Artificial Microsaccade Compensation: Stable Vision for an Ornithopter

Levi Burner,[1*] Guido C. H. E. de Croon,[2] Yiannis Aloimonos[1,3]

[1]Department of Computer Science, University of Maryland, College Park

[2]Faculty of Aerospace Engineering, Delft University of Technology

[3]Institute for Advanced Computer Studies, University of Maryland, College Park

[*]Corresponding author. Email: lburner@umd.edu (Levi Burner)

Abstract

Animals with foveated vision, including humans, experience microsaccades, small, rapid eye movements that they are not aware of. Inspired by this phenomenon, we develop a method for "Artificial Microsaccade Compensation". It can stabilize video captured by a tailless ornithopter that has resisted attempts to use camera-based sensing because it shakes at 12-20 Hz. Our approach minimizes changes in image intensity by optimizing over 3D rotation represented in $SO(3)$. This results in a stabilized video, computed in real time, suitable for human viewing, and free from distortion. When adapted to hold a fixed viewing orientation, up to occasional saccades, it can dramatically reduce inter-frame motion while also benefiting from an efficient recursive update. When compared to Adobe Premier Pro's warp stabilizer, which is widely regarded as the best commercial video stabilization software available, our method achieves higher quality results while also running in real time.

Introduction

Flapping flying robots, or ornithopters, stand to revolutionize micro air vehicles, because they are safer, quieter, resilient to contact, and more aesthetically appealing than their rotor based counterparts. However, their wing-based actuation induces shaking that makes camera-based perception difficult (*1*). In particular, tailless flapping robots are of interest because they can hover and are more agile than their tailed counterparts (*2–5*) and can even imitate agile insect behaviors (*6*). However, tailless ornithopters also shake aggressively, which creates difficulties when using vision-based sensing (*7–9*). The deteriorated visual sensing is a major obstacle on the road to real-world application of flapping wing robots, as it makes it hard to use cameras for autonomous flight and applications.

A particularly popular platform is the Flapper Nimble+, which is a tailless ornithopter whose original design was used to demonstrate that fruit flies rely on torque coupling to perform rapid bank maneuvers (*6*). As a result, the platform is quite agile, however, it also shakes aggressively because it cannot rely on the tail for damping. Moreover, the rotations due to shaking are irregular since the two wing pairs are actuated independently to generate the moments necessary for flight maneuvers. There is potential for significant advancements in the mechanical capabilities of flapping robots (*10*). However, applications remain limited because it has proven difficult for tailless flapping robots to control themselves via an onboard camera, given their limited payload capacity and aggressive vibration (*11*).

One work introduced a mechanical camera stabilization system, however, this introduces complexity and weight, and so a software solution is preferred if possible (*9, 12*). Recent work has shown that event cameras are the ideal sensor for these platforms, however the weight of currently available systems, and their need for a powerful onboard computer has made implementation challenging (*7, 8*). Vision based control of a tailed flapping robot has been achieved

with wireless monocular (*3*) and a specialized miniature stereo camera system (*13*). However, rolling shutter effects remain a persistent challenge (*9*). Recently, video stabilization has been shown to be beneficial for both frame and event based perception with a focus on flapping robots. However, the method relies on access to ground truth orientation (*14*). Hence, the shaking of vision sensors remains a challenging problem for autonomous flapping wing robots.

An interesting parallel can be drawn to animals with foveated vision, such as humans, who constantly experience rapid, small eye movements called microsaccades. These small rotational eye movements are not perceived in our consciousness, and while their purpose or lack thereof is still a matter of debate, there is a consensus that the brain employs several techniques to mask, or suppress, these movements so that the world appears to be viewed from a stable orientation (*15*). Thus, it is of interest to develop an artificial method for compensating microsaccade like motion for robots that shake.

In this paper, we describe a real-time method for video stabilization, which we call "Artificial Microsaccade Compensation". It directly minimizes differences in image intensity, without relying on feature matching. The resulting rotations are used to estimate a stable, slowly varying orientation to which frames are rendered via a stabilizing rotation. The optimization is conducted over the rotation group of $SO(3)$ using an efficient, equivariant approach based on the inverse compositional Lucas-Kanade method (*16*).

While direct methods for computing image motion have been known for some time (*17*), modern open-source libraries lack the capabilities required to directly estimate nonlinear warps, such as 3D rotations, in a direct and efficient manner without resorting to feature matching. Consequently, we open-source our implementation, in the hope that other practitioners with shaking robots can stabilize their image streams by rendering them to a stable viewpoint.

We make the following contributions:

- A framework and implementation for "Artificial Microsaccade Compensation". That makes clear how a robot can represent a video from a stabilized orientation without apparent distortion, and which can be adapted for other applications.

- An efficient implementation of a direct Lucas-Kanade style method for image rotation estimation based on equivariance and optimization in $SO(3)$ with updates computed in the Lie Algebra $so(3)$. To the best of our knowledge, an equivalent implementation is not available.

- Evaluation of the method in a particularly difficult stabilization problem arising from the shaking of a tailless ornithopter, the Flapper Nimble+, which has thus far admitted only limited application of camera based sensing (*11*). The result is the first stable videos from a camera on the platform.

Results

The algorithm was tested on a Flapper Nimble+ manufactured by Flapper Drones equipped with a wireless FPV camera. Videos showing stabilization results are available in the supplemental material. Figure 1 shows several qualitative results, demonstrating a dramatic improvement of the image stability. In particular, the motion vectors from normal flow estimation, shown as red arrows, show that the apparent image motion is reduced drastically. Figure 2 shows per-frame metrics for sequence FF2. The metrics are described below. Tables 1 and 2 detail quantitative results averaged over each sequence.

