JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 5, August 2008

Stabilization of Video from Miniature Air Vehicles for Target Localization

David L. Johansen[*] , James K. Hall[†] , Randal W. Beard[‡] , and Clark N. Taylor[§]
Brigham Young University, Provo, UT, 84602, USA

DOI: 10.2514/1.34228

Miniature air vehicles equipped with lightweight, inexpensive cameras have recently been employed in a greater variety of missions owing to their increased capability and relatively low cost. Unfortunately, miniature air vehicle video is often difficult to watch owing to high-frequency jitter caused by environmental disturbances such as wind gusts. This paper describes a methodology for producing real-time stabilized video by eliminating jitter. Video stabilization is enabled by estimating the intended video motion through the methodical selection and tracking of image features from one frame to the next. The effectiveness of the video stabilization routine is demonstrated in a real-world surveillance task of localizing stationary and moving targets on a mobile ground station. The paper also describes the unique mobile platform that is used to create and refine these miniature-air-vehicle-specific computer vision technologies.

Nomenclature

C	camera calibration matrix
$\mathbf{M}$	*a* 3 × 3 matrix representing motion of the frame
$\mathbf{M}_i^j$	matrix representing motion from frame i to frame j
$\mathbf{M}_s^k$	motion applied to the kth image to stabilize the frame
$\mathbf{M}_m$	matrix representing all motion in the video up to frame m
$\bar{m}$	average movement of object pixels in the frame
o_i^k	virtual object i at frame k
$R_{\theta_{\text{gim}}}$, $R_{\psi_{\text{gim}}}$	rotation matrices about the gimbal pitch and yaw axes
R_ϕ, R_θ, R_ψ	rotation matrices about the roll, pitch, and yaw axes
s	uniform image scaling factor
s_I	scale factor between the rendering context and the rendered image size
T_i^j	transformation from coordinate frame i to coordinate frame j
T_x, T_y	image translations
t_x, t_y	position of the image in the rendering context
u_x, u_y	image coordinates of the operator's input
V_{ray}	direction of ray to the target coordinates

Received 28 August 2007; accepted for publication 3 May 2008. Copyright © 2008 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/08 $10.00 in correspondence with the CCC.

[*] Engineer, Raytheon Missile Systems, David_Johansen@raytheon.com

[†] Graduate Research Assistant, Mechanical Engineering, hallatjk@gmail.com

[‡] Professor, Electrical and Computer Engineering, beard@byu.edu

[§] Assistant Professor, Electrical and Computer Engineering, clark.taylor@byu.edu

x_i, y_i	location of the ith feature in the previous image frame
$x_i^{'}$, $y_i^{'}$	location of the ith feature in the current image frame
θ_{gim}, ψ_{gim}	pitch and yaw angles of the gimbal relative to the body frame
ϕ, θ, ψ	roll, pitch, and yaw angles

I. Introduction

SMALL, inexpensive, onboard cameras have greatly contributed to the increased optical surveillance capabilities of miniature air vehicles (MAVs). These capabilities enable the MAV to be used in a wide variety of missions such as area surveillance by search-and-rescue teams, tactical reconnaissance by military squads, and forest fire perimeter tracking by firefighters. The list of potential applications will continue to grow as MAVs become more robust and effective at gathering and disseminating information. However, MAVs have a significant deficiency in their role as an optical surveillance platform: MAV-mounted cameras often experience abrupt lateral and longitudinal accelerations owing to atmospheric turbulence. As a result, MAV video streams are frequently characterized by high-frequency jitter, making it difficult for a human operator to obtain useful information from the video. Ratches et al. [1] observed that a robust target identification tool exists: the human visual system. The operator's ability to identify objects in the MAV video stream is greatly enhanced by removing video stream jitter through the stabilization methodology presented in this paper.

While several solutions exist for stabilizing video, they each have shortcomings that make them impractical for use with MAVs. One approach for creating stabilized video uses specialized hardware such as those presented in references [2–4]. In these cases, the hardware must be installed in a peripheral component interconnect (PCI) slot of a desktop computer. For many MAV applications, the objective is to minimize the amount of equipment carried by the operator, which includes a laptop computer and antennae for communication. Additional equipment dedicated solely to video stabilization would be unwieldy. Therefore, operational requirements for real-time MAV video stabilization dictate a software solution.

Various software video stabilization innovations have been developed in recent years, including optical flow feature tracking with iterative least squares [5], affine models and pyramidal techniques [4], profile matching and sub-sampling [6], and estimated camera motion with image-based rendering techniques [7]. While most video stabilization techniques build on a foundation of frame-to-frame feature tracking, MAV video often contains large amounts of noise owing to analog transmission of the video to the ground station, causing false features to be identified. Additionally, the motions considered in previous publications were typically produced by a person holding a hand-held camera [7, 8], whereas the tools developed in this paper address the relatively large velocities typical of MAV flight.

Stabilized MAV video enables other surveillance and reconnaissance tasks such as target identification and target localization. Once a target has been identified as part of a surveillance mission, operators often need to know its latitude, longitude, and elevation. This process, called target localization, uses the position and attitude of the MAV (determined by on-board sensors) in conjunction with the video stream to estimate the world coordinates of the target. The accuracy of target localization is sensitive to synchronization errors between the sensor data and the video stream. In our application, sensor data and video data are transmitted from the MAV at different rates and via different communication channels: one analog and the other digital. Owing to the two disparate rates and types of transmission, synchronizing the sensor data and the video stream is a challenging problem.

Synchronizing video and sensory data is currently an active research area with two main thrusts. The first, off-line synchronization, matches video with data after capturing the entire data stream. Research involving offline synchronization offers important insights into various methods for time stamping data from disparate sensors (accelerometers, gyroscopes, cameras, and so on) before synchronization [9–11]. The techniques presented in this paper use elements of offline synchronization in the sense that the data from several aircraft sensors are gathered into a state vector and transmitted independent of the time stamped video. The second research thrust in video-data synchronization is online synchronization, where data from the aircraft sensors and video stream are aligned during the collection process. Online synchronization is demonstrated by Rieger [12] and Zhang [13], but both methods require special hardware whose weight and power requirements make them unsuitable for MAV applications. Reference [14] demonstrates data synchronization following transmission, provided the video and data are time-stamped. This paper extends those results to MAVs controlled from mobile ground stations, achieving online synchronization of video and aircraft data for immediate display to the operator during time-critical surveillance missions.

Creating a video stream that is synchronized with aircraft position and attitude data is an essential intermediate step for obtaining a precise geo-location for a specified target. References [15] and [16] have both demonstrated initial results for this critical MAV surveillance capability. In the current paper, we extend their results by demonstrating localization of both fixed targets and moving targets from the stabilized video stream. The ability continuously to track moving targets is based on the same feature tracking methodologies developed for video stabilization.

A description of the development platform used for creating and refining new MAV computer vision technologies is found in Sec. II. The development platform is used to refine three computer vision technologies for MAVs: video and data synchronization, video stabilization, and target localization. Section III discusses feature rating and selection techniques that are key to video stabilization. Section III also discusses feature tracking methods, frame motion estimation schemes, and noise rejection methods. Section IV presents an operational application of the MAV video stabilization technology: target localization. Key results that demonstrate the video stabilization tools in localizing stationary and moving targets are presented in Sec. V. General observations and recommendations are presented in Sec. VI.

II. Computer Vision Development Platform

The first contribution of this paper is a set of MAV ground station software tools that together constitute a mobile computer vision development platform to act as the central hub for interaction between the MAV, the video processing software (FrameProcessor), and the operator. The MAV asynchronously transmits video and telemetry data to the ground station, which synchronizes and sends the two signals to the FrameProcessor. The FrameProcessor returns commands to the ground station and displays the processed video to the operator. Finally, the ground station transmits operator and FrameProcessor commands to the MAV.

A. MAV System Development Software

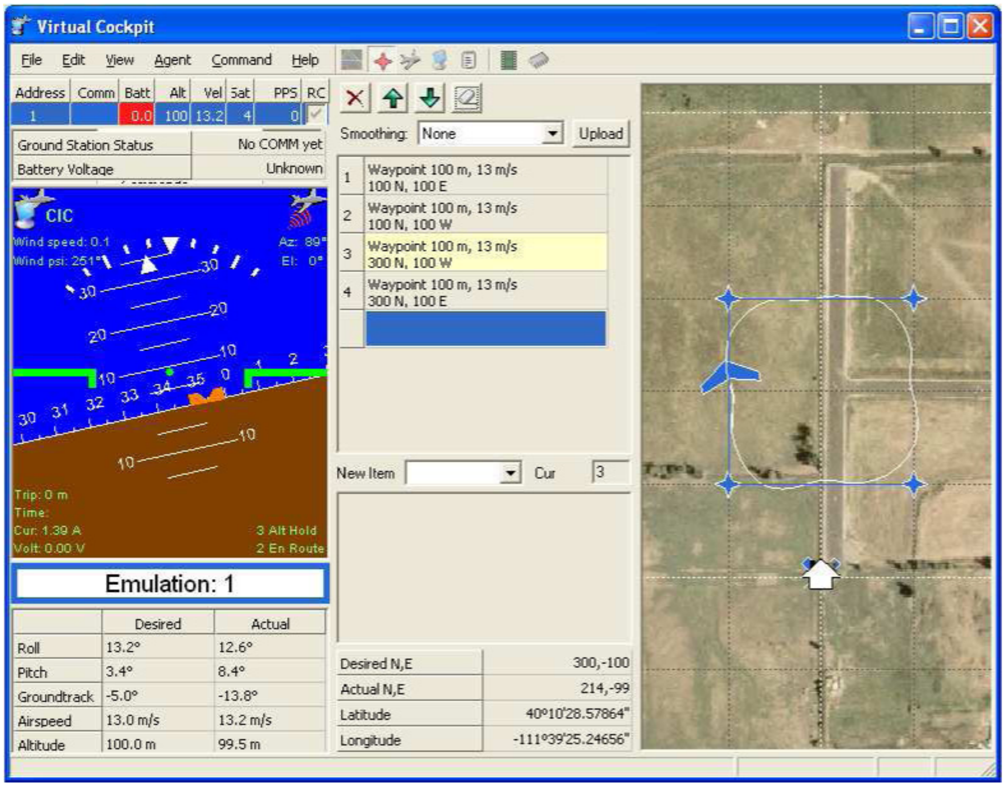
The software for developing MAV technologies created at Brigham Young University (BYU) is built on several key pieces of software and hardware including the Virtual Cockpit, the flight simulator software (Aviones), the Kestrel$^{\text{TM}}$ Autopilot, and the vehicle platforms. The Virtual Cockpit is a software application that was developed to allow operator interface with MAVs. It has the ability to communicate with a flying MAV through a 900 MHz radio modem or with an emulated MAV in the flight simulator software. The ability to interface with both a hardware platform and a software simulator allows for quick development and testing of control algorithms.

