2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Adaptive Sampling-based Particle Filter for Visual-inertial Gimbal in the Wild

Xueyang Kang[1], Ariel Herrera[2], Henry Lema[2], Esteban Valencia[2], Patrick Vandewalle[1]

Abstract

In this paper, we present a Computer Vision (CV) based tracking and fusion algorithm, dedicated to a 3D printed gimbal system on drones flying in nature. The whole gimbal system can stabilize the camera orientation robustly in challenging environments by using skyline and ground plane as references. Our main contributions are the following: a) a light-weight Resnet-18 backbone network model was trained from scratch, and deployed onto the Jetson Nano platform to segment the image specifically into binary parts (ground and sky); b) our geometry assumption from the skyline and ground cues delivers the potential for robust visual tracking in the wild by using the skyline and ground plane as references; c) a manifold surface-based adaptive particle sampling can fuse orientation from multiple sensor sources flexibly. The whole algorithm pipeline is tested on our 3D-printed gimbal module with Jetson Nano. The experiments were performed on top of a building in a real landscape. The public code link: https://github.com/alexandor91/gimbal-fusion.git.

I. Introduction

Gimbal platforms have been widely used in photogrammetry and robot perceptual modules to stabilize the camera pose, thereby improving the captured video quality. Usually, a gimbal is mainly composed of sensors and actuators. The sensor's orientation measurements can be utilized directly by an actuator to steer the camera toward the desired pose. However, off-the-shelf custom gimbals are either quite expensive or depend on a high-precision IMU or a brushless DC motor with a Hall sensor to read angles, which is prone to noise drift over long-term operation. The goal of our project is to deploy a gimbal platform carried by a UAV for volcanic eruption surveillance in an unpopulated region full of mountains. Hence, our work differs from prior work in that a robust platform should be devised, in operation over a long term and with endurance in the challenging wild. The main contribution of this paper is threefold.

- A lightweight binary segmentation model is trained to label the ground and sky pixels specifically, aiming for real-time inference on the embedded device.
- Natural cues in the challenging mountainous region, such as the ground plane and skyline isolated from the aforementioned binary mask, are utilized as references for the gimbal stabilization to infer the rotation angles.
- To fuse the roll and pitch rotation angles from multiple sensor modalities, i.e., IMU and CV pipeline, a non-linear particle filter at varying resolutions over the manifold surface is proposed, and it is implemented on the Jetson Nano board with a demo test.

*This work was supported by VLIR-UOS project under agreement No. EC2020SIN278A101.

[1] The authors are with PSI Department of Electrical Engineering (ESAT), KU Leuven, Belgium. Email: alex.kang@kuleuven.be

[2] The authors are with Department of Mechanical Engineering, Escuela Politecnica Nacional (EPN), Ecuador

II. Related work

The problem of video stabilization can be traced back to the Electronic Image Stabilization (EIS) technique [29], which relied on handcrafted feature points in consecutive image frames to search for correspondences for image alignment. Feature point-based tracking [26], or optical flow tracking [31], required distinctive feature points or consistent raw pixel intensity distributions across video frames as shown in the work by Kulkarni et al. [22]. Thus, it is sensitive to pixel noise induced by motion blur. On the other hand, the humanoid robot vision system [34], or complex video task [35] compensated for the rotation and shift of video images by using orientation from IMU directly. A good review of traditional motion prediction work applied to video stabilization can be found in a study by Rawat et al. [30].

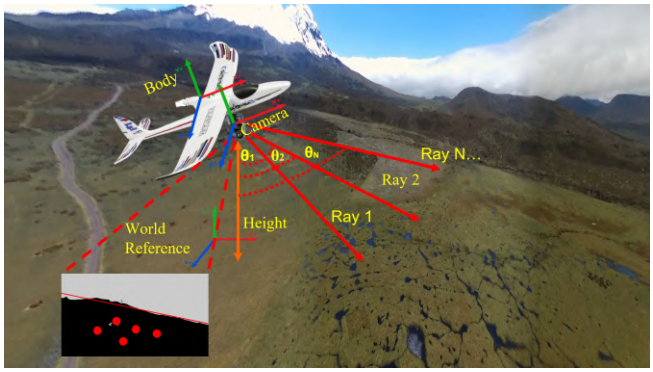


Fig. 1: Demo of gimbal platform on the fixed-wing airplane.

Video stabilization can also be considered as an image deblurring task. The main goal is to infer the pixel intensity in a blurry area from neighboring pixels in single or multiple images, incorporating both temporal and spatial constraints from the video. Recent deep learning techniques boost video processing performance by introducing a prior from a dataset in specific domains, e.g., Jiyang Yu et al. [32] warped the selfie image patch locally by decoding the scene representation from multiple encoder outputs, such as foreground

contours, feature points on the background, and 3D facial mesh. Chen Li et al. [33] trained a motion estimation model by using raw IMU data to predict the visual odometry in a sliding time window, similar to a filter approach.