The videos in the supplementary information include side-by-side comparison with Adobe Premier Pro's warp stabilizer, which is widely regarded as the best commercial video stabilization software available. Adobe's documentation indicates that their method relies on feature
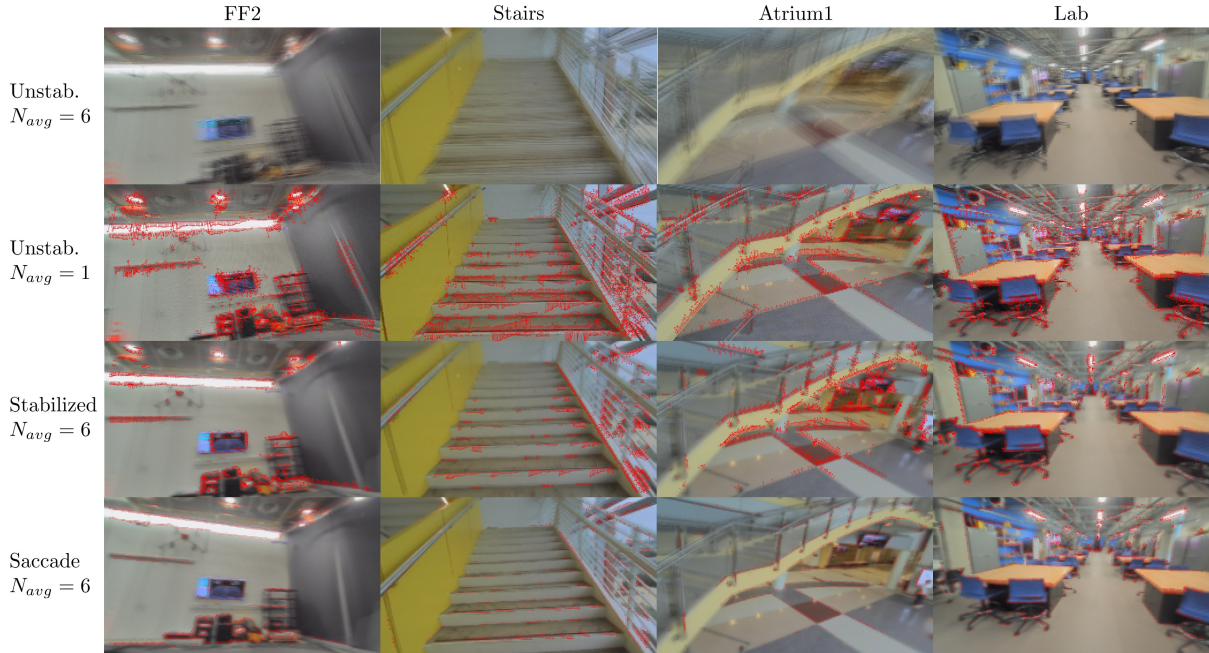
Figure 1: Qualitative comparison of stabilized and unstabilized frames. Top row, 6 averaged consecutive frames from a sequence illustrating the extreme shaking of a camera onboard the Flapper Nimble+ tailless ornithopter. Second row, a single unstabilized frame with normal flow (the projection of optical-flow onto gradients) illustrated with red arrows. Third row, 6 stabilized and averaged frames, which were warped to the stabilized viewpoint. Bottom row, 6 stabilized and averaged frames which were warped to a stationary viewpoint that occasionally sacccades. The flow magnitude can be seen to reduce with each row, with the saccading variation of the algorithm achieving a drastic reduction in flow magnitude. Videos showing stabilization results are available in the supplementary information.
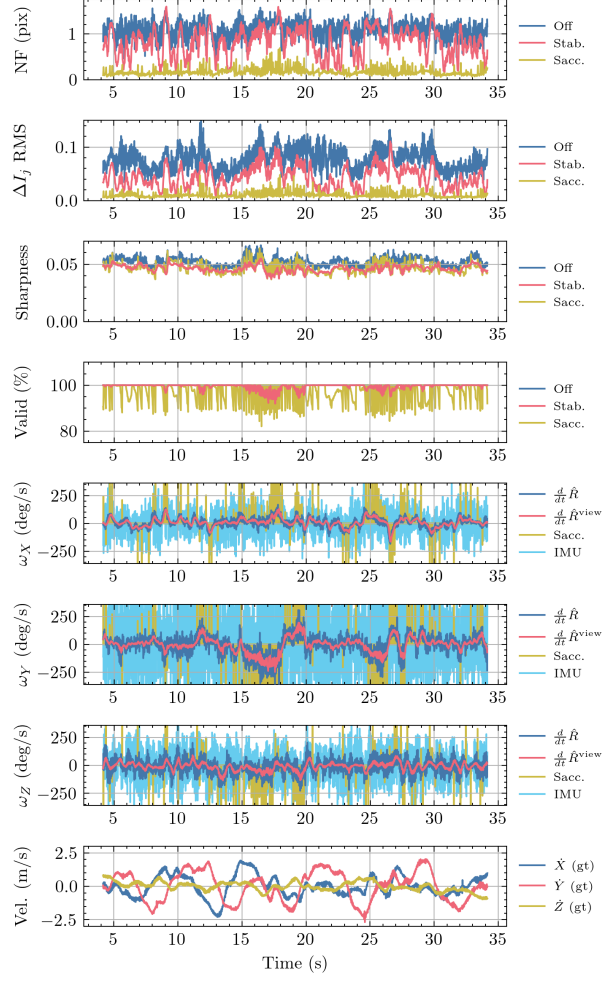
Figure 2: Per frame metrics computed on sequence FF2. Normal flow and the RMS change in image brightness are reduced by the stabilization and significantly reduced by saccading instead of continuously rotating the stable view. The proposed frame averaging, which reduces the effects of rolling shutter and image artifacts, slightly reduces the stabilized image sharpness. The stabilization algorithm maintains a high quantity of valid pixels given 12.5% image margins. The saccading variant results in slightly fewer valid pixels. During flight, the angular velocities from the Flapper's onboard IMU do not align with the angular velocities of the image based orientation estimate. The stabilized orientation does not contain the high frequency oscillations of the estimated orientation. The saccade style stabilization maintains a viewpoint angular velocity of zero, except when it saccades, which results in an angular velocity spike.

matching and subspace warping (*18*). In most sequences, the video stabilized by Adobe Premier Pro is still vigorously shaking, and frequent distortions are introduced which make nearby objects look like they are stretching. In some sequences, Adobe Premier Pro fails to produce a suitable result, with most of the frame remaining empty. The warp stabilizing plugin was configured with the maximum smoothness setting (100%) and with defaults for all other settings. On some sequences, the software automatically advised disabling cropping, which we did. It must be noted that the Adobe software uses several additional gigabytes of RAM during stabilization, which is conducted in an offline, batch processing mode. Our method is capable of running online, achieves real-time performance, and uses several hundred megabytes total, most of which can be attributed to its research-quality code which is written in Python and uses the high level JIT compilation and autodifferentiation framework Jax (*19*). Specifically, autodifferentiation is used to precompute the static Jacobians used for every Gauss Newton step.

For all results, the images were downsampled by 4x to 320×180 resolution. 12.5% margins were applied to each side of the output frames to limit the appearance of border effects due to the stabilized view's field of view being outside of the real camera's field of view. The final output resolution was 240×152. At this resolution, the algorithm achieved real-time performance as needed for future downstream applications. Specifically, on sequence Atrium2, it achieved 67 and 77 fps in the stabilized and saccade modes, respectively on a i7-12800HX laptop CPU. Note that our code is implemented in Python, is almost entirely single threaded, loads images from disk in an uncompressed format, and does not use hardware acceleration for image interpolation which would speedup undistortion, the Lucas-Kanade tracker, and computation of I^{stab}. Figure 5 contains examples of high-resolution stabilized frames without downsampling.