The Virtual Cockpit presents all the vital MAV telemetry data on a heads-up display (HUD). It also displays a geo-referenced terrain map with the ability to define a flight plan using point-and-click mouse commands. The current position of the MAV is continuously displayed on the map allowing the operator to monitor the MAV's progress along the path. The Virtual Cockpit also contains a framework for accessing other autopilot variables, allowing for in-flight gain tuning and parameter verification. Additional features include the ability to stream video and record important telemetry data. A screenshot of the Virtual Cockpit HUD and map display is shown in Fig. 1.

The flight simulation environment used in conjunction with the Virtual Cockpit is called Aviones. It was developed at BYU and the source code is available at SourceForge[¶]. Aviones supports the simulation of multiple MAVs, as well as different types of aircraft with the associated physics and sensor models. Aviones is capable of rendering MAVs flying above three-dimensional (3-D) geo-referenced terrain maps and virtual urban environments. An Aviones screenshot of a MAV flying through a virtual city is shown in Fig. 2. Aviones allows specification of environmental factors such as local magnetic field and wind velocity. The essential feature of Aviones is the autopilot emulation framework that allows code written for the aircraft autopilot to be compiled as a dynamically linked library (DLL). Thus, the code that will actually be flown on the MAV can be loaded into the simulation and tested. As a result, the control algorithms can be validated in the software environment before flight tests on the actual autopilot.

Aviones communicates with the Virtual Cockpit over a TCP/IP socket as though the simulated aircraft were a physical aircraft communicating via modem. The physics engine of Aviones provides state information to the

[¶] http://sourceforge.net/projects/aviones



Address	Comm	Batt	Alt	Vel	Sat	PPS	RC
1		0.0	100	13.2	4	0	✓

Ground Station Status	No COMM yet
Battery Voltage	Unknown

Wind speed: 0.1
Wind psi: 251°
Trip: 0 m
Time:
Cur: 1.39 A
Volt: 0.00 V

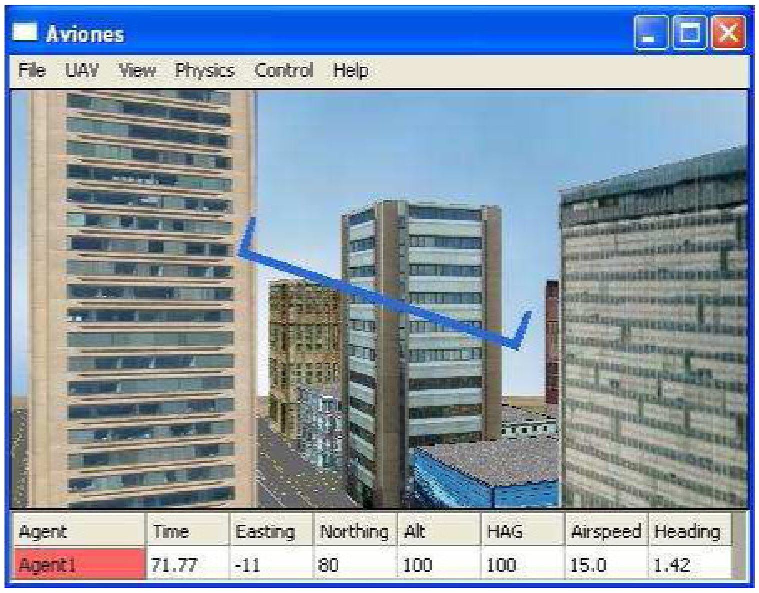
Emulation: 1

	Desired	Actual
Roll	13.2°	12.6°
Pitch	3.4°	8.4°
Groundtrack	-5.0°	-13.8°
Airspeed	13.0 m/s	13.2 m/s
Altitude	100.0 m	99.5 m

1	Waypoint 100 m, 13 m/s 100 N, 100 E
2	Waypoint 100 m, 13 m/s 100 N, 100 W
3	Waypoint 100 m, 13 m/s 300 N, 100 W
4	Waypoint 100 m, 13 m/s 300 N, 100 E

Desired N,E	300,-100
Actual N,E	214,-99
Latitude	40°10'28.57864"
Longitude	-111°39'25.24656"

Fig. 1 A screenshot of the Virtual Cockpit map page, which shows the simulated heads-up display, the waypoint list, and a bird's eye view of the aircraft and waypoint path with respect to home station. Other aircraft functions are controlled using the other menu pages.



Agent	Time	Easting	Northing	Alt	HAG	Airspeed	Heading
Agent1	71.77	-11	80	100	100	15.0	1.42

Fig. 2 The Aviones flight simulation software is used to develop and test new MAV guidance and control algorithms. The software simulates realistic environments for both urban and rural terrain.

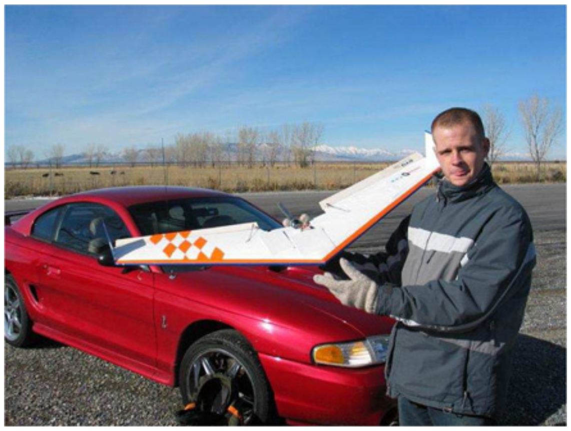


Fig. 3 A typical MAV is seen here being held by BYU student Ryan Holt.

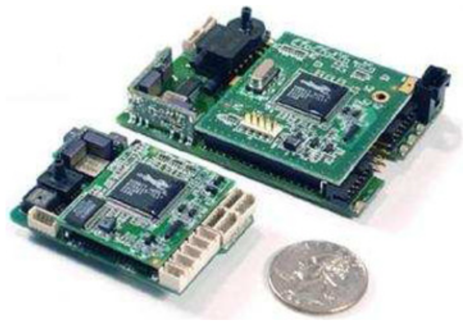


Fig. 4 The Kestrel^{TM} autopilot, versions 1.45 (top) and 2.0, can be reprogrammed to flight test new guidance and control techniques.

autopilot DLL and replaces the sensors of the actual autopilot with models. The Virtual Cockpit and Aviones are tools that enable quick transition from development to flight testing.

B. MAV System Development Hardware

The flight test hardware consists of an autopilot-equipped air platform which communicates via radio modem to the ground station: a laptop computer running the Virtual Cockpit software. The airframe used in these experiments is a commercial off-the-shelf flying-wing design of expanded polypropylene (EPP) construction, with a 48-in. wingspan (shown in Fig. 3). The EPP construction is extremely durable and allows additional components to be easily added. Propulsion is from an electric motor, giving a fully loaded aircraft weight of under two pounds.

The autopilot used in these experiments is the Kestrel^{TM} Autopilot available from Procerus Technologies (see Fig. 4). The autopilot is equipped with a Rabbit microprocessor operating in conjunction with solid-state rate gyros, accelerometers, and pressure sensors. The autopilot is also equipped with a serial peripheral interface (SPI) and general-purpose digital I/O ports. The MAV is also equipped with a Furuno GH-80 global positioning system (GPS) receiver.

Given the information from the sensors, the autopilot software computes vehicle state information such as altitude, airspeed, and GPS position. The autopilot allows the MAV to execute auto-takeoff climb-to-altitude, maintain constant altitude, follow geo-referenced waypoints, and land autonomously. Two key enabling capabilities for video stabilization are 1) continuous communication of aircraft telemetry to the ground station and 2) recording specified variables for subsequent download to the Virtual Cockpit for analysis.

C. Time Stamps

Time stamps used for synchronizing the video and telemetry data are dependent on the availability of Coordinated Universal Time (UTC) from the GPS unit. The ground station must synchronize the video and telemetry data to create accurate information as the video is received at a constant rate (30 Hz with a constant transmission delay) and telemetry data are transmitted at an irregular rate (between 3–5 Hz with a variable transmission delay). Each video frame is tagged with the current UTC time minus the constant value of transmission delay to reflect the approximate time when the video frame was originally captured by the camera. The telemetry data are time stamped on the MAV using the current UTC time. Matching the two time stamps allows the ground station to synchronize the telemetry data with the buffered video.

The constant video transmission delay is the time it takes for the video to pass through the camera, transmitter, receiver, and framegrabber before arriving at the ground station. The video transmission delay was estimated by oscillating the MAV about its roll axis with a camera pointed out the wing at a fixed object. The autopilot recorded and transmitted the roll angle along with the UTC time stamp to the ground station. The FrameProcessor located the object in the video and both the y-pixel coordinate of the object center and the UTC time stamp on the ground station were recorded. The peaks of the two data sets were found and the mean phase shift was calculated, giving a reasonably accurate estimate of video transmission delay. Figure 5 shows the MAV roll and the y-pixel coordinate of the center of mass of the object after synchronization. The final estimate of the video transmission delay was found to be 95 milliseconds.

D. Video Handling Pipeline

The FrameProcessor video handling system is designed using a software pipelining architecture that exploits modern multiprocessor and multithreading systems. To enable software pipelining, the video handling and synchronization processes are scheduled according to the block diagram shown in Fig. 6. Each task is performed in its own thread and outputs the data to the next task through a queue. This isolation of the Frame Processor adds stability to the ground station software because it cannot be frozen by the FrameProcessor.