The video stabilization was integrated into a large mapping and localization framework via graphs in a self-supervised manner by Lee et al. [36]. To tackle the problem of video stabilization in dynamic scenes, a dense warping field as scene representation was trained from consecutive video frames by Liu et al. [38], then the warped parts are blended to synthesize the stabilized image. The framework by J. Choi et al. [37] even made use of a motion prediction model based on optical flow tracking. However, all those models involve a lot of processing overhead and are quite heavy to deploy on an edge device in real time. Video stabilization was also widely applied in aerial surveillance by [39], or UAV attitude stabilization as the work done by Chung-Cheng Chiu et al. [3]. Using natural cues to estimate the orientation of UAVs or fixed-wing airplanes was performed in many real applications [1],[2],[6], all relying on skyline tracking. However, they require delicate tuning to search for a boundary between sky and ground, even input images captured under ideal conditions. Other works depend on the tracking of geometric primitives on 2D images or in 3D, either by discrete feature points [23][25][26], or a curved boundary [5]. Five-point algorithm [9] or curvature alignment can be used to predict the ego-pose of the camera from feature correspondences. Moving objects were detected in the image view through a tracking filter proposed by Ahlem Walha et al. [40], over the "SIFT" feature points [7]. However, in a natural setting, there are a lot of spurious features, such that tracking algorithms relying on feature points may fail.

III. Overview

As shown in Figure 1, the gimbal system is placed underneath an airplane body including a camera, IMU, and barometer. To overcome the aforementioned correspondence issues of feature points, the gimbal system can take advantage of natural cues such as skyline and ground to stabilize the camera pose. The bottom of Figure 1 exhibits a binary mask with a skyline and points in the ground region. Rays passing through the red dots along with the height from the barometer, allow us to determine the 3D position of the ground plane through trigonometry. The hardware and software diagrams are presented.

A. Open source Hardware

The gimbal platform design is based on open-source hardware. The main processing unit is a Jetson Nano, featured with 2GB GPU memory and Quad-core ARM A57, connected to IMU, camera, and barometer sensors. An OpenCR 2.0 driver board maps the driving command to control commands for two servo motors.

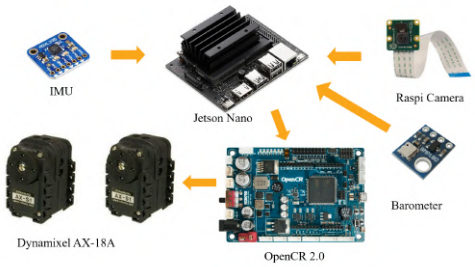


Fig. 2: Open source hardware setup.

B. ROS nodes

The software is composed mainly of three parts: the preprocessing part including the network model and geometric primitive extraction, followed by a tracking module to align the skyline and normal of the ground plane in the current frame with those in the reference frame. The compensation angles from various pipelines are then fed into the proposed particle filter presented in Section V to obtain fusion orientations, further as input for the controller to stabilize the camera.

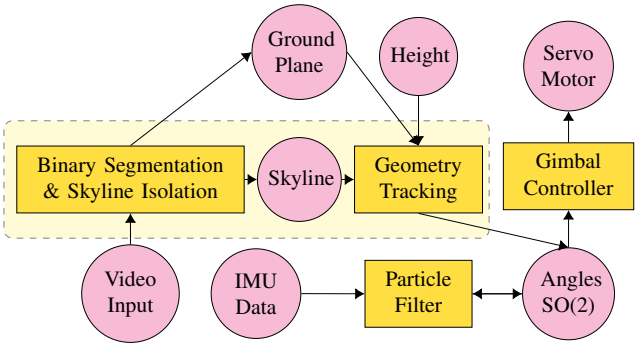


Fig. 3: Block diagram of the presented algorithm. Circular nodes in pink are signals or controlled targets. Rectangular boxes in yellow are ROS nodes for the algorithm, the dashed region is the front-end perception part, including tracking of skyline and ground plane.

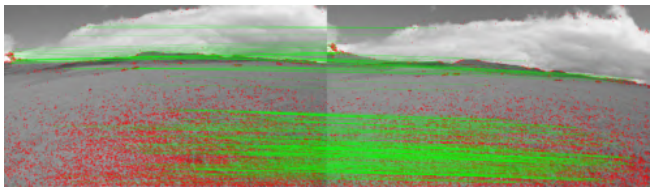
IV. Perception

The perception part is structured into preprocessing and rotation estimation, where the rotation estimation can be further separated into two pipelines: roll and pitch prediction from skyline tracking and rotation estimation from ground plane tracking. The following subsections follow this structure.