7

Metric	Method	Sequence									
		UD	LR	FB	Yaw	Circle1	Circle2	FF1	FF2	FF3	Hand
NF RMS	None	1.047	1.106	1.077	1.168	1.082	1.085	1.120	1.102	1.155	1.025
(pix.)	Stab.	0.598	0.739	0.801	0.959	0.855	0.810	0.874	0.876	0.973	0.912
	Sacc.	0.129	0.155	0.157	0.207	0.169	0.175	0.182	0.174	0.197	0.103
ΔI_j RMS	None	0.075	0.080	0.070	0.089	0.073	0.076	0.083	0.079	0.080	0.078
	Stab.	0.032	0.041	0.038	0.062	0.049	0.040	0.049	0.048	0.053	0.062
	Sacc.	0.009	0.010	0.010	0.014	0.011	0.011	0.012	0.011	0.012	0.007
Sharpness	None	0.056	0.055	0.050	0.054	0.052	0.052	0.053	0.053	0.051	0.056
	Stab.	0.050	0.049	0.045	0.046	0.046	0.046	0.047	0.046	0.045	0.051
	Sacc.	0.050	0.048	0.045	0.048	0.047	0.045	0.047	0.047	0.046	0.051
Valid Pix.	None	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
(%)	Stab	100.0	99.8	99.7	97.1	99.2	99.9	99.4	99.4	99.1	99.2
	Sacc.	97.8	97.4	97.6	96.8	97.2	97.4	97.2	97.1	97.2	96.7
ω RMS	MoCap	238	229	259	294	282	258	260	249	310	143
(deg/s)	IMU	301	289	268	368	347	276	308	311	325	90
	Image	85	97	88	157	128	93	107	105	124	92
ω^{view} RMS	Stab.	32	50	43	130	99	48	71	70	87	82
(deg/s)	Sacc.	21	44	39	129	97	42	66	65	89	75
V RMS (m/s)	MoCap	0.908	1.157	1.294	0.815	1.410	1.576	1.338	1.313	1.658	0.591

Table 1: Image quality, stabilization quality, angular velocity, and translational velocity metrics for ten sequences collected in a motion capture lab, each featuring different types of motion.

Metric	Method	Sequence				
		Hall	Stairs	Atrium1	Atrium2	Lab
NF RMS	None	1.026	1.104	1.053	1.119	1.052
(pix.)	Stab.	0.554	0.694	0.734	0.853	0.574
	Sacc.	0.229	0.231	0.170	0.192	0.203
ΔI_j RMS	None	0.048	0.073	0.078	0.079	0.078
	Stab.	0.018	0.029	0.043	0.048	0.033
	Sacc.	0.009	0.011	0.010	0.010	0.012
Sharpness	None	0.047	0.053	0.054	0.053	0.060
	Stab.	0.042	0.045	0.047	0.046	0.053
	Sacc.	0.041	0.045	0.047	0.046	0.052
Valid Pix.	None	100.0	100.0	100.0	100.0	100.0
(%)	Stab.	100.0	100.0	99.7	99.5	100.0
	Sacc.	97.3	98.0	97.7	97.4	97.6
ω RMS	IMU	250	273	273	276	234
(deg/s)	Image	80	99	101	110	80
ω^{view} RMS	Stab.	24	39	57	63	32
(deg/s)	Sacc.	17	32	49	58	22

Table 2: Image quality, stabilization quality, and angular velocity metrics for five sequences collected indoors.

Sequences

Ten sequences, each approximately 20 seconds in length, were collected in a motion capture room. The Flapper was piloted manually to achieve the following trajectories. Up-down (UD), left-right (LR), front-back (FB), yaw in place, fly in a circle (Circle1, Circle2), fly freely according to the operators discretion (FF1, FF2, FF3). The tenth sequence was collected by holding the flapper in hand, and quickly tilting it back and forth along each primary axis, followed by a smooth figure 8 motion. This in-hand trajectory is meant to demonstrate the best possible image quality when the camera's motion is smooth.

An additional five sequences, each approximately 20 seconds in length, were collected in a variety of indoor settings. Due to airspace restrictions, we were unable to collect data outside. The sequences include a hallway, stairwell, a large atrium (2x), and a robotics laboratory. These sequences test the algorithm with a variety of visual features and scene depths.

Metrics

We compute a variety of metrics to demonstrate the image stabilization quality. The first is the RMS difference between consecutive output images. As can be expected, the stabilized frames feature a noticeable reduction in RMS difference between subsequent frames, often achieving an approximately 50% reduction. Further, the saccade variant, which results in stabilized frames with a fixed orientation, achieves an approximately $7\times$ reduction. These results can be expected because the Lucas-Kanade tracker minimizes the MSE between frames, though not necessarily pairs adjacent in time.

Next, we estimate image motion in the unstabilized, stabilized, and saccade stabilized frames using RMS normal flow magnitude. Normal flow has units of pixel displacement that is the magnitude of the optical flow onto the image gradients and can be estimated without the assumptions required for feature matching. The normal flow at a pixel is given by

$$n(p) = -\nabla_t I(p) \frac{\nabla_p I(p)}{\|\nabla_p I(p)\|^2}. \tag{1}$$

The gradients are estimated using a Sobel filter with a kernel size of 3. To avoid erroneous estimation from gradients of small magnitude, pixels with gradients less than $15/255$ are discarded from the RMS normal flow magnitude calculation. The unstabilized frames experience flow magnitudes of approximately 1.1 pixels per frame. Stabilization reduces the normal flow magnitude to 0.5 to 0.9 pixels, however, it must be remembered that stabilization does not eliminate the measured rotational motion. Instead, it limits the high-frequency components of that motion. Finally, saccade-style stabilization reduces the normal flow magnitude to approximately 0.1 to 0.23 pixels.

Averaging frames results in a slight reduction in image sharpness. We measure image sharpness as the RMS magnitude of the image gradients estimated across all three color channels. While RMS contrast (or image variance) is a more typical measure of sharpness, it failed to quantify the reduction in sharpness due to frame averaging in this application. Our chosen measure indicates an approximately 10% reduction in the sharpness measure when $N_{avg} = 6$.

The average number of valid pixels, that is, pixels in I^{stab} which were filled by each of the averaged frames, is reported. The stabilized frames typically contain more than 99% valid pixels when $N_{avg} = 6$. Saccade stabilization achieves a lower score, as expected, but still typically above 97%.

We report the average angular velocity estimated by motion capture, the Flapper's onboard IMU, image based orientation estimate, and the stabilized view. As will be discussed later, the IMU and motion capture angular velocity estimates appear to be relatively inaccurate during flight, but accurate when the flapper is moved in-hand. We include them for completeness. The average angular velocity measured by the IMU is typically more than 250 deg/s and can be as high as 370 deg/s. Motion capture reports lower angular velocities ranging from 230 to

about 300 deg/s. The image-based orientation estimator reports much lower angular velocities between 80 and 160 deg/s. Notably, in the in-hand sequence, where the flapper was not shaking, the IMU and image based angular velocity estimates are almost identical. The stabilized view's average angular velocity is between 30 and 130 deg/s in all sequences and typically below 90 deg/s. Note that the saccade stabilized view's angular velocity is zero at all times except when a saccade occurs.