The video is initially captured and frames are added to the ToConvert Queue. The frames are subsequently processed by the Conversion Thread and converted from the YUV to RGB color-space. The FrameProcessor can then

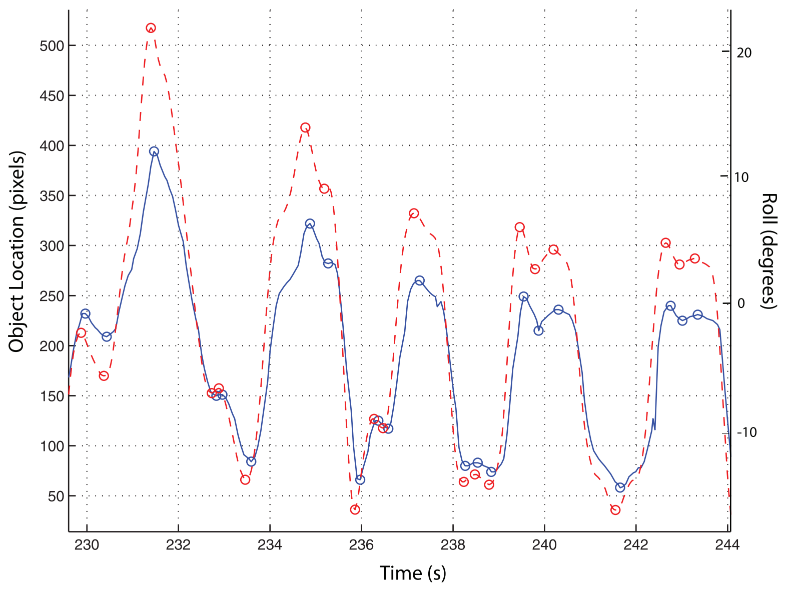


Fig. 5 The solid blue line is the y-pixel coordinates and the dotted red line shows the MAV roll angle. The circles represent the matched peaks used to estimate the mean phase shift between the two data sets to determine the typical transmission delay.

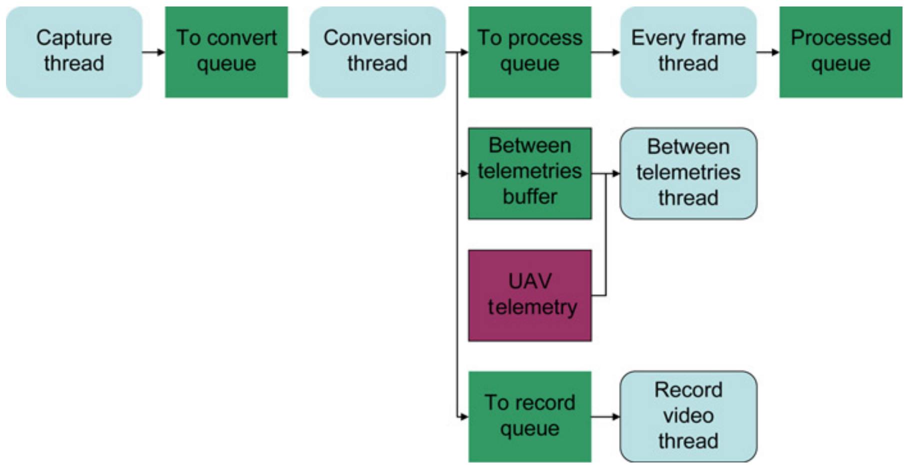


Fig. 6 This figure shows the video handling pipeline. The rounded boxes represent the tasks in the pipeline that are each performed in their own thread. The square boxes represent data that flow through the pipeline.

convert the images to any image format required for later processing. The Conversion Thread inserts the frame into the ToProcess Queue, Between Telemetries Buffer, and the ToRecord Queue to be handled by their corresponding threads. The Every Frame Thread takes the frame from the ToProcess Queue and performs the operations required by the FrameProcessor. This thread will be discussed in more detail in Sec. III. Upon completion of the Every Frame Thread, the FrameProcessor can request that the frame be stored in the Processed Queue for later use. If requested by the operator, the Record Video Thread records the frame to a file along with the associated telemetry. The Between Telemetries Buffer is a running buffer that stores a fixed number of frames, and as telemetry data are received from the MAV, the Between Telemetries Thread extracts the frames that lie between the time stamp of the previous and current telemetry data from the Between Telemetries Buffer. The FrameProcessor then performs the processing of synchronized video on the extracted frames. This system allows for handling and displaying video streams either as frames are received or when telemetry is received. The FrameProcessor performs all vision processing in the ground station software, creating custom conversions of video frames, processing frames of the video before and after synchronization, and displaying both of these video streams to the operator simultaneously. A mechanism for capturing and displaying a paused frame of video for more detailed processing is also available.

III. Video Stabilization

Figure 7 shows a graphical representation of the software pipeline that performs real-time MAV video stabilization to enable target localization in world coordinates. Each step in the MAV video stabilization process relies only on image processing techniques and is performed in the Every Frame Thread, whereas the target localization process is accomplished in the Between Telemetries Thread of the video handling pipeline, as shown in Fig. 6. Stabilized video requires that the following five steps are performed on every frame received from the MAV: feature selection, feature tracking, frame motion estimation, estimation of intended video motion, and video display. Each of these steps will be discussed in the sections that follow.

A. Feature Selection

Even though the foundation of video stabilization is feature tracking, and feature tracking depends entirely on selecting features that can be accurately tracked, very little has been discussed in the literature about feature selection during the stabilization process. It is therefore vital to understand the characteristics of a good feature and to specify feature ratings to quantify these attributes.

1. Feature Attributes

The ideal feature is identifiable, unique, provides new information, and persists from frame-to-frame. An identifiable feature is defined as an area within an image having high gradient values. Closely related to identifiability is

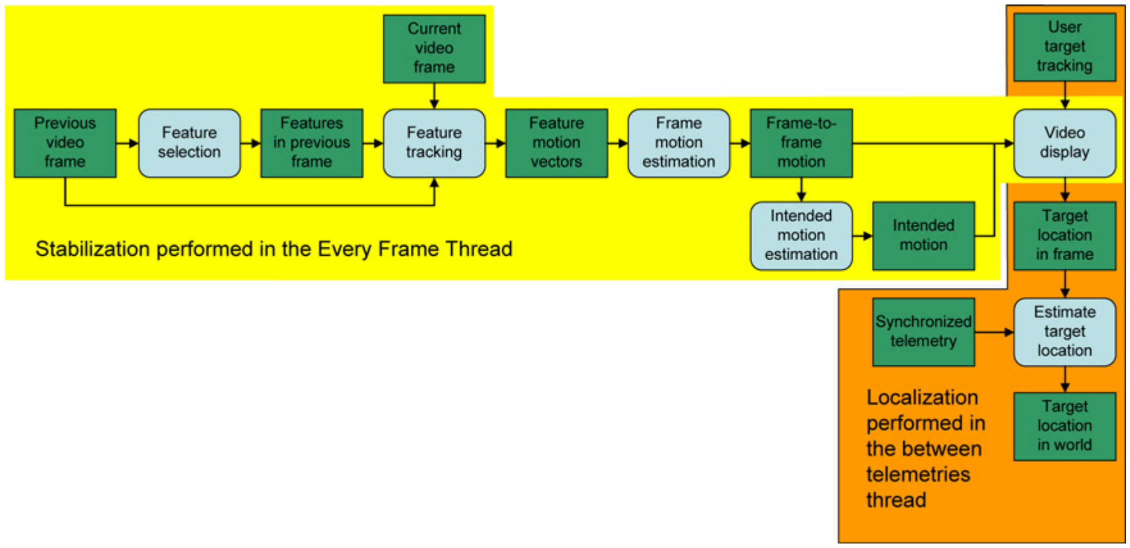


Fig. 7 The rounded boxes represent tasks in the stabilization and localization pipeline and the square boxes represent data that flow through the pipeline.

the quality of uniqueness. Lines are an example of image features that have high gradient values but are not unique because points on the line cannot be distinguished from other points along the line using image processing techniques.

Each feature also needs to provide new information to the stabilization process. Tracking a feature directly adjacent to a previously tracked feature does not provide new information relative to the original feature. This is avoided by ensuring the features are broadly distributed in the image.

Frame-to-frame persistence is not a characteristic that can be enforced during the feature selection process. However, understanding the causes of feature disappearance between frames facilitates the detection of invalid features. The primary reasons for feature disappearance are noise, occlusion, and translation: each presenting a unique challenge.

The transmission noise in the MAV video stream is the most challenging problem to overcome when selecting features for stabilization. The significance of video noise can be seen in Fig. 8, where sequential frames from a video stream show substantial image degradation caused by noise. Unfortunately, noise tends to have very high gradient values, producing high feature ratings in most feature selection techniques. Although existing techniques cannot satisfactorily remove noise during feature selection, noise that was improperly selected as a trackable feature can usually be detected during later steps of the stabilization process. Properly handling signal noise is an essential capability for MAV video stabilization.

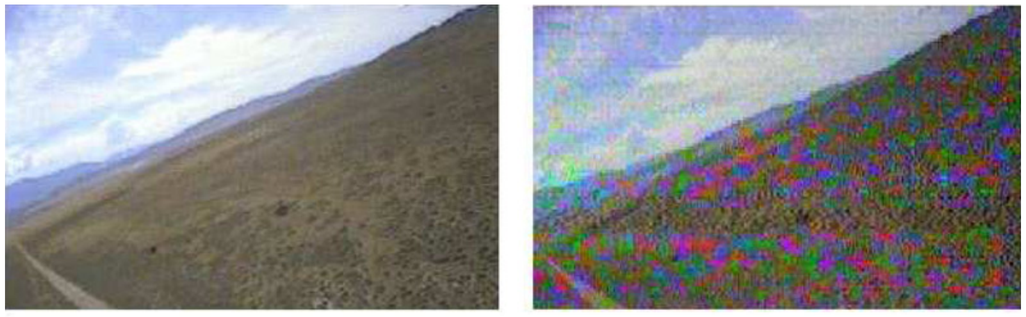


Fig. 8 The two images are subsequent frames in a video stream, with (b) showing the effects of transmission noise.

Occlusion can only be detected during feature selection with computationally intensive image analysis techniques, which are not feasible to accomplish in real-time on our mobile ground station. However, the same techniques used for detecting noise features can also detect feature occlusion.

As continuous translational and rotational motions are expected in MAV video, the problem of features translating out of the video images from one frame to the next must be addressed. The most direct solution to this problem, and the approach used in our work, is to select new features in each video frame.

2. *Feature Rating*

To quantify the feature characteristics, a rating is assigned to every pixel in the image. This quantification is used to determine the image regions that can be most accurately tracked between frames. Six different feature rating methods were tested: gradient using Sobel kernel [17], Canny edge detector [18], Forstner interest operator [19], Harris corner detector [20], binary corner detector [21], and the SIFT feature selection method [22, 23]. For MAV video, the performance of gradient and Canny edge detector methods was highly degraded owing to video noise. The Forstner interest operator was susceptible to assigning an incorrectly high rating to nonunique features such as lines. Although the binary corner detector had better noise rejection properties than the Harris corner detector, it was unable to provide a rating of feature strength. As the rating of feature strength is essential for ensuring the best features are used in the estimated video motion calculations, the Harris corner detector was determined to be the most suitable for use in the real-time MAV video stabilization routine.