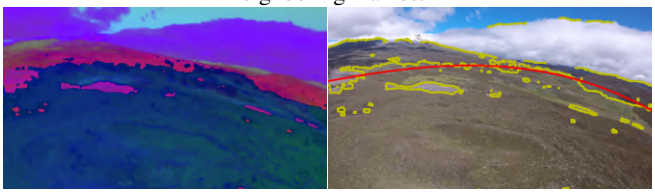
A. Preprocessing

We first tried the general computer vision processing pipelines, but they all failed due to spurious feature candidates. As illustrated in Figure 4, the similar appearances in the grassland region and cloudy sky all pose a great challenge to correspondence search. In Figure 4a, many false correspondences are found. Canny edge detection [8] is applied to find the boundary between the ground and sky region on HSV image, nevertheless, some brightness of the sky is cast onto the grass ground, generating the wrong boundary in Figure 4c. To tackle this challenging segmentation task, we finally choose the data-driven approach, by

training a Resnet-18 [19] network on "Skyfinder" dataset [4] first, followed by a fine-tuning on the self-collected dataset with one hundred images. Binary cross entropy loss is applied for pre-training through 100 epochs, and fine-tuning with 30 epochs respectively. The total training time is less than two hours. Some training data samples along with ground truth masks are presented in Figure 5. The model is exported into "ONNX" and optimized by "TensorRT" to convert to "FP16" precision for Jetson Nano deployment. We found the model can achieve above 90% success rate for the segmentation on average, only when under some extreme cases like overexposure, the failure may happen. Additionally, our use case is for mountainous terrain, the scenario with mirror effects and reflections by water on the ground is not advisable, due to the similar color distributions of the sky and the ground.



(a) Correspondences based on "SIFT" [7] feature points of neighboring frames.



(b) HSV image converted from a raw RGB image. (c) Boundary curve-fitting on the detected Canny edges.

Fig. 4: Failure case demo using OpenCV pipeline.

Fig. 5: Sample images and ground truth masks for training.

B. Skyline Tracking

Starting from the binary mask results gained from the network model, the skyline can be extracted along the boundary direction. The extracted skyline can be further considered as a cue to estimate the roll and pitch of the camera, as shown in Figure 6. The boundary points, represented by yellow dots in Figure 6c, can be implemented firstly over the whole reference image. A straight line is fitted as illustrated in red (Figure 6c), using two parameters, slope m' and intercept b' in Equation 1. For the subsequent images, we use a constant angular velocity model derived from skyline tracking of the previous two frames to predict the skyline position in the current frame, like the green line presented in Figure 6d. The assumption is that the camera remains not upside down, due to the airplane maneuverability constraints. The boundary can always be searched along the vertical direction of the predicted skyline (green). In practice, the skyline points are further down-sampled to speed up the search. Once the current sample points of skyline are attained, the estimated skyline (red) in the current frame can be derived via least-square in Figure 6d.

$$m'x + b' = y' \tag{1}$$

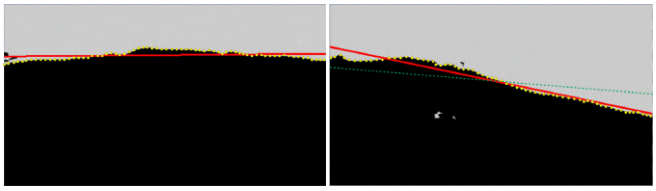
$$mx + b = y \tag{2}$$

$$\alpha = arctan(m) - arctan(m') \tag{3}$$

$$\beta = arctan\left(\frac{h_1 - c_y}{f_y}\right) - arctan\left(\frac{h_2 - c_y}{f_y}\right) \tag{4}$$

The roll α and pitch β angles can be derived from Equations 3, and 4 respectively by subtracting the angles. c_y is half of the image height, and h_1 and h_2 are the height of the center point of skyline in the current image frame and reference frame respectively. f_y is the focal length y of the camera.

(a) Reference image (b) Current image frame



(c) Reference mask with skyline (d) Current mask with skyline

Fig. 6: Segmented images for skyline search.

Roll and pitch have a specific tolerance range to avoid unnecessary operations, so processing is only triggered when the movement is out of this range. Roll angle can be predicted from the slope m of the skyline, followed by pitch estimation, which is on top of the image result after roll compensation. There is a total of three cases in our system, that is pure roll, pure pitch, or both happening simultaneously. Here the height shift resulting from translation is subtracted before rotation processing by barometer readings.