For the sequences recorded in the motion capture room, we report the flapper's average translational velocity which ranged between 0.9 and 1.7 m/s.
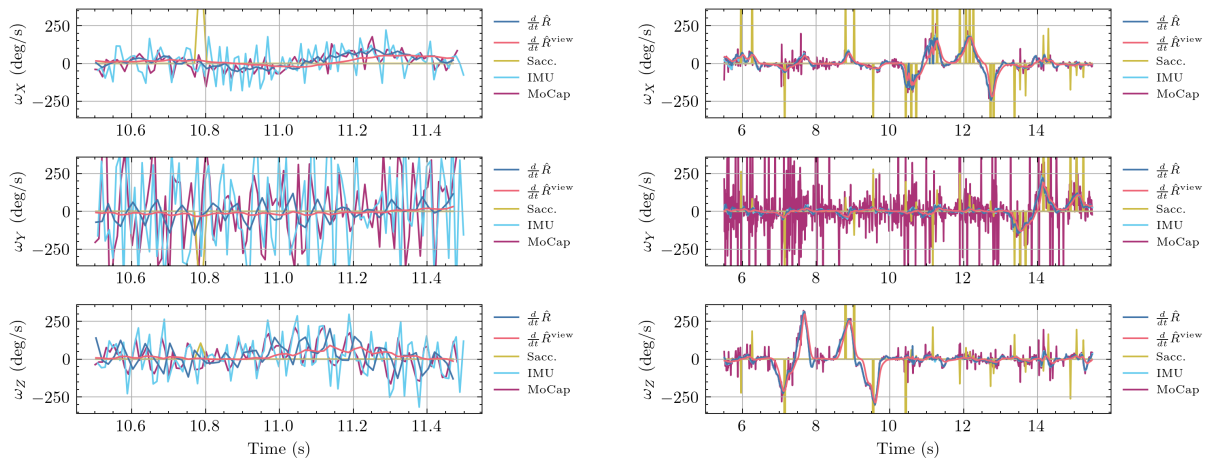
IMU Angular Velocity Estimates

Figure 3: Zoomed in view of angular velocity estimates. **Left**: Angular velocity estimates from an onboard IMU, motion capture, the image orientation tracker, stabilized view, and saccade times during flight. The Flapper's onboard IMU fails to capture accurate enough angular velocity estimates for image stabilization during flight. **Right:** When moved smoothly by hand, the Flapper's onboard IMU's angular velocity estimates almost perfectly align with the image based angular velocity estimates. The motion capture system's angular velocity estimates also align, except for estimates along the Y axis, which are poor in quality because of occluded markers.

The angular velocity estimates from the Flapper's onboard IMU are not accurate enough during flight to achieve an appreciable stabilization effect. This can be seen clearly from the left

side of Figure 3 where the IMU estimates are not aligned with image based velocity estimates computed from $\hat{R}$. In particular, the estimates from the IMU have a much larger magnitude than those measured from the image. However, when the Flapper is moved smoothly, as in the Hand sequence, the IMU's angular velocities almost perfectly match with our image-based estimate as shown on the right side of Figure 3. The motion capture system also has difficulty estimating the Flapper's angular velocity. However, this can be expected because the Flapper's body is semi-flexible, so the motion capture markers do not move rigidly with respect to each other.

Discussion

Artificial Microsaccade Compensation produces the first stable, smooth videos from a tailless ornithopter. The algorithm runs in real time and relies on a carefully selected combination of video stabilization techniques that are suited for the platform. Further, it works with an inexpensive rolling shutter camera. Next, we contrast this specially designed stabilization algorithm with existing work on video stabilization.

Video stabilization is a well studied topic due to its numerous applications. Several distinct techniques have evolved (*20–22*). Widely known techniques attempt to estimate the scene geometry and the camera's motion, then re-render the video to look as if it came from a camera that smoothly rotates and translates (*23*). Another popular technique is subspace methods, which split the image into many distinct parts that are stabilized separately, with some constraints on smoothness (*24*). Many methods result in gaps between separately stabilized regions of the image, which has led to the study of inpainting techniques that automatically fill those gaps (*25, 26*). The predominant techniques rely heavily on feature matching, which requires significant effort to ensure feature trajectories are long enough and smooth enough to achieve satisfactory results (*27, 28*). More recently, methods that incorporate the aforementioned concepts, but are based on deep learning and advanced representations such as Gaussian Splatting,

12

have achieved impressive results but are typically far from achieving real-time performance and require significant training data (*22, 29*).

Robotic applications require real-time, robust performance, and it is well known that feature matching across rapidly moving frames can fail because of motion blur. Further, our emphasis on microsaccade like motions means any estimated image motion should be constrained to that of a 3D rotation, which is difficult to enforce on feature trajectories. Stabilization based on 3D rotation and known intrinsics has been previously studied, but feature matching is still assumed (*30*). Optical image stabilization based on angular velocity sensors have also been studied for some time (*31*). However, applications typically focus on hand stabilization, which is relatively low bandwidth ($<$10 Hz) (*32*). Deep learning based variants of angular velocity sensor based stabilization have also appeared (*33*). In contrast, our desired application of stabilizing the video from a tailless ornithopter involves a base frequency of 12-20 Hz, which is augmented with higher order harmonics. Additionally, we find that the IMU on our tailless ornithopter does not accurately estimate the system's angular velocity during flight. Lastly, using gyroscopic sensors introduces additional complications, such as time synchronization and calibration of the gyroscopic sensor, which are often not well handled by low-cost hardware.

There are many directions for future work. Alternative methods for fusing frames, without reducing sharpness are of particular interest. Additionally, incorporating automatic identification of camera intrinsics would eliminate the need for prior calibration (*34*). Further, while the Flapper's IMU was not helpful, it is of interest to more carefully study the use of IMU's for rough compensation during aggressive shaking. Finally, applications to robot dogs and humanoids are of interest because they also shake when walking and running.

Determining the exact reason for the inaccuracy of the Flapper's IMU is the subject of future work. However, there are several possible explanations. The IMU is not mounted directly to the camera, instead it is on a nearby circuit board. Thus, the vibrations and semi-flexible body

of the Flapper may cause the camera and IMU to move differently. Additionally, the Flapper's IMU may not be configured appropriately at the hardware level for measuring the Flapper's vibration patterns. Finally, it may be necessary to record the IMU at a rate higher than the maximum allowed by the Flapper's firmware, which is 100 Hz. Regardless, the current results are acceptable for use in applications.

Methods

The algorithm has four parts. First, we describe the direct method for aligning frames, which is an inverse Lucas-Kanade method that assumes a rotational warp function and known image intrinsics. Next, we describe the computation of stable view parameters via filtering on the group of rotations $SO(3)$. Next, we describe the stabilization procedure, which uses a buffer of previous frames and the computed orientations to suppress high-frequency rotations in the image. Finally, a variant based on purposeful saccades is described which results in an efficient recursive update. A block diagram is given in Figure 4 while Figure 5 provides a more detailed overview.

Rotation Estimation

The Flapper has a mean flapping frequency of about 12 Hz, which causes the body and camera to oscillate with both translation and rotation at a similar frequency. However, the translational motion's contribution to optical flow is small in many applications. Thus, we propose ignoring the translational component of optical flow and only estimating the rotation under the assumption that the two frames are closely spaced in time. Formally, we assume the projection equations