After rating the features, the next step is to ensure each provides new information. In this paper, two methods are discussed, namely region-based and minimum separation. The region-based feature selection method divides the image into rectangular boxes and selects the best feature from each. This method guarantees that all of the features will be distributed throughout the image. However, it may select several features in very close proximity when an area with high texture value lies on the intersection of several regions. The minimum separation method chooses the features with the highest rating from the image but guarantees that selected features are separated by a specified minimum distance. This guarantees that the best possible features are selected and that grouping does not occur.

For example, Fig. 9 shows the distribution of features (found using Harris corner detection) when the minimum separation and region-based feature selection methods are used. It can be seen in Fig. 9(a) that the minimum separation method ensures that features are not tightly grouped while selecting the best features available in the image. Figure 9(b) shows the improper grouping that can occur when region boundaries fall on an area with high feature rating. The region-based method also requires that exactly one feature be selected in each region. Therefore, regions with poor overall feature rating result in the selection of a suboptimal feature and regions with high overall feature rating

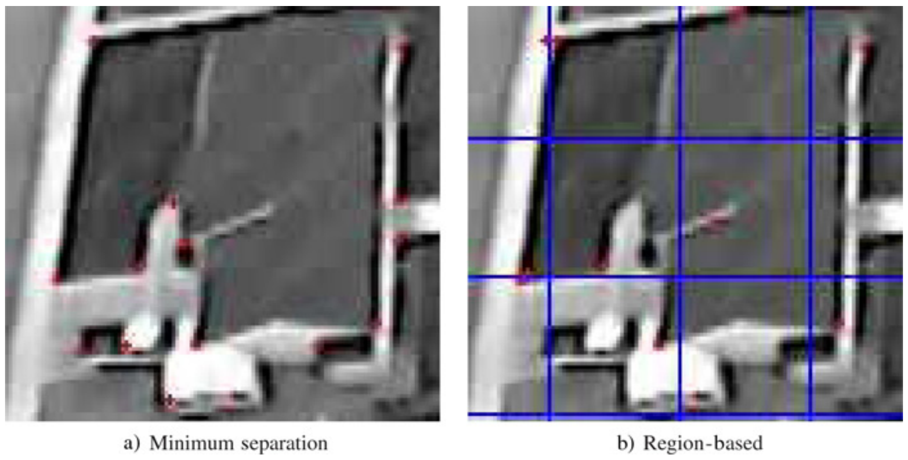


Fig. 9 The red crosses in both figures show the selected features for the minimum distance and region-based methods. There are two instances in (b) where a region boundary split two features causing both to be selected even though they were very near to each other.

can reject several high-quality features. Given these factors, minimum separation is the preferred feature selection technique.

B. Feature Tracking

Feature tracking uses the list of selected features from the previous frame and estimates their position in the current frame, creating a set of feature motion vectors for use in calculating the estimated frame-to-frame motion. Three feature tracking techniques—template matching, profile matching, and optical flow—and their effectiveness with MAV video were evaluated. The optical flow algorithm, which was implemented pyramidally, had difficulty tracking features in high noise images and was more computationally expensive than template matching with pyramidal techniques. Profile matching [6], a reduced computational approach to template matching, was found to be too inaccurate and sensitive to noise for MAV video. Therefore, template matching (with pyramidal techniques) was determined to be the preferred option.

Template matching is accomplished by using a window centered on the feature in the previous frame compared with a search window in the current frame using sum of absolute differences. This method assumes that significant rotation does not occur at the feature level and that significant lighting changes do not occur between frames. These assumptions are valid for video captured by MAVs flying typical surveillance missions.

Feature selection problems such as repeating patterns and noise can be detected by adding constraints to the template matching criteria. The first constraint identifies features caused by noise. These features will not be consistent from frame-to-frame, implying a close match will not exist in the next frame. Therefore, by mandating a maximum error threshold, these features can be rejected. This threshold was found by examining the typical error of correctly tracked features in a variety of MAV video footage. The maximum error threshold constraint rejected more than 90% of the invalid features while maintaining more than 95% of the valid features.

The second constraint handles features that occur in a problem area, such as features that are part of a repeating pattern. These features can be detected by requiring a unique match in the search window. These methods make use of the data available in the image without the complex image analysis techniques required to reject them during the feature selection process, thus accurately detecting invalid features without an excessive increase in computational costs.

The performance of template matching can be improved through pyramidal techniques, by using a small template window to track the feature at the top level of the pyramid and again on successive levels, each with progressively larger template windows until the original frame is reached. As the template matching on each of the pyramid levels is performed using the described template matching techniques, the same methods for detecting noise during the tracking process can be applied as the feature is tracked down the pyramid. The pyramidal template matching technique reduces the average computational cost from 0.75 milliseconds per feature to 0.05 milliseconds per feature on a 3.2 GHz Pentium IV processor. This reduction in computational cost allows for a dramatic increase in the number of real-time trackable features, but results in a 10% decrease in the number of correctly tracked features.

The most important aspect of feature tracking is accurately recording the position of real features while rejecting false features, so that frame-to-frame motion can be correctly calculated. Figure 10 shows the results of template matching on an area of the image with high noise levels. Figures 10(a) and 10(b) show an image before and after translation, and Fig. 10(c) shows the features that were tracked and those that were rejected.

C. Frame Motion Estimation

The set of feature motion vectors retrieved during feature tracking are used to estimate the motion needed to overlay the current frame on the previous frame to minimize the visible motion. The motion is estimated from the feature motion vectors based on a frame motion model used in conjunction with methods for detecting improperly tracked features for each of these models. Homography-based motion models are inapplicable to many MAV video streams owing to the horizon being visible. The translational motion model did not stabilize the video sufficiently due to its inability to remove rotational jitter. Therefore, we chose an affine motion model to represent the frame-to-frame motion of the video.

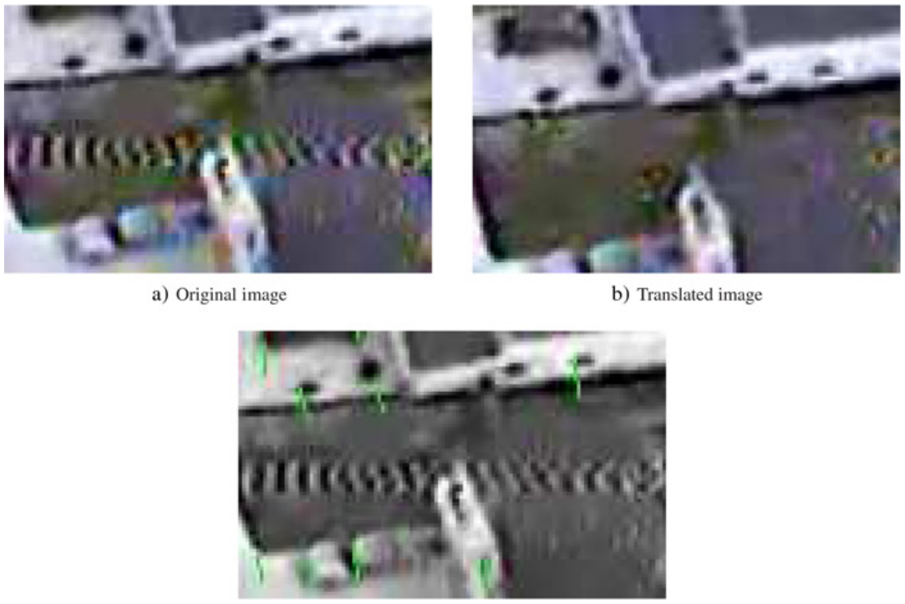


Fig. 10 The feature motion vectors are shown as green lines and the features that could not be correctly tracked are shown as red dots. The template matching method correctly rejects the features in the areas with high noise levels.

1. Affine Motion Model

The affine motion model is defined as follows: θ is a rotation angle, s is a uniform x and y scaling factor, and T_x and T_y are the respective translations. The equation representing the affine model is

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = s \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \end{bmatrix}. \tag{1}$$

The parameters of the affine model are estimated using the least squares method from the feature motion vectors as follows. Let (x_i, y_i) be the location of feature i in the previous frame and (x_i', y_i') be the location of the same feature in the current frame. Also, let $k_1 \triangleq s\cos\theta$ and $k_2 \triangleq s\sin\theta$ and let $x = \begin{bmatrix} k_1 & k_2 & T_x & T_y \end{bmatrix}^T$, then x can be estimated from the equation, $x = (A^T A)^{-1} A^T b$, where

$$A = \begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ y_2 & x_2 & 0 & 1 \\ & \vdots & & \\ x_n & -y_n & 1 & 0 \\ y_n & x_n & 0 & 1 \end{bmatrix} \tag{2}$$

and

$$b = \begin{bmatrix} x_1' & y_1' & x_2' & y_2' & \cdots & x_n' & y_n' \end{bmatrix}^T \tag{3}$$

After computing x, θ and s can be found from the relationships

$$\theta = \tan^{-1}\left(\frac{k_2}{k_1}\right) \tag{4}$$

and

$$s = \frac{k_1}{\cos\theta} = \sqrt{k_1^2 + k_2^2} \tag{5}$$

2. *Outlier Data Rejection*

Two methods exist for rejecting outlier data points in the affine motion model: iterative least squares [5, 24] and RANSAC [25]. Although, the iterative least squares algorithm can be extended to the problem of MAV video stabilization, we found that if the features are not evenly distributed, this method will exaggerate the biasing caused by features that are grouped together. Hence, we chose to use RANSAC, which can be used to detect features that should not be tracked.

RANSAC is performed by selecting two random feature motion vectors from the data set and estimating the affine motion for those vectors. All other vectors in the data set are compared to this model and vectors that lie within a specified threshold of the model are defined as inliers. This process is repeated several times and the affine motion with the largest set of inliers is used for determining the final estimate. The effectiveness of our frame motion estimation technique and the benefits of using RANSAC are shown in Sec. V.

D. Removal of Unwanted Video Motion

The frame-to-frame motion in MAV video consists of two components: 1) intended video motion owing to the MAV flight path and 2) unwanted jitter caused by atmospheric disturbances. In Fig. 11, we illustrate the difference between intended and unwanted video motion. The blue line in Fig. 11 denotes the rotational component of the frame-to-frame motion measured in video footage captured by an MAV orbiting around a fixed object on the ground. Owing to the orbit pattern of the MAV, there is a nonzero, constant portion of the rotational video motion (illustrated by the green line in Fig. 11) that represents the intended motion present in the video. The remainder of the motion is undesirable, causing extra motion of the objects in the video. The goal of video stabilization is to remove the unwanted motion while leaving the intended motion.