C. Estimation from Ground Plane Tracking

Ground plane tracking relies on the normal vector of the ground plane, as demonstrated in Figure 1. A set of points in the ground region of the binary mask are sampled evenly, followed by a back projection to the camera frame

corresponding to Equations 5-11.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\rho_{\mathbf{i}} = [u, v, 1]^T \tag{6}$$

$$\mathbf{P_i} = \mathbf{K^{-1}}\rho_{\mathbf{i}}, i \in (1...N) \tag{7}$$

$$\mathbf{N_G} = [0, 0, g_z]^T \tag{8}$$

$$\cos\theta = \frac{\mathbf{P_i} \cdot \mathbf{N_G}}{||\mathbf{P_i}|| \cdot ||\mathbf{N_G}||} \tag{9}$$

$$l_i = \frac{h}{\cos\theta} \tag{10}$$

$$\mathbf{P'_i} = l_i \mathbf{P_i} \tag{11}$$

The height h is measured from the barometer, and the ray direction passing through the pixel position is $\mathbf{P_i}$ on the left side of Equation 7. $\mathbf{K}$ is the intrinsic matrix obtained from calibration [17]. θ is derived from the dot product of the gravitational vector and the ray direction. Length scale l_i can be calculated by trigonometry in Equation 10. Finally, the current normal vector $\mathbf{m}$ of the ground plane is shaped by the cross product of points as below:

$$\mathbf{m} = (\mathbf{P'_i} - \mathbf{P'_j}) \times (\mathbf{P'_i} - \mathbf{P'_k}) \tag{12}$$

The ground plane tracing mode is only triggered when the camera is over 300 meters above ground so that the variance of uneven grassland can be approximated by a flat plane compared to the height. Next, the rotation matrix to align the normal vector $\mathbf{m}$ in the current frame and the reference normal $\mathbf{n}$ at the start can be derived as follows.

$$s = \frac{\mathbf{m}}{||\mathbf{m}||} \cdot \frac{\mathbf{n}}{||\mathbf{n}||}, \tag{13}$$

$$\mathbf{k} = \frac{\mathbf{m}}{||\mathbf{m}||} \times \frac{\mathbf{n}}{||\mathbf{n}||}, \tag{14}$$

$$\mathbf{k}_\times = \begin{bmatrix} 0 & -k_3 & k_2 \\ k_3 & 0 & -k_1 \\ -k_2 & k_1 & 0 \end{bmatrix} \tag{15}$$

$$\mathbf{R} = \mathbf{I} + \mathbf{k}_\times + \mathbf{k}_\times^{\mathbf{2}} \frac{1}{1+s} \tag{16}$$

$\mathbf{k}_\times$ is the skew matrix, where non-zero elements are in off-diagonal positions, corresponding to the components of the cross product of $\mathbf{m}$ and $\mathbf{n}$. s is a scale derived from the dot product of two vectors. The rotation matrix is calculated following Rodrigues' rotation formula in Equation 16.

In the end, the 3D Euler angles are retrieved from the rotation matrix according to Equations 17-20, in a right-hand order, "yaw←pitch←roll". Only roll and pitch are used for later fusion.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{17}$$

$$\alpha = arctan(r_{32}, r_{33}) \tag{18}$$

$$\beta = arctan(-r_{31}, \sqrt{r_{32}^2 + r_{33}^2}) \tag{19}$$

$$\gamma = arctan(r_{21}, r_{11}) \tag{20}$$

V. Adaptive resolution based Particle Filter

The Particle Filter is easy to implement and applicable in non-linear problems, in particular for positioning [20]. Here, a variant of the vanilla particle filter [28], sampling on the spherical surface adaptively is proposed in the pseudo-code below. In a real configuration, the roll and pitch are virtually constrained by a limit range due to mechanical kinematics, e.g., roll in a range from -45°to 45°. The general idea

Algorithm 1: Particle Filter on Spherical Surface

Data: $S_I = (\alpha_I, \beta_I), S_{C_{1,2}} = (\alpha_{C_{1,2}}, \beta_{C_{1,2}}), \Omega_{1,2,3}$.
Result: Output $\bar{S}_k = [\alpha,\ \beta]^T$.

1 **if** new S^I available **then**
2 Initialize the particle $s_0^{(j)}$ or sampling $s_k^{(j)}$ (on Ω_1) from $\mathcal{N}(\tilde{\mu}_I | \mu_I + \omega\delta t, (\sigma_I + b))$, $j = 1...N$;
3 **end**
4 **if** new S_{C_1} and S_{C_2} both are available **then**
5 **for** $s_k^{(j)}$ in the particle set of N samples **do**
6 $\omega_k^{(j)} = \omega_{k-1}^{(j)} \exp(s_k^{(j)} - S_{C_{1,2}})$;
7 **if** $||s_k^{(j)}\text{-}S_{C_{1,2}}||_2 \leqslant \varepsilon$ **then**
8 Sampling $\hat{s}_k^{(1...m)}$ from $\mathcal{N}(\tilde{\mu}_f | \mu_f, \delta_f)$ (on Ω_3)
9 initialize new weights: $\omega_k^{(1...m)} = \omega_k^{(j)} \exp(\hat{s}_k^{(1...m)} - \mu_f)$;
10 **end**
11 **end**
12 **end**
13 **if** new S_{C_1} or S_{C_2} is available **then**
14 **for** $s_k^{(j)}$ in N samples (on Ω_2 resolution) **do**
15 Sampling from $\mathcal{N}(\tilde{S}_{C_1} | S_{C_1}, \delta_{C_1})$ or $\mathcal{N}(\tilde{S}_{C_2} | S_{C_2}, \delta_{C_2})$, same as line 6-9;
16 **end**
17 **end**
18 $\hat{s}_k^{(1...m)} \cup S_\Omega$;
19 Resampling $s_k^{(j)}$ according to $\omega_k^{(j)}$;
20 $\bar{S}_k = \sum_{j=1}^{N+m} \omega_k^{(j)} \hat{s}_k^{(j)}$;