$$p_c = \frac{1}{X_c^z} \begin{bmatrix} K & 0 \end{bmatrix} T_{c,w} X_w, \tag{2}$$
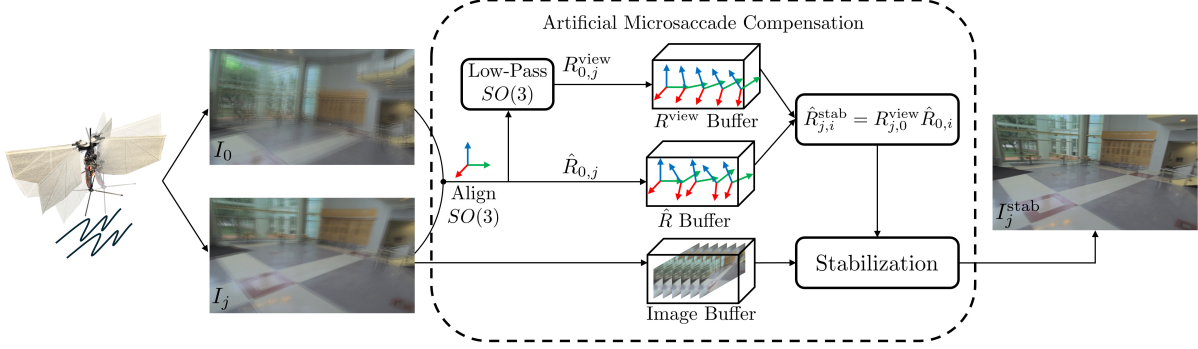
Figure 4: Artificial Microsaccade Compensation allows stabilizing unstable videos, such as those captured by the Flapper Nimble+, by directly matching incoming frames to a periodically updated template frame. Under the assumption of small rotational disturbances, a direct optimization of the mean squared error between images is used to continuously update an orientation estimate $\hat{R}$. Subsequently, a smoothed viewpoint R^{view} is computed. The simplest option for the stable viewpoint is a constant orientation. However, a more sophisticated approach that smoothly tracks the system's viewpoint can be realized with a low-pass filter on the group of rotation matrices $SO(3)$. The frames and rotations are buffered and used in the Artificial Microsaccade Compensation process, which combines multiple frames, taken at multiple times, to realize a high quality and stable video that is free from distortion.

where p_c is the pixel corresponding to the projection of point X_w in the world frame into the camera frame, X_c^z is the depth at pixel p_c, $T_{c,w}$ is the $SE(3)$ transform from world to camera frame, and K is an invertible camera intrinsics matrix. In practice, the intrinsics of the fisheye lens used on the Flapper can be estimated beforehand. Thus, for the rest of the analysis, it is sufficient to assume $K = I$.

Then (2) can be expanded to consider the rotational and translation components of $T_{c,w}$ as $R_{c,w}$ and $t_{c,w}$ respectively,

$$p_c = \frac{1}{X_c^z} \left(R_{cw} X_w + t_{c,w} \right). \tag{3}$$

If it is assumed that $t_{c,w}$ can be considered zero, then the projection can be manipulated to reveal a relationship that does not depend on depth. Let $r_{c,w}^3$ be the third row of $R_{c,w}$ so then $X_c^z = r_{c,w}^3 X_w$. Also, define $p_w = X_w / X_w^z$, that is, the projection of world point X_w into the
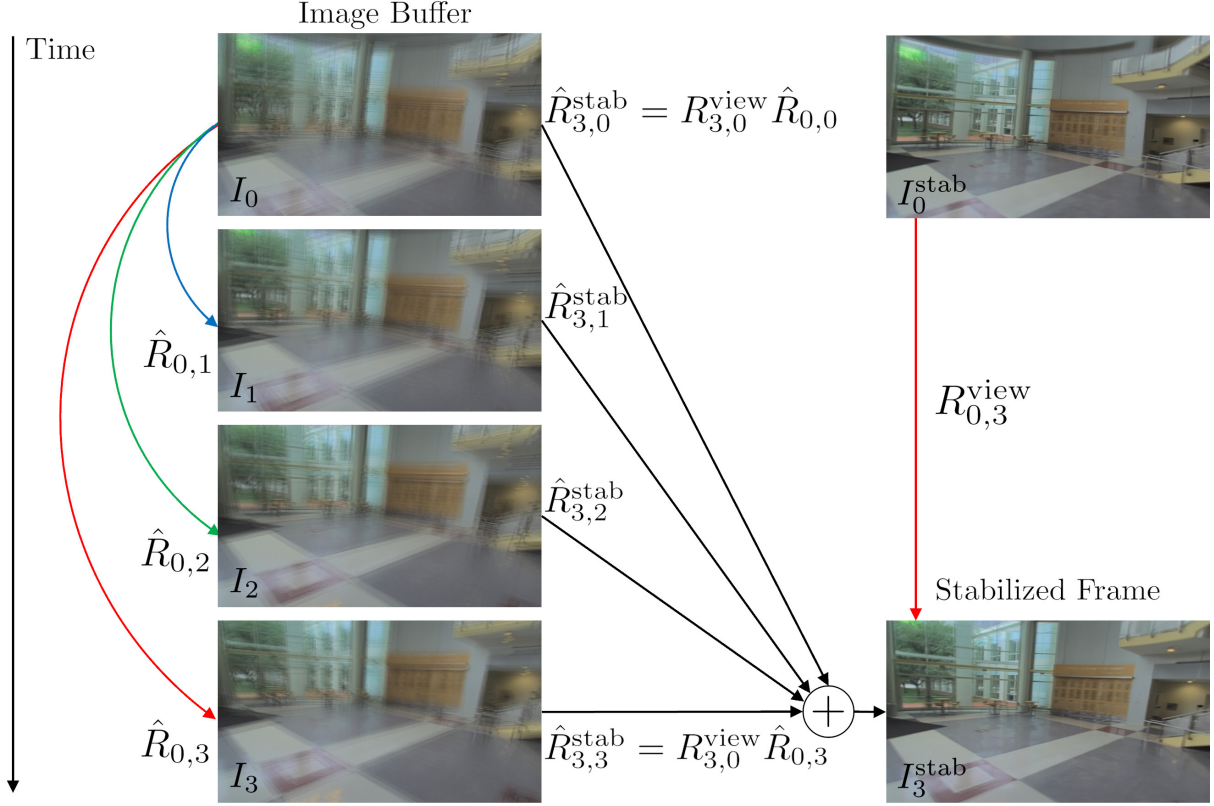
15

Figure 5: A detailed view of the "Stabilization" block in Figure 4. The left column is the average of 6 consecutive frames. While the individual frames are sharp, this averaging results in blur, which illustrates that the camera is shaking aggressively. The right column consists of the same six frames after stabilization and then averaging. In this paper, multiple frames are warped to the current stabilized view and averaged to reduce the effects of rolling shutter and transmission errors of a wireless camera onboard the Flapper. All displayed frames were computed using the camera's full resolution. Each timestep is associated with estimated orientations, $\hat{R}$ computed via a Lucas-Kanade tracker on the group of rotation matrices, $SO(3)$, and R^{view} a list of stable orientations computed by low-pass filtering $\hat{R}$. The stabilized output frame at the current time is computed by warping current and previous frames to the current stabilized viewpoint using stabilizing rotations $\hat{R}_{i,j}^{\text{stab}} = R_{i,0}^{\text{view}} \hat{R}_{0,j}$.

image when the camera is at and aligned with the origin. Then

$$p_c = \frac{R_{c,w}X_w}{r^3_{c,w}X_w} = \frac{R_{cw}X_w/X^z_w}{r^3_{c,w}X_w/X^z_w} = \frac{R_{c,w}p_w}{r^3_{cw}p_w}. \tag{4}$$