To remove unwanted jitter from the video stream, the motion equation from (1) is written in matrix form as $\mathbf{x} = \mathbf{M}\mathbf{x}'$, where $\mathbf{M}$ is a 3×3 matrix representing the motion of the frame and $\mathbf{x}'$ and $\mathbf{x}$ are the homogeneous, planar

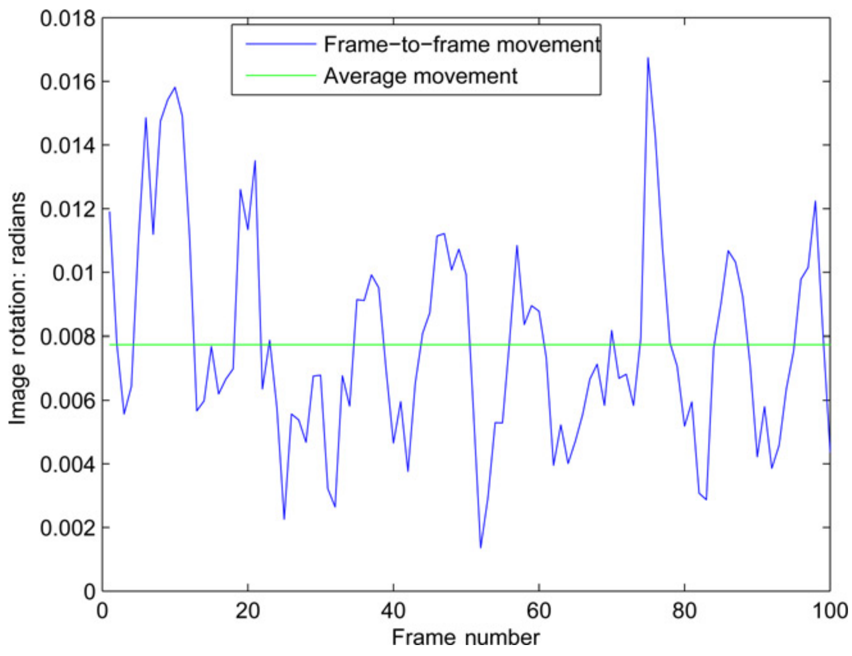


Fig. 11 This graph demonstrates the difference between the measured frame-to-frame motion of the video and the intended motion caused by the desired flight path of the MAV.

locations of a pixel before and after frame movement, respectively. To equate this notation with Eq. (1), we set

$$\mathbf{M} = \begin{bmatrix} s\cos\theta & -s\sin\theta & T_x \\ s\sin\theta & s\cos\theta & T_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Combining and removing motion from the video is achieved by performing simple matrix operations. Specifically, matrix multiplication can be used to combine different motions together. For example, given the motion from frame 1 to frame 2 ($\mathbf{M}_1^2$) and from frame 2 to frame 3 ($\mathbf{M}_2^3$), the motion from frame 1 to frame 3 is $\mathbf{M}_2^3\mathbf{M}_1^2$, a simple matrix multiplication. To remove motion, the inverse matrix of that motion is applied to the video. Therefore, to remove unwanted motion from a video, while applying the intended motion of the video, we apply a matrix $\mathbf{M}_s$ that is composed as

$$\mathbf{M}_s = \mathbf{M}_i\mathbf{M}_m^{-1} \tag{7}$$

where, $\mathbf{M}_m$ is the matrix representing all motion measured in the video up to the current video frame, $\mathbf{M}_i$ is the intended motion of the video until the current frame, and $\mathbf{M}_s$ is the motion that will be applied to the current image to stabilize the frame. Because $\mathbf{M}_s$ is composed of the inverse of all the motion and the intended motion, one can think of stabilization as removing all motion and adding intended motion to the displayed frame.

Unfortunately, the intended motion of a video is not known a priori even if the desired path of the MAV is known. This is because the motion in a video is derived from a nonlinear relationship between the location of the objects being observed, the desired movement of the camera, and the intrinsic characteristics of the camera. Because the location of each object being observed is not known beforehand, the intended video motion must be estimated from the frame-to-frame motion that was measured as described in Sec. III.C.

To understand the characteristics that an intended motion estimator should have, we analyze in more detail the equation for unwanted motion removal introduced in Eq. (7). First, note that $\mathbf{M}_i$ is the matrix that denotes how much objects in the video should move on the screen. At first glance, it might appear that $\mathbf{M}_i$ should be set equal to the identity matrix so that the objects in the video never move. However, because $\mathbf{M}_s$ is the motion that is actually applied to the video frame currently being displayed, if the motion in $\mathbf{M}_m$ is large and $\mathbf{M}_i$ is the identity matrix (in an attempt to minimize the motion of all objects in the image), then a large motion will be applied to the current frame being displayed. Because of this large motion, the frame may not even appear on the mobile ground station computer screen. Therefore, a tradeoff exists between removing the motion of objects in the video and maintaining the objects in view of the operator.

To estimate the intended motion that should be displayed on the screen, a low-pass finite impulse response (FIR) filter is applied to each of the four components (rotation, scaling, and the x and y translations) of the frame-to-frame motion. The goal is to design a low-pass filter that effectively estimates the intended motion in the video, while removing high-frequency motion with minimal delay. Reducing delay is critical because a large filter delay could cause the video to leave the screen before the filter estimates it is there. An example of video stabilization using a filter with too much delay is shown in the attached video [Click here to run TooDelayed].

In the subsections that follow, we describe our methodology for measuring the high-frequency rejection qualities of a low-pass FIR filter. We then present results which demonstrate the effects of modifying key parameters used in the filter design. Using this information, we present an effective filter design methodology given a constraint on the maximum allowable delay, which is specified directly by the number of FIR taps.

1. *Methodology for Evaluating Filter Effectiveness*

The filter performance in removing unwanted motion from a video stream depends on two factors: 1) the frequency characteristics of the filter input and 2) the effect of the frequency response of the filter's output on the movement of objects on the computer screen. To model the frequency characteristics of MAV video motion, we collected three videos from MAVs executing typical flight patterns. The first video shows the MAV orbiting a known GPS location, keeping the gimballed camera aimed at that fixed location [Click here to run OrbitVideo]. The second video shows an MAV searching a desert area during a simulated search and rescue operation [Click here to run SearchRescueVideo]. The third video shows a flight over a suburban neighborhood with a significant number of frames that include the horizon [Click here to run Neighborhood]. For all three videos, the frame-to-frame motion was estimated and stored to a file. This stored frame-to-frame motion allows the performance of the filter to be evaluated on several different types of motion.

To determine how the output of a filter would affect the video displayed to the computer screen, we developed a quantitative measurement for movement of objects in the video. A set of virtual objects at frame k, annotated as o_i^k, were distributed throughout a video frame, starting at pixel location (5,5) and spaced at every 10 pixels. By multiplying the virtual objects by the stabilization matrix, $\mathbf{M}_s$, for that frame, the location of the virtual objects on the computer screen were obtained using $\acute{o}_i^k = \mathbf{M}_s^k o_i^k$. To determine the location of these objects in the next frame, the measured motion from frame k to frame $k+1$ was used. The location of the objects in frame $k+1$ on the screen are then computed using the stabilization matrix from frame $k+1$. Therefore, the locations of the virtual object on the screen at frame k and $k+1$ are

$$\acute{o}_i^k = \mathbf{M}_s^k o_i^k \tag{8}$$

and

$$\acute{o}_i^{k+1} = \mathbf{M}_s^{k+1}\mathbf{M}_m^{k,k+1} o_i^k \tag{9}$$

We then average across all the virtual points to obtain the average movement of objects in the video screen, as

$$\bar{m} = \frac{1}{N}\sum_{i=1}^{N} |\acute{o}_i^{k+1} - \acute{o}_i^k| \tag{10}$$

The average movement of points in the video, $\bar{m}$, gives us a quantitative measurement of the filter effectiveness.

2. *Effect of Varying Filter Parameters on the Performance of MAV Video Stabilization*

Using the performance metric introduced above, the effect of varying different filter design parameters was evaluated. In Fig. 12, we show a basic diagram of the frequency response of a low-pass filter. The key parameters that affect the performance of the video stabilization routine include: 1) the pass-band frequency ω_{pb}, 2) the stop-band frequency ω_{sb}, and 3) the stop-band attenuation. Note that all low-pass FIR filter design methodologies attempt to approximate the idealized frequency response shown in Fig. 12, but are limited by the number of taps. Thus, the number of taps is a fundamental characteristic of any low-pass FIR filter design.

In Fig. 13, we show the effects of varying these three parameters on the movement of objects within the video. To create a low-pass filter, the Parks-McClellan filter design tool in MATLAB was used to approximate the desired frequency response for a given number of FIR filter taps. To understand the effect of the different parameters—stop-band frequency, pass-band frequency, and number of FIR taps—we modified the values that were input to the algorithm until the actual frequency response of the filter gave the numbers shown on the horizontal axes of the plots in Fig. 13. In each figure, the blue line represents the average movement (measured in pixels) of the virtual objects, and the magenta line represents the delay of the filter (measured in frames). Notice that for both stop-band attenuation and frequency, to improve the performance of the filter (decrease pixel movement), the number of taps needs to

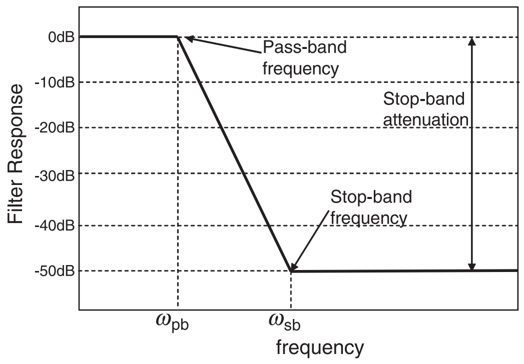


Fig. 12 The general frequency response characteristics of a low-pass FIR filter.