behind this adaptive particle filter is straightforward. The filter mainly comprises three steps: sampling from orientation measurements of the IMU s_I (line 2); sampling according to observation from the Computer Vision (CV) pipeline (lines 8 and 15); resampling proportionally to the updated weight of each particle (line 19). The weight in line 6 is derived from a normal distribution, as a function of the square root of the angular distance, which is between sampled cell position and sensor observation on the manifold surface. S_{C_1} and S_{C_2} indicate the orientation estimation from the skyline and the ground plane respectively.

It is noteworthy that all the particle samples are generated on the discretized cells, spreading over the manifold space formed by the roll and pitch angles. Each particle is a 2D vector represented by a cell position on the spherical surface. There are three levels of cell resolution ($\Omega_{1,2,3}$ in Algorithm 1) from coarse to fine, where the longitudinal direction of the spherical surface represents the pitch, while the latitudinal

direction is the roll. In line 2, particles are sampled from a Gaussian distribution $\mathcal{N}$, with a mean value at the IMU measurements plus a shift by a constant angular velocity propagating through a certain interval plus an offset b.

When both observations from the skyline and ground plane are available (line 4), more samples will be created around those cells close to sensor measurements (line 8), and their weights are initialized by multiplication of parent weight and local weight as a function of angular distance (line 9), whereas the other particles' weights will be down-weighed by an aforementioned normal distribution as a function of angular distance. Line 7 manifests the angular distance criteria for neighboring particle cells close to observation. The sample cells meeting the criteria are used as parents to create more children particles around μ_f at line 8 of Algorithm 1, and μ_f is derived from Equation 21, as a weighted sum of results from two Computer Vision pipelines, with each result used as a mean value of the normal distribution. A simple inverse of corresponding variance $\delta_{C_1}, \delta_{C_2}$ respectively is considered as weight.

$$\mu_f = \frac{\mu_{C_1}}{\delta_{C_1}} + \frac{\mu_{C_2}}{\delta_{C_2}} \tag{21}$$

$$\delta_f = \left(\frac{1}{\delta_{C_1}} + \frac{1}{\delta_{C_2}} \right)^{-1} \tag{22}$$

IMU and CV observation variances are set as a constant according to practical tests. μ_f and δ_f are fused results from two CV pipelines in the same weighted sum form of Equations 21. The same strategy repeats when a single CV observation pipeline is present (line 15), but sampling rather on a middle-level resolution.

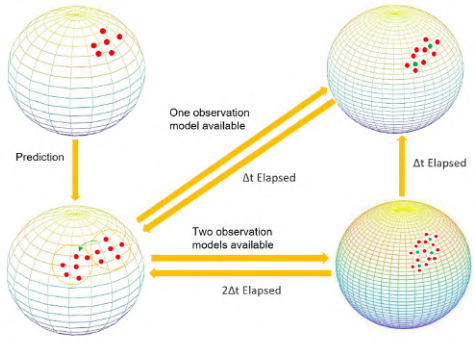


Fig. 7: Lifetime phases of particle filter sampling on a spherical surface.

Each particle's life cycle can be represented in four phases, as shown in Figure 7. Arrows between them stand for transition conditions. In our setting, the Computer Vision based orientation observation has a smaller variance compared to IMU. As aforementioned, three levels of cell resolution are employed. The top left part of Figure 7 represents the initial state, sampled from a normal distribution centered at the measurement of timestamp t. The bottom left is the prediction based on the propagation of the previous particle states by extrapolation in time. When the orientation from a single pipeline, either skyline or ground plane is available, the new samples in red are generated on finer resolution neighboring the green dots corresponding to μ_f in Equation 21. Each dot is located at the center of a cell. If both skyline and ground plane pipelines are available, the highest resolution Ω_3 is employed to generate more new samples. Each particle's timestamp of creation is kept as well. If the lapsed time exceeds a certain interval δt, the particles will be placed back at a coarse resolution, like the arrow direction from the bottom right to the top right. At a certain time point, the particles in the set have various precision. A lifetime check will be called periodically to eliminate the particles existing for a long time.