Here we see that a world point's projection into the image after a rotation depends only on its starting pixel projection and that rotation. Further, this relationship is easily inverted. Then, if we define two camera frames c_k and c_j we find

$$
\begin{aligned}
p_{c_k} &= \frac{R_{c_k,w}p_w}{r^3_{c_k,w}p_w} \\
p_{c_j} &= \frac{R_{c_j,w}p_w}{r^3_{c_j,w}p_w} \\
\implies p_{c_k} &= \frac{R_{c_k,w}\left(r^3_{c_j,w}p_w\right)R_{w,c_j}p_{c_j}}{r^3_{c_k,w}\left(r^3_{c_j w}p_w\right)R_{w,c_j}p_{c_j}} \\
&= \frac{R_{c_k,c_j}p_{c_j}}{r^3_{c_k,c_j}p_{c_j}}.
\end{aligned} \tag{5}
$$

Thus, the correspondences between the pixel coordinates in frame c_k and c_j are obtained using just the rotation R_{c_k,c_j}. Notably, this works only because translation was neglected. Otherwise, an estimate or model of depth would be required to compute p_k from p_j. It is worth noting that homographies, also known as perspective warps, are commonly used to stabilize video feeds and generate panoramic images. However, homographies correspond to a planar depth assumption, which is well known to result in visible distortions when the scene is not planar. Thus, rotation is the only stabilizing effect that can be applied to an image without introducing distortion via erroneous assumption or estimation of depth.

In what follows, it can be assumed that all considered coordinate frames are a camera frame at some time. Thus, for brevity, the notation $R_{k,j}$ will be used to indicate the rotation from frame c_j to c_k. Additionally, p_k and p_j will represent corresponding pixels in frames k and j taken

from coordinate frames c_k and c_j. Further, all pixel coordinates are considered homogenous so that $(R_{k,j}p_j)/(r_{k,j}p_j)$ can be written as $R_{k,j}p_j$. That is, the division by the third element of $R_{k,j}p_j$ is assumed.

Then, we can set up an optimization problem to estimate $R_{k,j}$ from I_k and I_j by warping I_j so that the resulting pixel intensities are equal to those in I_k. The resulting problem is

$$\hat{R}_{k,j} = \underset{R_{k,j}\in SO(3)}{\text{argmin}} \sum_{p_k\in I_k} \left[I_k(p_k) - I_j(R_{j,k}p_k) \right]^2. \tag{6}$$

We solve this problem using an inverse compositional Lucas-Kanade method (*16*) specialized for a rotational warp given known camera intrinsics K. The inverse compositional formulation iterates over the following two steps

$$\omega_{k,k-1} = \underset{\omega\in so(3)}{\text{argGN}} \sum_{p_k\in I_k} \left[I_k\big(\exp(\omega)\, p_k \big) - I_j\big(R_{j,k-1}p_k \big) \right]^2$$
$$R_{j,k} = R_{j,k-1} \exp(\omega_{k-1,k}) \tag{7}$$

where argGN returns the result of a single step of a Gauss-Newton minimization and $R_{j,k}$ is used as $R_{j,k-1}$ in the next iteration. $\exp$ denotes the matrix exponential, which maps elements of $so(3)$ to $SO(3)$. The inverse compositional formulation is essential because each iteration can reuse precomputed gradients of I_k and the derivative of the rotational warp parameterized by the $so(3)$ Lie Algebra which determine the approximate Hessian used by the Gauss-Newton minimization. These computations would otherwise be the most expensive computation of each iteration. To encourage stable tracking, we also incorporate a line search after each iteration. This guarantees that the loss decreases with each iteration, and prevents the estimate from diverging.

While the Lucas-Kanade method is well known, implementations of this variation are not widely available. Instead, most implementations consider the limited case of purely translational warps with coarse-to-fine refinement, as needed for tracking points. Additionally, some

18

student implementations based on affine homographies are available, presumably because they are described in detail in (*16*). However, as previously noted, affine homographies and full homographies correspond to an incorrect assumption on depth. Further, they require estimating more parameters (six or eight, respectively), while our method requires only three. We include our implementation, which is based on Jax, with this paper's open-source code and the supplementary information.

It remains to track rotational changes across multiple frames. To do so, we propose the procedure described in Algorithm 1. In brief, the procedure estimates rotational deltas, $\hat{R}_{k,j}$, using the current frame I_j and a fixed template frame I_k using the previously described Lucas-Kanade method. Each optimization is warmstarted with the previous rotation estimate $\hat{R}_{k,j-1}$. The first template frame is the first frame I_0. After N_{track} frames, it is assumed that I_k should no longer be compared directly with the current frame I_j because of the compounding effects of the unmodeled translation. At such times, the template frame I_k is updated to be the current frame. Additionally, an estimate of the total rotation from time 0 to the new time k, called $\hat{R}_{0,k}$, is updated using the value of $\hat{R}_{k,j}$ at the time of the template reset. The process repeats indefinitely.

Reusing the template frame I_k for several iterations has several advantages. It limits drift in the cumulative orientation estimate because several new frames are compared to the same reference frame. Additionally, the expensive computation of the problem's gradients and approximate Hessian can be further postponed. While many criteria can be proposed to determine when to reset the template, such as the value of the residuals or the magnitude of the total rotation, we take the pragmatic approach of resetting every 5 frames.

Algorithm 1 Artificial Microsaccade Compensation

Require: $N_{\text{track}}, N_{\text{avg}} \geq 0$

  $I_k \leftarrow \text{get\_frame}()$

  $\hat{R}_{0,k}, \hat{R}_{k,j}, R_{0,j}^{\text{view}} \leftarrow I$

  $B_I, B_{\hat{R}}, B_{R^{\text{view}}} \leftarrow [\quad]$

  $j \leftarrow 0$

  while True **do**

  /* Update Orientation Estimate */

     $I_j \leftarrow \text{get\_frame}()$

     $\hat{R}_{k,j} \leftarrow \text{LK\_inv\_comp}(I_k, I_j, \hat{R}_{k,j})$

     $\hat{R}_{0,j} \leftarrow \hat{R}_{0,k} \hat{R}_{k,j}$

  /* Update Stabilized View */

     $\omega \leftarrow \log\left(R_{j,0}^{\text{view}} \hat{R}_{0,j}\right)$

     $R_{0,j}^{\text{view}} \leftarrow R_{0,j}^{\text{view}} \exp\left(\alpha\left(\|\omega\|\right)\omega\right)$

  /* Append to Buffers */

     $B_I \leftarrow \text{append}(B_I, I_j)$

     $B_{\hat{R}} \leftarrow \text{append}(B_{\hat{R}}, \hat{R})$

     $B_{R^{\text{view}}} \leftarrow \text{append}(B_{R^{\text{view}}}, R^{\text{view}})$

  /* Periodically Reset Template Image */

     if $j \mod N_{\text{track}} = N_{\text{track}} - 1$ **then**

        $I_k \leftarrow I_j$

        $\hat{R}_{0,k} \leftarrow \hat{R}_{0,j}$

        $\hat{R}_{k,j} \leftarrow I$

     end if

  /* Compute Stabilized Frame */

     $B_{I^{\text{stab}}} \leftarrow [\quad]$

     for $i \in \{j - N_{\text{avg}}, \dots, j\}$ **do**

        $\hat{R}_{j,i}^{\text{stab}} \leftarrow B_{R^{\text{view}}}[j]^T B_{\hat{R}}[i]$

        $I_{j,i}^{\text{stab}} \leftarrow I_j^{\text{stab}} + \text{warp}(B_I[i], \hat{R}_{j,i}^{\text{stab}})$

        $B_{I^{\text{stab}}} \leftarrow \text{append}(B_{I^{\text{stab}}}, I_{j,i}^{\text{stab}})$

     end for

     $I_j^{\text{stab}} \leftarrow \text{mean}\left(B_{I^{\text{stab}}}\right)$

     $j \leftarrow j + 1$

  end while

Computation of a Stable View

Next, we consider the estimation of a stable viewpoint. That is, a coordinate frame that follows the estimated orientation but does not shake. In unconstrained problems, a standard approach would be to use a low pass filter of the estimated parameters. However, in this case, the estimated parameters are on $SO(3)$ and so a low-pass filter on $SO(3)$ is needed. For completeness, we describe our methodology, but note that a similar approach appeared in (*30*).