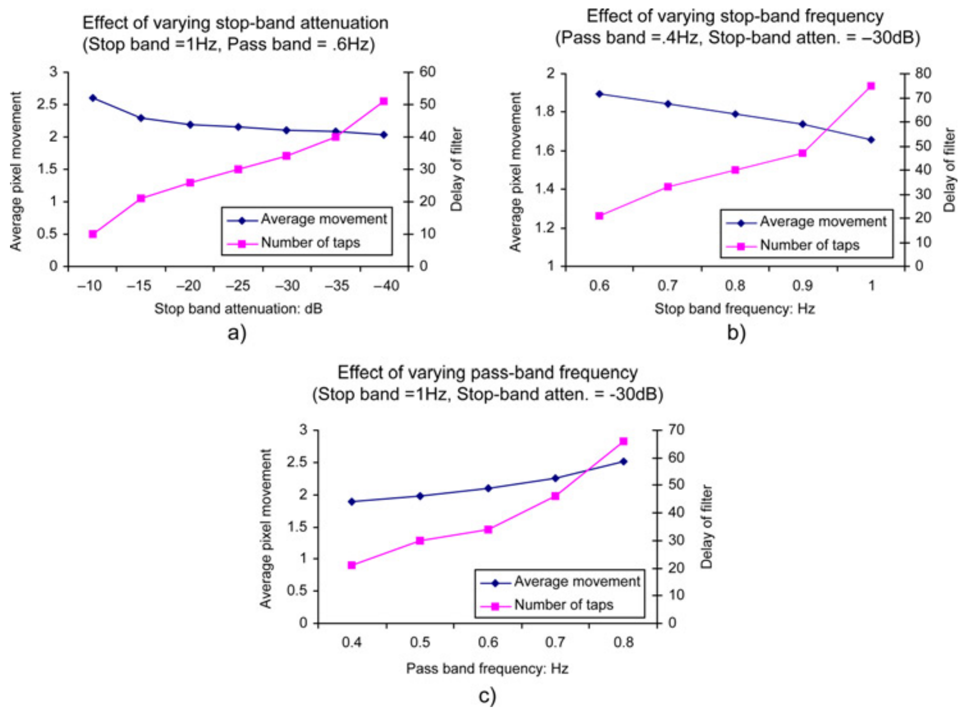


Fig. 13 Three charts showing the effects of different low-pass filter parameters on video stabilization effectiveness. These charts show the different movement (in pixels) for varying (a) stop-band attenuation, (b) stop-band frequency, and (c) pass-band frequency.

be increased. Conversely, the filter performance improves as the pass-band frequency is decreased, which has an associated reduction in the number of filter taps. Therefore, when designing a low-pass filter for video stabilization, the desired pass-band frequency is set as low as possible to reduce movement and minimize delay.

3. Methodology for Designing MAV Video Stabilization Filters

Assuming a fixed number of FIR filter taps, varying the stop-band frequency has two opposing effects. Intuitively, one would expect that as stop-band frequency is decreased, the video stabilization performance should increase. However, for a fixed number of taps, decreasing the stop-band frequency causes a decrease in stop-band attenuation, thereby decreasing the video stabilization effectiveness. Because of these conflicting effects, there is an optimal stop-band frequency for a given number of taps and for a specific video stream.

The first step in designing a low-pass filter for MAV video stabilization is to determine the maximum allowable delay D of the filter. From our analysis of MAV video, the maximum allowable delay is 12 frames to avoid video leaving the field of view of the camera. Once D is determined, the number of taps in the filter is set to $2D - 1$. Given this fixed number of filter taps, different filters can be designed using several stop-band frequencies. The effectiveness of each of these filters can then be measured using the performance metric introduced in Eq. (10). Using this information, the optimal stop-band frequency for a given delay D can be determined. The performance of this filter design methodology is discussed in Sec. V.

Given a maximum allowable delay of 12 frames, we used our methodology to design several filters. The performance of these filters, working on the three video streams mentioned in section Sec. III.D.1, is shown in Fig. 14. Note that for each video stream, there is a stop-band frequency value that produces a minimum amount of object movement in the video. Because there is a different optimum value for each video stream, we also observed that the effects of choosing too high of a stop-band frequency are less detrimental than choosing too low of a frequency. Therefore, a value of 1.1 Hz was chosen for the stop-band frequency.

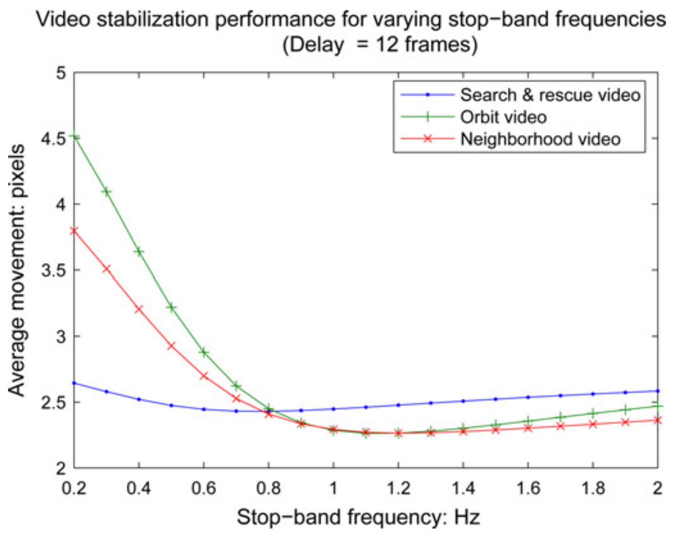


Fig. 14 The effect of varying stop-band frequencies on filters applied to three different video streams. Given the fixed number of taps in the FIR filter, there is a stop-band frequency that produces the minimum movement for each video stream.

E. Video Display

The final step in the stabilization process is displaying the video on the mobile ground station. At this point in the process, the estimated frame-to-frame motion of the video stream has been calculated. This estimated motion and the intended video motion found using the technique discussed in Sec. III.D are used to calculate the unwanted jitter motion, which is used to compute a reverse warping that is applied to each image prior to being displayed. The final result is the stabilized video stream displayed on the mobile ground station for the human operator.

IV. Target Localization

As observed by Ratches et al. [1], completely autonomous target recognition is not possible now or in the foreseeable future. However, stabilized video allows a human operating the ground station to better perform target recognition and identification. To localize a target, the input received from the operator must be synchronized properly with telemetry data to generate screen coordinates, which must be transformed to stabilized image coordinates and then transformed into a ray on which the target lies. The point which minimizes the distance to a series of these rays is then used to estimate the target location in world coordinates.

A. Synchronization

To demonstrate the importance of video and telemetry synchronization for target localization, Fig. 15 shows results of localizing a target with and without synchronization, where the units of the plots are meters. Fig. 15(a) shows the individual estimates of target location with no synchronization and Fig. 15(b) shows the individual estimates of target location with full synchronization. It can be seen that synchronization creates a distinct grouping of estimates near the actual target. The mean error for target location estimates without synchronization is 4.1 m and with synchronization the mean error is 2.3 m.

B. Operator Input

Synchronization allows the ground station properly to combine video and telemetry data for use by human operators in finding and tracking objects of interest. We developed and explored three methods for the operator to interact with the synchronized video stream when tracking targets. These three methods are hold-and-follow, selection area, and click-to-follow. We determined that click-to-follow with feature tracking to be the optimal balance between operator-workload and tracking accuracy. This method gives the operator a real-time ability to select targets to track in the MAV video and is designed for tracking high-velocity moving targets.

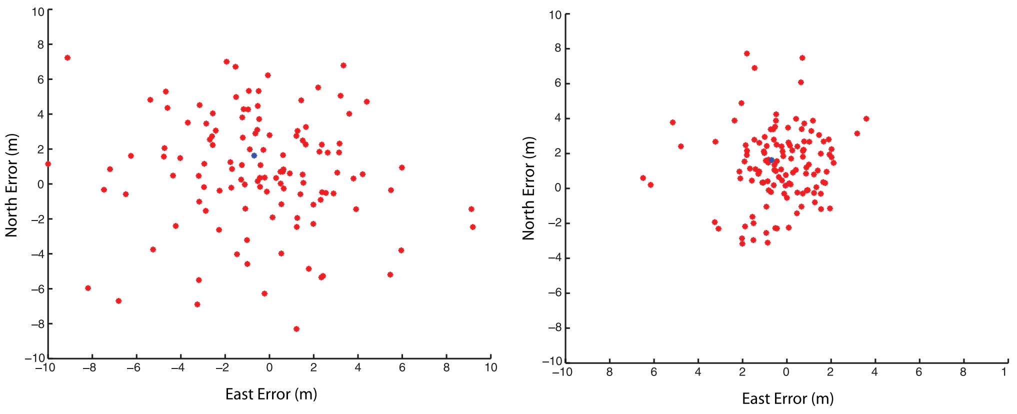


Fig. 15 Subplot (a) shows localization results without synchronization, and subplot (b) shows localization results using synchronization.

Rather than rely on the frame-to-frame motion, this method uses the feature motion vectors near the identified target to update the location of the target in the image. This reduces the drift caused by the errors introduced from estimating the video motion using the affine model. As the exact size and shape of the target is unknown, a rectangle is used to select the features whose motion vectors will be used in the estimate. The estimate is further enhanced using the previously discussed outlier data rejection methods.

C. Screen to Image Transformation

To estimate the GPS location of the identified target, the operator-input must be converted from screen coordinates to image coordinates. This transformation is completed in the video handling pipeline. First, let s_R be the scale factor between the screen size in pixels and the size of the rendering context, where the rendering context is the software window used to display the image on the computer screen. Let s_I be the scale factor between the rendering context and the rendered image size, and t_x and t_y are the position of the image in the rendering context. The transformation matrix T_s^I is then defined as

$$T_s^I = \begin{bmatrix} s_R & 0 & 0 \\ 0 & s_R & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_I & 0 & t_x \\ 0 & s_I & t_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_I & 0 & s_R t_x \\ 0 & s_I & s_R t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{11}$$

In this derivation it is important to understand that the screen, rendering context, and image are all referenced from the upper left-hand corner, thus a transformation of the axes is unnecessary.

To compensate for stabilization, the resulting coordinates must be transformed between intended video motion and modeled video motion. Since both of these motions are two-dimensional (2-D) transformations involving scaling, rotation, and x and y translation, the transformation between the intended video motion T_I, and the measured video motion T_M, can be found by

$$T_I^M = T_M {T_I}^{-1} \tag{12}$$

Therefore, $T_M {T_I}^{-1}$ transforms points from T_I to T_M. Since T_I is removed from T_M during the display process, T_I can be treated as the origin and T_I^M is the transformation used to map the operator-input from the image coordinate frame to the stabilized image coordinate frame. After this transformation, the ray in world coordinates on which the target lies can be found.