VI. Experiment & Results

In practice, the video is scaled down to 640×480 resolution to achieve a raw frame rate of 20, while the overall frame rate scales down to 12-15 after the fusion on Jetson Nano. Intrinsics of the camera are acquired following the calibration guide of [17]. Extrinsic calibration between low-cost IMU (BNO055) and Raspi-camera is established using an open source tool "Kalibr" [15][16]. Figure 8 shows the simulation setup on top of the building in the landscape. There are three parts, a motor driver board, Jetson Nano, and the gimbal part. All 3D-printed cases have enhanced connections, taking aerodynamics into account for flying efficiency. The camera on the gimbal is placed to be forward facing the landscape. Then the gimbal cases along with the sensors are attached to a pole end (not in the view of Figure 8). The other end of the pole is controlled manually to simulate random rotation. Here, the ground truth roll and pitch angles are read from the servo motors, and the protractor in the figure is only adopted for verification of the test. In the demo test, we use the fused estimation from our particle filter to steer the motors. Closed-loop PID controller is leveraged for actuation. All the following test sequences were recorded from a static position at the start. Furthermore, it is guaranteed the ground plane should be orthogonal to the gravitational vector at start. This configuration remains unchanged for the real UAV test.

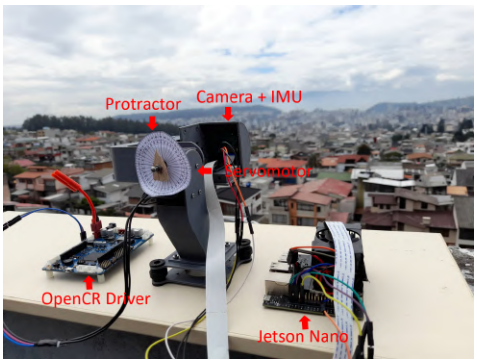


Fig. 8: Gimbal simulation test setup on top of the building.

Open-source datasets for orientation estimation research were mostly overlapping with SLAM research, and the SLAM datasets were often captured indoors or within urban region, rarely including unpopulated areas, viewed from the top. We thus recorded sequences by using the aforementioned

gimbal setup in a real mountain landscape on a tall building roof nearby, which should be quite similar to the camera view on the airplane. During recording, each pose configuration of the gimbal is kept on par with angle readings from motors containing hall sensors as ground truth. For comparisons, the SOTA visual-inertial frameworks "ORBSLAM3" [11], "R-VIO" [13], and "DM-VIO" [14] were selected at first, but we found these algorithms are dedicated to 6-DoF visual odometry and are all relying on feature points extracted from the images, which are not suitable for the challenging feature-less scenes of our datasets, so we compare our fusion algorithm against the IMU filtered by quaternion-based Madgwick, CV only pipelines, like skyline or ground respectively.

	Sequence	IMU (Madgwick Filter)	Skyline only	Ground plane only	Fusion
Roll test (125s)	test01	0.0121	0.0182	0.0344	**0.0090**
	test02	0.0147	0.0236	0.0457	**0.0126**
	test03	0.0162	0.0325	0.0593	**0.0152**
Pitch test (127s)	test01	0.0147	0.0196	0.0325	**0.0118**
	test02	0.0174	0.0214	0.0291	**0.0139**
	test03	0.0208	0.0241	—	**0.0165**
Mixed test (960s)	test01	0.386	0.0713	0.0674	**0.0451**
	test02	0.415	0.0651	—	**0.0584**
	test03	0.491	0.0742	—	**0.0617**

TABLE I: Average RMSE of roll and pitch angles (in radians). The frame-wise error bigger than radians 0.3 occurring over the half sequence length is considered as failure.

We can conjecture from Table I that our fusion approach consistently outperforms the other baseline approaches without fusion by a considerable margin on all sequences. The sequences cover three movement patterns: sequences with pure roll, pure pitch movement, and random rotation on both axes, and each case is implemented at different angular speeds, ordered from three levels, 3, 9, and 15 degrees per second. The good performance with the lowest RMSE in most tests can be attributed to our good assumptions of the environment, skyline, and ground plane in the wild. The IMU results in the table are from the 6-DoF Madgwick quaternion filter [10], without the use of a magnetometer. This is because in our case, the mountain region with active volcanoes is affected by the disturbances from the earth's magnetism field change. The filtered IMU results are prone to drift, as presented in the mixed test of 16 minutes, and the errors are nearly one order of magnitude bigger than roll and pitch test sequences. Either the use of skyline or the ground plane as a tracking cue can guarantee the error remaining at a lower level compared to Madgwick filter over IMU measurements, but in some fast rotation cases, the ground planes are partially or not present in the image, which may result in the failure. All of the results in the table justify the merits of using an adaptive particle filter over the manifold, improving the robustness, sensor redundancy, and accuracy.