Let $R_{0,j}^{\text{view}}$ be the chosen stable viewpoint and recall that $\hat{R}_{0,j}$ is the estimate of the current cumulative orientation. Then the rotation from the current orientation to the stable viewpoint is

$$R_{j,j}^{\text{stab}} = R_{j,0}^{\text{view}} \hat{R}_{0,j} \tag{8}$$

because, by definition,

$$R_{0,j}^{\text{view}} R_{j,j}^{\text{stab}} = \hat{R}_{0,j}. \tag{9}$$

If $R_{0,j}^{\text{view}}$ follows $\hat{R}_{0,j}$, then $R_{j,j}^{\text{stab}}$ will be close to I. Thus, it is safe to assume $w_{j,j}^{\text{stab}} = \log(R_{j,j}^{\text{stab}}) \in so(3)$ exists and represents the geodesic direction and magnitude from $R_{0,j}^{\text{view}}$ to $\hat{R}_{0,j}$. This geodesic can then be used to interpolate between the two rotations according to

$$R^{\text{interp}}(\beta) = R_{0,j}^{\text{view}} \exp(\beta w_{j,j}^{\text{stab}}), \quad \beta \in [0,1]. \tag{10}$$

By definition, $R^{\text{interp}}(0) = R_{0,j}^{\text{view}}$ and $R^{\text{interp}}(1) = \hat{R}_{0,j}$. Suppose, $\beta \in (0,1)$, then a low pass filter on $SO(3)$ can be realized via the recursion

$$R_{0,j+1}^{\text{view}} = R_{0,j}^{\text{view}} \exp(\beta w_{j,j}^{\text{stab}}). \tag{11}$$

This filter is an analogue of a linear exponential filter. Such a filter may not converge fast enough to keep the stabilized view close to the true field of view, especially when the system

is rotating quickly. We can improve the convergence at the expense of increasing the rotational noise by considering the magnitude of the rotation delta between the view and orientation estimate. Consider a monotonically increasing function $\alpha : \mathbb{R} \to \mathbb{R}$. Then the formulation can be generalized as

$$R_{0,j+1}^{\text{view}} = R_{0,j}^{\text{view}} \exp\left(\alpha\left(\|\omega_{j,j}^{\text{stab}}\|\right) \omega_{j,j}^{\text{stab}}\right) \tag{12}$$

In practice, we find the $\alpha(\|\omega\|) = a + b\|\omega\|$ with $a = 2\Delta t$, $b = 40\Delta t$, where Δt is the interframe period provides reasonable results.

Stabilization

At this point, the orientation of each frame has been estimated, $\hat{R}_{0,j}$ and a method for computing a stable viewpoint at each time, $R_{0,j}^{\text{view}}$, has been proposed. It remains to estimate a stabilized video. Consider the rotation delta going from a frame's estimated orientation at time i to any stable viewpoint at time j, we call this $R_{j,i}^{\text{stab}}$, and it can be computed via

$$R_{j,i}^{\text{stab}} = R_{j,0}^{\text{view}} \hat{R}_{0,i}. \tag{13}$$

With this rotation estimate, we can warp any previous frame so that it appears to have been taken from the orientation of any stabilized viewpoint. We refer to such a rotation as a stabilizing rotation. The stabilized frames can be computed via

$$I_{j,i}^{\text{stab}}(p_j) = I_i(R_{i,j}^{\text{stab}} p_j). \tag{14}$$

If several such frames are computed for a particular time j, they can be combined, depending on the application, to produce a higher-quality stabilized frame. Specifically, in this paper, the stabilized frames are produced by averaging the last $N_{avg} = 6$ frames after aligning them to the

current stabilized view. This reduces rolling shutter effects and transmission artifacts introduced by the wireless camera mounted on the flapping robot. That is, the final stabilized image is given by

$$I_j^{\text{stab}} = \frac{1}{N_{avg}} \sum_{i=j-N_{avg}}^{j} I_{j,i}^{\text{stab}} \tag{15}$$

Specialization to Saccades

The previously described stabilization method is useful for human viewing because the stabilized frames appear as if they were captured by a camera rotating smoothly. However, it is also possible to remove all rotation from an image for short periods of time by forcing $R_{0,j}^{\text{stab}}$ to be constant until it needs to be changed to maintain overlap between the stabilized and true field of views. This results in the least possible apparent image motion because the stabilized frames become as identical as possible, until the static orientation is updated. Further, the stabilized frame can be computed via a recursion which saves significant computation. Consider, if $R_{0,j}^{\text{stab}} = R_{0,j+1}^{\text{stab}}$ then we can drop the subscripts from these terms, because they are identical, resulting in the recursion

$$
\begin{aligned}
I_{j+1}^{\text{stab}}(p_j) &= \frac{1}{N_{avg}} \sum_{i=j-N_{avg}+1}^{j+1} I_i(\hat{R}_{i,0} R^{\text{view}} p_j) \\
&= \frac{1}{N_{avg}} \Big[I_j^{\text{stab}}(p_j) + I_{j+1}(\hat{R}_{i,0} R^{\text{view}} p_j) \\
&\qquad\qquad - I_{j-N_{avg}}(\hat{R}_{i,0} R^{\text{view}} p_j) \Big].
\end{aligned}
\tag{16}
$$

This expression adds and subtracts a frame from the previous stabilized frame to the get the next stabilized frame. In comparison, Equation 15 requires summing $N_a vg$ frames to produce a stabilized frame, which results in substantially more computation if $N_{avg} > 2$.

This recursion will produce suitable results until the true orientation of the camera rotates such that the true camera's field of view has little overlap with the stabilized image's field of view. At such times, the stabilized view must be shifted to align better with $\hat{R}_{0,j}$, similar to a saccade in human vision. For the purpose of study, when running in this recursive "saccade" mode, we reset R^{stab} to $\hat{R}_{0,j}$ when less than 90% of I^{stab}'s pixels are filled by each of the averaged frames.

Hardware

The algorithm was tested on a Flapper Nimble+ manufactured by Flapper Drones. This tailless ornithopter beats its wings between approximately 12 and 20 Hz. The lack of a tail makes the platform agile, but also contributes to aggressive shaking. Our Flapper is equipped with a wireless HDZero Nano V3 FPV rolling shutter camera running at $1280{\times}720$ resolution and 60 fps. The camera is popular among racing drone pilots.

