, 2023, pp. 12699–12708. doi: 10.1109/ICCV51070.2023.01166

[17] A. Karpenko, D. Jacobs, J. Baek, and M. Levoy, "Digital video stabilization and rolling shutter correction using gyroscopes," *CSTR*, vol. 1, no. 2, 13, 2011.

[18] P. Eline, "Drone provided with a video camera delivering corrected image sequences of the wobble effect," French Patent Application FR3044141A1, 2017. (in French) https://patents.google.com/patent/FR3044141A1/en

[19] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

[20] A. G. Howard, M. Zhu, B. Chen *et al*., "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv Preprint, arXiv:1704.04861, 2017.

[21] M. Sandler, A. Howard, M. Zhu *et al*., "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[22] C. Li, Y. Qu, Z. Song *et al*., "Research on SMA motor modelling and control algorithm for optical image stabilization," *Microsystem Technologies*, vol. 31, no. 6, pp. 1381–1400, 2025.

[23] Z. Li, M. Lao, S. K. Phang *et al*., "Development and design methodology of an antivibration system on micro-UAVs," in *Proc. International Micro Air Vehicle Conference and Flight Competition (IMAV)*, 2017, pp. 223–228.

[24] J. Choi, J. Park, and I. S. Kweon, "Self-supervised real-time video stabilization," arXiv Preprint, arXiv:2111.05980, 2021.

[25] D. Wulich and N. S. Kopeika, "Image resolution limits resulting from mechanical vibrations," *Optical Engineering*, vol. 26, no. 6, pp. 529–533, 1987.

[26] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263–7271.

[27] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," arXiv Preprint, arXiv:1804.02767, 2018.

[28] G. Jocher, A. Chaurasia, A. Stoken *et al*. (2022). ultralytics/yolov5: v6.1-TensorRT, TensorFlow edge TPU and OpenVINO export and inference version v6.1. [Online]. Available: https://zenodo.org/records/6222936

[29] M. Gochoo, M. E. Otgonbold, E. Ganbold *et al*., "Fisheye8k: A benchmark and dataset for fisheye camera object detection," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 5305–5313.
[30] R. Tobiasz, G. Wilczynski, P. Graszka *et al*., "Edge devices inference performance comparison," *Journal of Computing Science and Engineering*, vol. 17, pp. 51–59, 2023.
[31] A. R. Sekkat, Y. Dupuis, V. R. Kumar *et al*., "SynWoodScape: Synthetic surround-view fisheye camera dataset for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 7, pp. 8502–8509, 2022.