Figure 9 further validates the consistency of our method. The test is repeated 10 times per sequence, and then an average error of all trials is taken over the mean error of the whole sequence. The slowest angular speed of the sequence (test01) for pure roll or pitch in Table I is employed. The variances of our fusion results are always the smallest compared to the results of the single sensor modality.

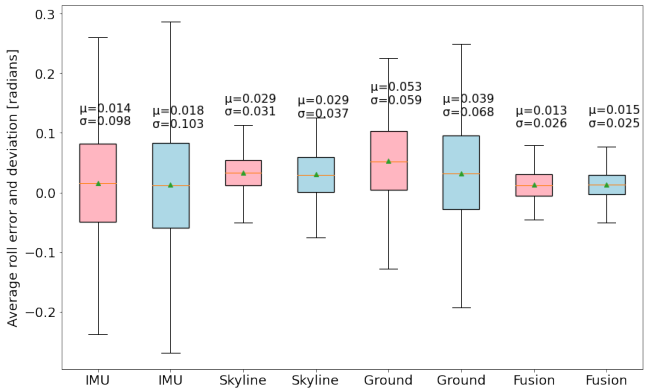


Fig. 9: Green arrow is mean error, the orange line is median error, and the box bounds represent the min/max errors. Roll and pitch results are in pink and blue respectively.

VII. Conclusion & Outlook

A new stand-alone gimbal system is proposed in this paper, based on tracking of natural geometry primitives, skyline, and ground plane approximations. The current frame rotation with respect to a reference frame can be derived by using two lines and normal vectors. Next, a specific particle filter with adaptive resolution-based sampling can fuse orientation from both CV and IMU pipelines, according to the various lifetime phases of one particle. The final experimental results are implemented on a 3D-printed gimbal platform. All simulation tests in the landscape are performed in real-time on a Jetson Nano platform. Four types of popular SOTA visual-inertial-based SLAM methods are chosen as references, revealing the best accuracy and robustness to drift and disturbances of our approach amongst all comparisons.

Our approach depends on a simple geometric primitive assumption. In challenging weather conditions, abrupt illumination changes, and cloud occlusions pose great challenges to feature tracking-based methods. Our gimbal system cannot work in a different scenario where the skyline is invisible, so for increased robustness, a hybrid system could be applied to general scenes by fusing feature points and the skyline. The estimated skyline is now simplified into a straight line for easy matching and real-time performance, on an affordable edge device. In reality, the curved mountains in the view challenge this assumption. The traditional iterative closest point (ICP) can be applied to the image level for matching. In addition, a Fisheye camera with a full view or multiple cameras at different views should be beneficial to the robustness of the gimbal platform.

VIII. Appendix

Dataset link: https://peridot-sailor-9cd.notion.site/

Acknowledgment

We thank our colleague Maarten Vandersteegen for the setup advice. We are grateful to the CSC scholarship to fund Xueyang for his research, the EPN international mobility scholarship to fund Ariel for his internship in Belgium, and research Project PIM21-01 as funding for the flight tests.

REFERENCES

[1] R. P. Workman Mihail, S. Bessinger, Z. & Jacobs, N. (2016, March). Sky segmentation in the wild: An empirical study. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1-6). IEEE.

[2] Adrian Carrio, Hriday Bavle, and Pascual Campoy. "Attitude estimation using horizon detection in thermal images." International Journal of Micro Air Vehicles 10.4 (2018): 352-361.

[3] Chung-Cheng Chiu, et al. "Vision-Based automatic flight control for small UAVs." parameters 1.1 (2011): 1.

[4] La Place, Cecilia, Aisha Urooj, and Ali Borji. "Segmenting sky pixels in images: Analysis and comparison." 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). IEEE, 2019.

[5] Abd EI Rahman Shabayek,, et al. "Vision based uav attitude estimation: Progress and insights." Journal of Intelligent & Robotic Systems 65.1 (2012): 295-308.

[6] Damien Dusha, Wageeh Boles, and Rodney Walker. "Fixed-wing attitude estimation using computer vision based horizon detection." Proceedings of AIAC12: 2nd Australasian Unmanned Air Vehicles Conference. Waldron Smith Management, 2007.

[7] Tony Lindeberg, et al. "Scale invariant feature transform." (2012): 10491.

[8] John Canny, et al. "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence 6 (1986): 679-698.

[9] David Nistér. "An efficient solution to the five-point relative pose problem." IEEE transactions on pattern analysis and machine intelligence 26.6 (2004): 756-770.

[10] Madgwick, Sebastian. "An efficient orientation filter for inertial and inertial/magnetic sensor arrays." Report x-io and University of Bristol (UK) 25 (2010): 113-118.

[11] Carlos Campos, et al. "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam." IEEE Transactions on Robotics 37.6 (2021): 1874-1890.