Data Availability

All data needed to interpret the conclusions of the paper are presented in the paper or supplementary information. Raw datasets generated and analyzed during the study are available at `prg.cs.umd.edu/AMC`.

Code Availability

Code for the stabilization algorithm and analysis of raw data is available in the supplementary information and at `prg.cs.umd.edu/AMC`.

Acknowledgments

Author Contributions

L.B. developed the algorithm, designed the study, conducted the experiments, and wrote the manuscript. G.C. helped designed the study, connected the results to the existing literature, and helped write the manuscript. Y.A. helped develop the algorithm and write the manuscript. All authors read and approved the final manuscript.

Competing Interests

The authors declare no competing interests.

References

1. de Croon, G. Flapping wing drones show off their skills. *Science Robotics* **5**, eabd0233 (2020).

2. Park, J. H. & Yoon, K.-J. Designing a biomimetic ornithopter capable of sustained and controlled flight. *J. Bionic Eng.* **5**, 39–47 (2008).

3. De Croon, G., De Clercq, K., Ruijsink, R., Remes, B. & De Wagter, C. Design, aerodynamics, and vision-based control of the delfly. *International Journal of Micro Air Vehicles* **1**, 71–97 (2009).

4. Nian, P., Song, B., Xuan, J., Zhou, W. & Xue, D. Study on flexible flapping wings with three dimensional asymmetric passive deformation in a flapping cycle. *Aerosp. Sci. Technol.* **104**, 105944 (2020).

5. Phan, H. V. & Park, H. C. Insect-inspired, tailless, hover-capable flapping-wing robots: Recent progress, challenges, and future directions. *Prog. Aerosp. Sci.* **111**, 100573 (2019).

6. Karásek, M., Muijres, F. T., De Wagter, C., Remes, B. D. & De Croon, G. C. A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science* **361**, 1089–1094 (2018).

7. Tapia, R. *et al.* A comparison between framed-based and event-based cameras for flapping-wing robot perception. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 3025–3032 (2023).

8. Tapia, R. *et al.* Flight of the future: An experimental analysis of event-based vision for online perception onboard flapping-wing robots. *Advanced Intelligent Systems* 2401065 (2025).

9. Rafee Nekoo, S. *et al.* A review on flapping-wing robots: Recent progress and challenges. *The International Journal of Robotics Research* (2025).

10. Lau, G.-K., Chin, Y.-W., Lin, S.-C., Lai, Y.-H. & Khoo, B. C. Spring and power in hovering ornithopters. *Advanced Intelligent Systems* **7**, 2400477 (2025).

11. Olejnik, D. A. *et al.* A tailless flapping wing mav performing monocular visual servoing tasks. *Unmanned Systems* **08**, 287–294 (2020).

12. Pan, E., Liang, X. & Xu, W. Development of vision stabilizing system for a large-scale flapping-wing robotic bird. *IEEE Sensors Journal* **20**, 8017–8028 (2020).

13. De Wagter, C., Tijmons, S., Remes, B. D. & de Croon, G. C. Autonomous flight of a 20-gram flapping wing mav with a 4-gram onboard stereo vision system. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 4982–4987 (IEEE, 2014).

14. Rodriguez-Gomez, J. P., Martínez-de Dios, J., Ollero, A. & Gallego, G. On the benefits of visual stabilization for frame-and event-based perception. *IEEE Robotics and Automation Letters* (2024).

15. Poletti, M. & Rucci, M. A compact field guide to the study of microsaccades: Challenges and functions. *Vision research* **118**, 83–97 (2016).

16. Baker, S. & Matthews, I. Lucas-Kanade 20 years on: A unifying framework. *International Journal of Computer Vision* **56**, 221–255 (2004).

17. Lucas, B. D. & Kanade, T. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, vol. 2, 674–679 (1981).

18. Adobe Premiere Pro video stabilization tutorial. `https://www.adobe.com/creativecloud/video/discover/stabilize-video.html`. Accessed: 2025-08-28.

19. Bradbury, J. *et al.* JAX: composable transformations of Python+NumPy programs (2018). URL `http://github.com/jax-ml/jax`.

20. Wang, Y. *et al.* Video stabilization: A comprehensive survey. *Neurocomputing* **516**, 205–230 (2023).

21. Roberto e Souza, M., Maia, H. d. A. & Pedrini, H. Survey on digital video stabilization: Concepts, methods, and challenges. *ACM Computing Surveys (CSUR)* **55**, 1–37 (2022).

22. Guilluy, W., Oudre, L. & Beghdadi, A. Video stabilization: Overview, challenges and perspectives. *Signal Processing: Image Communication* **90**, 116015 (2021).

23. Liu, F., Gleicher, M., Jin, H. & Agarwala, A. Content-preserving warps for 3D video stabilization. *ACM Trans. Graph.* **28** (2009).

24. Liu, F., Gleicher, M., Wang, J., Jin, H. & Agarwala, A. Subspace video stabilization. *ACM Trans. Graph.* **30** (2011).

25. Matsushita, Y., Ofek, E., Tang, X. & Shum, H.-Y. Full-frame video stabilization. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 50–57 vol. 1 (2005).

26. Matsushita, Y., Ofek, E., Ge, W., Tang, X. & Shum, H.-Y. Full-frame video stabilization with motion inpainting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**, 1150–1163 (2006).

27. Lee, K.-Y., Chuang, Y.-Y., Chen, B.-Y. & Ouhyoung, M. Video stabilization using robust feature trajectories. In *2009 IEEE 12th International Conference on Computer Vision*, 1397–1404 (2009).

28. Wang, Y.-S., Liu, F., Hsu, P.-S. & Lee, T.-Y. Spatially and temporally optimized video stabilization. *IEEE Transactions on Visualization and Computer Graphics* **19**, 1354–1361 (2013).

29. You, Z., Georgoulis, S., Chen, A., Tang, S. & Dai, D. GaVS: 3d-grounded video stabilization via temporally-consistent local reconstruction and rendering. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers* (2025).

30. Jia, C., Sinno, Z. & Evans, B. L. Real-time 3D rotation smoothing for video stabilization. In *2014 48th Asilomar Conference on Signals, Systems and Computers*, 673–677 (2014).

31. Sato, K., Ishizuka, S., Nikami, A. & Sato, M. Control techniques for optical image stabilizing system. *IEEE Transactions on Consumer Electronics* **39**, 461–466 (1993).

32. La Rosa, F., Virzì, M. C., Bonaccorso, F. & Branciforte, M. Optical image stabilization (OIS). *STMicroelectronics. Available online: http://www. st. com/resource/en/white_paper/ois_white_paper. pdf (accessed on 12 October 2017)* (2015).

33. Liu, S. *et al.* DeepOIS: Gyroscope-guided deep optical image stabilizer compensation. *IEEE Transactions on Circuits and Systems for Video Technology* **32**, 2856–2867 (2022).

34. Hartley, R. I. Self-calibration from multiple views with a rotating camera. In Eklundh, J.-O. (ed.) *Computer Vision — ECCV '94*, 471–478 (Springer Berlin Heidelberg, Berlin, Heidelberg, 1994).