D. Image to World Ray Transformation

The transformations used to generate a world ray from the image coordinates are based on the transformations derived by Redding [15] and Barber et al. [16]. The origin of the ray is the current location of the camera in the world, found using four transformations: inertial frame to vehicle frame T_I^v, vehicle frame to body frame T_v^b, body frame to gimbal frame T_b^g, and gimbal frame to camera frame T_g^c.

Defining x_{MAV} as the north component, y_{MAV} as the east component, and z_{MAV} as the altitude component of the MAV's GPS location, the transformation from inertial to vehicle frame is

$$T_I^v = \begin{bmatrix} 1 & 0 & 0 & x_{\text{MAV}} \\ 0 & 1 & 0 & y_{\text{MAV}} \\ 0 & 0 & 1 & -z_{\text{MAV}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Letting ϕ, θ, and ψ denote the MAV's roll, pitch, and yaw angles, the transformation between the vehicle and body frames is defined as

$$T_v^b = R_\phi R_\theta R_\psi, \tag{14}$$

where

$$R_\phi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_\theta = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad R_\psi = \begin{bmatrix} \cos\psi & \sin\psi & 0 & 0 \\ -\sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{15}$$

To obtain the transformation from body to gimbal frame, we let θ_{gim} and ψ_{gim} be the pitch and yaw of the gimbal relative to the body frame. We also define x_{gim}, y_{gim}, and z_{gim} as the offset of the gimbal from the center of mass of the MAV. Thus, the transformation from body to gimbal frame is defined as

$$T_b^g = R_{\theta,\text{gim}} R_{\psi,\text{gim}} Tr_b^g \tag{16}$$

where

$$R_{\theta,\text{gim}} = \begin{bmatrix} \cos\theta_{\text{gim}} & 0 & -\sin\theta_{\text{gim}} & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta_{\text{gim}} & 0 & \cos\theta_{\text{gim}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

$$R_{\psi,\text{gim}} = \begin{bmatrix} \cos\psi_{\text{gim}} & \sin\psi_{\text{gim}} & 0 & 0 \\ -\sin\psi_{\text{gim}} & \cos\psi_{\text{gim}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{18}$$

$$Tr_b^g = \begin{bmatrix} 1 & 0 & 0 & x_{\text{gim}} \\ 0 & 1 & 0 & y_{\text{gim}} \\ 0 & 0 & 1 & z_{\text{gim}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

The offset of the gimbal-mounted camera is accounted for by defining x_{cam}, y_{cam}, and z_{cam} as the offset of the camera from the center of mass of the gimbal. Therefore, the camera offset transformation T_g^c can be written as

$$T_g^c = \begin{bmatrix} 0 & 0 & -1 & x_{\text{cam}} \\ 0 & 1 & 0 & y_{\text{cam}} \\ 1 & 0 & 0 & z_{\text{cam}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

T_g^c also flips the x and z axes to allow for a more intuitive image coordinate definition: x and y are the coordinates of the image and z is the depth into the image. The translational component of the resulting transformation matrix is the current world location of the camera and becomes the origin of the ray.

Obtaining the direction of the ray requires knowledge of the camera. The Camera Calibration Toolbox described by Bouguet [26] is used to obtain the values needed to estimate the ray direction. We specify f_x and f_y as the focal length of the camera and o_x and o_y are the optical center of the camera in pixels. The camera calibration matrix is written as

$$C = \begin{bmatrix} 0 & f_x & o_x & 0 \\ -f_y & 0 & o_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

More complicated camera calibration matrices can be used to account for skewing and tangential distortions, but these effects are negligible for MAV video owing to the narrow angle lenses that are used.

Given the image coordinates of the operator input u_x and u_y, the direction of the ray can be defined as

$$V_{\text{ray}} = T_c^v C u, \tag{22}$$

where

$$T_c^v = (T_v^b T_b^g T_g^c)^{-1} \tag{23}$$

and

$$u = \begin{bmatrix} u_x & u_y & 1 & 1 \end{bmatrix}^T \tag{24}$$

and where V_{ray} is a ray in world coordinates that intersects the target. A sequence of rays can be collected as the MAV continues to move during a specified time span. The GPS location of the target can then be determined using a recursive least squares algorithm, as described by Barber et al. [16].

V. Results

A. Video Stabilization

To quantify the performance of the video stabilization algorithm, we define the mean movement per feature per frame (MMPFPF) metric for measuring frame-to-frame motion. This metric quantifies the motion of continuously visible features on the display.

Employing Harris corner detection to rate features and minimum separation distance to select which features to track, the affine model (alone and with RANSAC noise rejection methods) was evaluated on the three different video streams. Table 1 shows the MMPFPF statistically and Fig. 16 shows the MMPFPF graphically, both before and after stabilization. Note that in Fig. 16, all frame-to-frame motion estimates are calculated using the same feature motion vectors and therefore share the same MMPFPF before stabilization. Because the camera is mounted on an aircraft, the mean μ of the intended motion is expected to be non-zero, but the standard deviation σ should be close to zero.

The results shown in Table 1 and Fig. 16 demonstrate quantitatively our algorithm's performance in removing jitter from the MAV video stream presented to the display screen. The motion estimate from the affine model dramatically reduces the MMPFPF by itself, but an additional reduction of 10% in MMPFPF was achieved using the RANSAC method in conjunction with the standard affine model. Similar results were seen on all three MAV videos tested without changing the parameters of the stabilization filter for different video types, thus demonstrating the wide applicability of the technique for stabilizing MAV video.

One of the key measures of the effectiveness of our frame motion estimation technique is a qualitative assessment of how well two images in the video align with each other. An example of this can be seen in Fig. 17, which shows two images, four frames apart in the video, with one drawn transparently on top of the other at half-brightness. Notice

Table 1 Mean and standard deviation of MMPFPF

Method	μ of MMPFPF	σ of MMPFPF
No stabilization	6.66	2.49
Affine model	1.97	0.5
Affine model: RANSAC	1.82	0.41

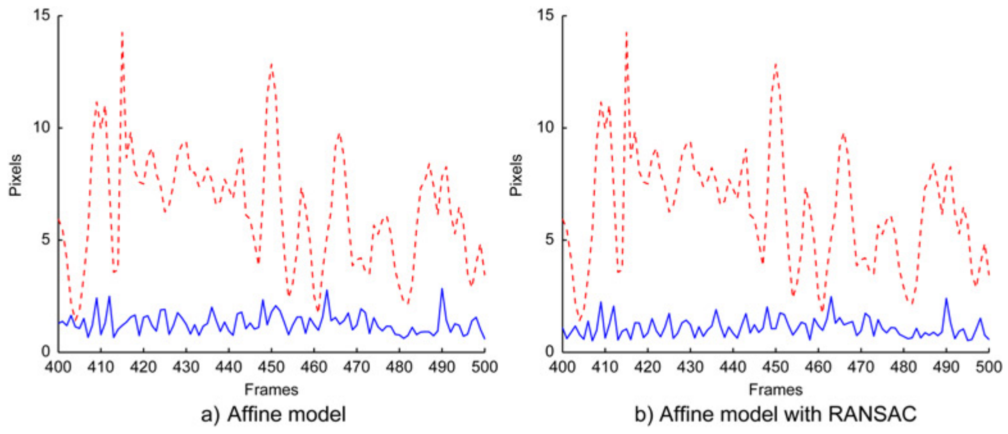


Fig. 16 The dashed line is the MMPFPF before stabilization and the solid line is the MMPFPF after stabilization using the affine motion model, with and without RANSAC. Our MAV video stabilization routine dramatically decreases movement in the video footage.

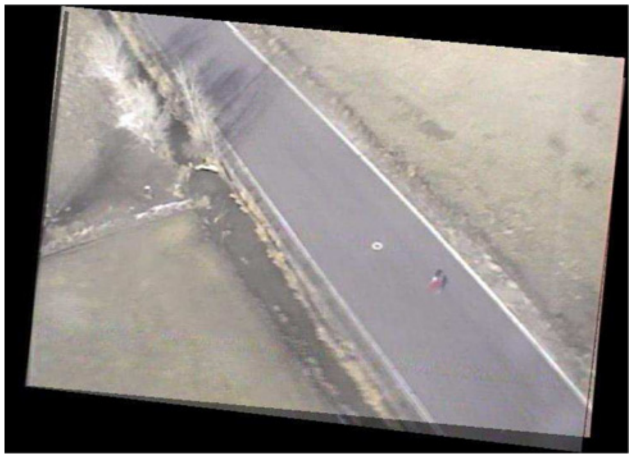


Fig. 17 Two registered video images, one overlaid upon the other, demonstrating proper frame motion estimation.

that although the second image extends below the first, the features in the image are correctly aligned. If they were misaligned, it would be evidenced by seeing double features such as two manhole covers, two lines down each side of the road and so on.

Although the MMPFPF metric shows quantitatively that our MAV video stabilization technique is highly effective, it is perhaps more convincing to see the results on actual video footage. To demonstrate the effectiveness of selecting a stop-band frequency using our methodology, we have created three videos. The first video shows the raw footage [Click here to run OrbitVideo]. The next two videos show the effect of varying stop-band frequency on the stabilization performance. The second video was created using a stop-band frequency set to 0.2 Hz [Click here to run PointTwoHz] and the third video was created using a stop-band frequency of 1.1 Hz [Click here to run OnePointOneHz]. Qualitatively, one can see that the amount of movement in the second video is marginally less than the first, but the third video (1.1 Hz stop-band frequency) has much less movement. Watching the third video, it is evident that the jitter is nearly eliminated, thus demonstrating the effectiveness of our filter design methodology in stabilizing MAV video with a fixed delay.