[12] Tong Qin, Peiliang Li, and Shaojie Shen. "Vins-mono: A robust and versatile monocular visual-inertial state estimator." IEEE Transactions on Robotics 34.4 (2018): 1004-1020.

[13] Zheng Huai, and Guoquan Huang. "Robocentric visual-inertial odometry." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

[14] Lukas von Stumberg, and Daniel Cremers. "DM-VIO: Delayed Marginalization Visual-Inertial Odometry." IEEE Robotics and Automation Letters 7.2 (2022): 1408-1415.

[15] Joern Rehder, Janosch Nikolic, Thomas Schneider, Timo Hinzmann, Roland Siegwart (2016). Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 4304-4311, Stockholm, Sweden.

[16] Paul Furgale, Joern Rehder, Roland Siegwart (2013). "Unified Temporal and Spatial Calibration for Multi-Sensor Systems." In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan.

[17] Zhengyou Zhang, et al. "A flexible new technique for camera calibration." IEEE Transactions on pattern analysis and machine intelligence 22.11 (2000): 1330-1334.

[18] Aytaç Altan, and Rıfat Hacıoğlu. "Model predictive control of three-axis gimbal system mounted on UAV for real-time target tracking under external disturbances." Mechanical Systems and Signal Processing 138 (2020): 106548.

[19] Saining Xie, et al. "Aggregated residual transformations for deep neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[20] Fredrik Gustafsson, et al. "Particle filter theory and practice with positioning applications." IEEE Aerospace and Electronic Systems Magazine 25.7 (2010): 53-82.

[21] Thomas Albrecht, et al. "Omnidirectional video stabilisation on a virtual camera using sensor fusion." 2010 11th International Conference on Control Automation Robotics & Vision. IEEE, 2010.

[22] Shamsundar Kulkarni, M. D. S. Bormane, and S. L. Nalbalwar. "RANSAC algorithm for matching inlier correspondences in video stabilisation." International Journal of Signal and Imaging Systems Engineering 10.4 (2017): 178-184.

[23] Sebastiano Battiato, et al. "SIFT features tracking for video stabilization." 14th international conference on image analysis and processing (ICIAP 2007). IEEE, 2007.

[24] Yasuyuki Matsushita, et al. "Full-frame video stabilization." 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). Vol. 1. IEEE, 2005.

[25] Rong Hu, et al. "Video stabilization using scale-invariant features." 2007 11th International Conference Information Visualization (IV'07). IEEE, 2007.

[26] Binoy Pinto, and P. R. Anurenjan. "Video stabilization using speeded up robust features." 2011 International Conference on Communications and Signal Processing. IEEE, 2011.

[27] Wilbert G. Aguilar, and Cecilio Angulo. "Real-time model-based video stabilization for microaerial vehicles." Neural processing letters 43.2 (2016): 459-477.

[28] Junlan Yang, Dan Schonfeld, and Magdi Mohamed. "Robust video stabilization based on particle filter tracking of projected camera motion." IEEE Transactions on Circuits and Systems for Video Technology 19.7 (2009): 945-954.

[29] Hany Farid, and Jeffrey B. Woodward. "Video stabilization and enhancement." (2007).

[30] Paresh Rawat, and Jyoti Singhai. "Review of motion estimation and video stabilization techniques for hand held mobile video." Signal & Image Processing: An International Journal (SIPIJ) Vol 2 (2011).

[31] Jiyang Yu, and Ravi Ramamoorthi. "Learning video stabilization using optical flow." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[32] Jiyang Yu, et al. "Real-Time Selfie Video Stabilization." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[33] Chen Li, et al. "Deep Online Video Stabilization using IMU Sensors." IEEE Transactions on Multimedia (2022).

[34] Oswualdo Alquisiris-Quecha, and Jose Martinez-Carranza. "Video Stabilization of the NAO Robot Using IMU Data." In Robot Operating System (ROS), pp. 147-162. Springer, Cham, 2020.

[35] Jutamanee Auysakul, He Xu, and Vishwanath Pooneeth. "A hybrid motion estimation for video stabilization based on an IMU sensor." Sensors 18.8 (2018): 2708.

[36] Yao-Chih Lee, et al. "3D Video Stabilization with Depth Estimation by CNN-based Optimization." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[37] J. Choi, J. Park, I. S & Kweon. (2021). Self-Supervised Real-time Video Stabilization. arXiv preprint arXiv:2111.05980.

[38] Yu-Lun Liu, et al. "Hybrid neural fusion for full-frame video stabilization." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021.

[39] Ahlem Walha, Ali Wali, and Adel M. Alimi. "Video stabilization for aerial video surveillance." Aasri Procedia 4 (2013): 72-77.

[40] Ahlem Walha, Ali Wali, and Adel M. Alimi. "Video stabilization with moving object detecting and tracking for aerial video surveillance." Multimedia Tools and Applications 74.17 (2015): 6745-6767.