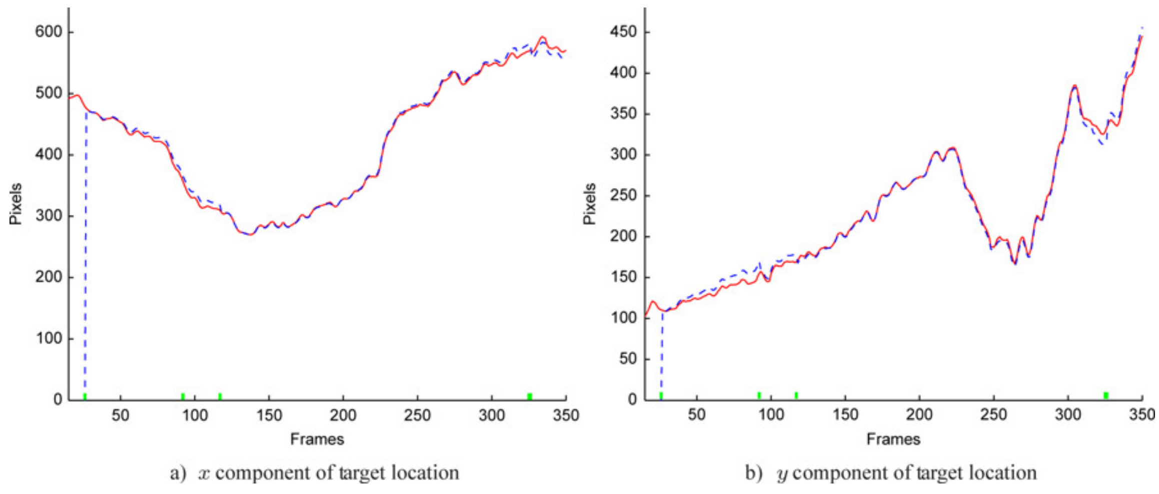


Fig. 18 The solid line is the manually identified target location in the MAV video and the dashed line is the target location identified by the operator. The bar at the base of the figure shows when there was user interaction.

B. Localization

The benefits of synchronization were discussed in Sec. IV.A and the effectiveness of the localization techniques have been described by Redding [15] and Barber et al. [16]. Therefore, the effectiveness of the developed algorithms to minimize operator input while identifying and localizing a specified target is presented in this section. For the analysis, truth data are generated by manual identification of the target position in the video. Figure 18 shows the tracking achieved by the click-to-follow method with feature tracking in the x and y-direction. The mean error with this video is 4.22 pixels in the x-direction and 4.47 pixels in the y-direction. The four small bars at the base of both (a) and (b) in Fig. 18 indicate time duration of operator interaction with the video. Obviously, the required operator interaction to maintain target tracking is quite small.

As with the MAV video stabilization, the effectiveness of the target localization algorithm can be seen quantitatively in Fig. 18. Additionally, the performance of the combined stabilization and tracking algorithms can be seen qualitatively in the video [Click here to run StabilizedandTracking].

VI. Conclusions

This paper presents a computer software and hardware architecture that was built specifically for developing MAV computer vision technologies. The paper presents two such technologies: MAV video stabilization, and target localization based on synchronized MAV telemetry and video. The effectiveness of MAV video stabilization is demonstrated through the reduction in MMPFPF. The stabilized video was synchronized with MAV telemetry data to localize moving and stationary ground targets with the human operator specifying which objects were to be tracked as targets.

One contribution of this paper is the description of a computer vision platform for MAV technology, making use of multi-core and simultaneous multi-threading architectures to enable real-time processing of MAV video. The second contribution is a technique for estimating actual and intended video motion in real-time to effectively stabilize the MAV video. These innovations include feature rating methods and enhanced feature selection and tracking techniques for MAV video. All of these enhancements have been shown to enable the stabilization of MAV video in real-time on a mobile ground station. The resulting stabilization enables the third contribution, the novel click-to-follow with feature tracking method, which allows for high-level control of a MAV to track moving targets. The most important contribution of this paper is the demonstration that small MAVs are capable of handling complex, real-time computer vision tasks for surveillance and reconnaissance missions.

As shown by Cardoze et al. [4], one possible use for the stabilization scheme would be to detect moving targets in the video by applying the RANSAC rejection techniques to extract the feature motion vectors associated with each moving object, further reducing the reliance on human input. Active contours [27] and CONDENSATION [28] present the possibility of improving the target tracking methodology presented in this paper, but would require additional development to facilitate model identification from a real-time video stream.

The use of a full 3-D motion model could more accurately describe the motion seen in MAV video, but would require a different display approach that does not rely on a 2-D mosaic technique. The integration of telemetry data into the intended video motion estimation process will improve accuracy and will also allow for reliable estimation of three-dimensional intended video motion with Euclidean distance. To achieve integration, however, will require development of a filtering technique to account for transmission delays and creation of a vehicle state estimation scheme to determine the MAV's position between received telemetry samples. One possible approach would be to use Kalman filtering techniques to resolve these telemetry data problems, making possible rather significant enhancements to the intended video motion estimation.

Acknowledgments

This work was partially funded by NASA under STTR contract No. NNAOYAA14C to Scientific Systems, Inc. and Brigham Young University, by AFOSR Young Investigator grant number FA9550-07-1-0167, and by a Utah State Centers of Excellence grant. The second author was sponsored by the United States Air Force through its Academy Instructor pipeline program.

References

[1] Ratches, J. A., Walters, C. P., Buser, R. G., and Guenther, B. D., "Aided and Automatic Target Recognition Based upon Sensory Inputs from Image Forming Systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 9, 1997, pp. 1004–1019.
doi: 10.1109/34.615449

[2] der Wal, G. V., Hansen, M., and Piacentino, M., "The Arcadia Vision Processor," *CAMP '00: Proceedings of the Fifth IEEE International Workshop on Computer Architectures for Machine Perception*. IEEE Computer Society, Washington, DC, 2000.

[3] Darmanjian, S., Arroyo, A. A., and Schwartz, E. M., "An Alternative Real-time Image Processing Tool," *Florida Conference on Recent Advances in Robotics*, 2003, available at http://www.mil.ufl.edu/publications/fcrar03/Image_Processing_Tool.pdf.

[4] Cardoze, D. E., Collins, T. R., and Arkin, R. C., "Visual Tracking Technologies for an Autonomous Rotorcraft," submitted to Image and Vision Computing Journal, 2005, available at http://www.cc.gatech.edu/grads/c/David.Cardoze/journal2_4.ps.

[5] Chang, H.-C., Lai, S.-H., and Lu, K.-R., "A Robust and Efficient Video Stabilization Algorithm," *ICME '04: International Conference on Multimedia and Expo, 2004*, Vol. 1, Institute of Electrical and Electronics Engineers, New York, NY, 2004, pp. 29–32.

[6] Ratakonda, K., "Real-time Digital Video Stabilization for Multi-media Applications," *ISCAS '98: Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, Vol. 4, Institute of Electrical and Electronics Engineers, Monterey, CA, 1998, pp. 69–72.

[7] Buehler, C., Bosse, M., and McMillian, L., "Non-Metric Image-based Rendering for Video Stabilization," 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), Vol. 2, IEEE Computer Society, Kauai, HI, 2001, pp. 609–614.

[8] Duric, Z., and Rosenfeld, A., "Shooting a Smooth Video with a Shaky Camera," *Machine Vision and Applications*, Vol. 13, No. 5-6, 2002, pp. 303–313.
doi: 10.1007/s00138-002-0085-y

[9] Gaskill, D. M., "Techniques for Synchronizing Thermal Array Chart Recorders to Video," *International Telemetering Conference*, Vol. 28, International Foundation for Telemetering, San Diego, CA, 1992, pp. 61–64.

[10] Zeng, S., Powers, J. R., and Hsiao, H., "A New Video-synchronized Multichannel Biomedical Data Acquisition System," *IEEE Transactions on Biomedical Engineering*, Vol. 47, No. 3, 2000, pp. 412–419.
doi: 10.1109/10.827316

[11] Anderson, D. P., and Stump, B. W., "Synchronization of Video with Seismic and Acoustic Data using GPS Time Signals," *Internet*, [Online Journal], http://www.geology.smu.edu/∼dpa-www/gps_video/index.html, accessed June 2006.

[12] Rieger, J. L., "Encoding of Telemetry Data in a Standard Video Channel," *International Telemetering Conference*, International Foundation for Telemetering, Los Angeles, CA, 1977, pp. 151–155.

[13] Zhang, J.-G., "Using Advanced Optical Multiple-access Techniques in High-speed Avionic Local Area Networks for Future Aircraft Applications. Part II: Optical Time-division Multiple-access Networks," *Instrument Society of America Transactions*, Vol. 36, No. 4, 1997, pp. 321–338.

[14] Walrod, J., "Using the Asynchronous Transfer Mode in Navy Communications," *Sea Technology*, Vol. 38, No. 5, 1997, p. 6.
[15] Redding, J., "Vision-based Target Localization from a Small Fixed-wing Unmanned Air Vehicle," Master's thesis, Brigham Young University, 2005.
[16] Barber, D. B., Redding, J. D., McLain, T. W., Beard, R. W., and Taylor, C. N., "Vision-based Target Geo-location Using a Fixed-wing Miniature Air Vehicle," *Journal of Intelligent and Robotic Systems*, Vol. 47, No. 4, 2006, pp. 361–382.
[17] Castleman, K. R., *Digital Image Processing*, Upper Saddle River, NJ, 1996.
[18] Canny, J., "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, 1986, pp. 679–698.
[19] Förstner, W., "A Framework for Low Level Feature Extraction," *ECCV '94: Proceedings of the Third European Conference on Computer Vision*, Vol. II. Springer-Verlag, New York, NY, 1994, pp. 383–394.
[20] Harris, C. G., and Stephens, M., "A Combined Corner and Edge Detector," *4th Alvey Vision Conference*, 1988, pp. 147–151.
[21] Saeedi, P., Lawrence, P. D., and Lowe, D. G., "Vision-based 3D Trajectory Tracking for Unknown Environments," *IEEE Transactions on Robotics and Automation*, Vol. 22, No. 1. 2006, pp. 119–136.
[22] Lowe, D., "Object Recognition from Local Scale-Invariant Features," *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference*, Vol. 2, 1999, available at http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=790410.
[23] Lowe, D. G., "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, Vol. 60, No. 2, 2004, pp. 91–110.
doi: 10.1023/B:VISI.0000029664.99615.94
[24] Björck, Å., *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, 1996.
[25] Fischler, M. A., and Bolles, R. C., "Random Sample Consensus: a Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, Vol. 24, No. 6, 1981, pp. 381–395.
doi: 10.1145/358669.358692
[26] Bouguet, J.-Y., "Camera Calibration Toolbox for MATLAB," 2004, available at http://www.vision.caltech.edu/bouguetj/calib_doc/.
[27] Kass, M., Witkin, A., and Terzopoulos, D., "Snakes: Active Contour Models," *International Journal of Computer Vision*, Vol. 1, No. 4, 1988, pp. 321–331.
doi: 10.1007/BF00133570
[28] Isard, M., and Blake, A., "Condensation-Conditional Density Propagation for Visual Tracking," *International Journal of Computer Vision*, Vol. 29, No. 1, 1998, pp. 5–28.
doi: 10.1023/A:1008078328650

Ella Atkins
Associate Editor