Mechatronics 19 (2009) 1043–1056

Contents lists available at ScienceDirect

Mechatronics

journal homepage: www.elsevier.com/locate/mechatronics

A hardware in the loop simulation platform for vision-based control of unmanned air vehicles

N.R. Gans [a,*], W.E. Dixon [b], R. Lind [b], A. Kurdila [c]

[a] University of Texas at Dallas, Richardson, TX, USA
[b] University of Florida, Gainesville, FL, USA
[c] Virginia Polytechnic Institute, Blacksburg, VA, USA

ARTICLE INFO

Keywords:
Hardware in the loop simulation
Vision-based control
Unmanned air vehicles

ABSTRACT

Design and testing of control algorithms for unmanned air vehicles (UAV's) is difficult due to the delicate and expensive nature of UAV systems, the risk of damage to property during testing, and government regulations. This necessitates extensive simulation of controllers to ensure stability and performance. However, simulations cannot capture all aspects of a flight control, such as sensor noise and actuator lag. For these reasons, hardware in the loop simulation (HILS) platforms are used. In this paper, a novel HILS platform is presented for vision-based control of UAV's. This HILS platform consists of virtual reality software to produce realistic scenes projected onto a screen and viewed by a camera. Flight hardware includes an UAV with onboard autopilot interfaced to the virtual reality software. This UAV can be mounted in a wind tunnel, allowing attitude regulation through servoing the airfoils.

1. Introduction

The development of a control system typically involves a period of simulation. Simulation provides a precise environment, access to physically unmeasurable variables and rapid redesign and testing, not to mention sparing wear and damage on equipment. Additionally, some systems may present a danger in the event of system failure. However, simulations often fail to capture critical issues. Issues not captured in simulation are often a matter of robustness, but also include problems such as unknown dynamics or errors in simulation code that cause inaccurate performance. Some problems may be known but difficult to model accurately.

For several decades, simulation and implementation has been bridged through the use of Hardware In the Loop Simulation (HILS). HILS combines a simulated system with physical hardware. For example, a software simulation of the system plant is augmented with actuators and sensors from the designed system. HILS systems have facilitated development in numerous fields, including automotive engineering [1,2], aerospace [3–5], power systems [6], manufacturing [7] and robotics [8,9].

For the past three years, a joint effort has been underway to develop a sophisticated simulation testbed for the vision-based control of Unmanned Air Vehicles (UAV's). This testbed provides multiple stages of increasing hardware interaction. The first stage is a virtual reality system capable of displaying environments and simulating the dynamics of various plant models, including UAV's. The second stage is a system of modular displays, digital cameras and computers for image and control processing. The third stage incorporates a fixed-wing UAV in a wind tunnel. The UAV is equipped with IMU motions sensors and an autopilot to actuate the airfoils.

Vision-based control systems can be designed and tested in the virtual reality stage. Once the system is satisfactory, the same 3D environment can be projected onto large monitors and viewed by a physical camera. Problems associated with cameras, such as signal noise, lens distortion, etc. are now incorporated. Additionally, multiple cameras can be rapidly developed and tested for use in the field. Communication between the camera and control processing computers and the environment rendering computers allows closed loop control of the virtual scene. In the final stage, the control signals generated by the vision controller are sent to the UAV in the wind tunnel. Data measured by avionics, such as attitude and wind speed, are fed back to the computer system and rendered in the virtual environment. An illustration of the complete HILS environment is shown in Fig. 1. There are currently two such HILS facilities. The first facility is at the University of Florida Research and Education Engineering Facility (REEF), in Shalimar, Florida. The second facility, which does not currently incorporate a wind tunnel, is at the University of Florida main campus in Gainesville, Florida.

To the authors' knowledge, this work presents the first development of a HILS system for UAV vision-based control design that incorporates camera hardware, flight avionics, and an airframe in a wind tunnel. HILS is often performed using the avionics and

* Corresponding author. Tel.: +1 850 833 9350; fax: +1 850 833 9366.
E-mail address: nrgans@gmail.com (N.R. Gans).

0957-4158/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mechatronics.2009.06.014

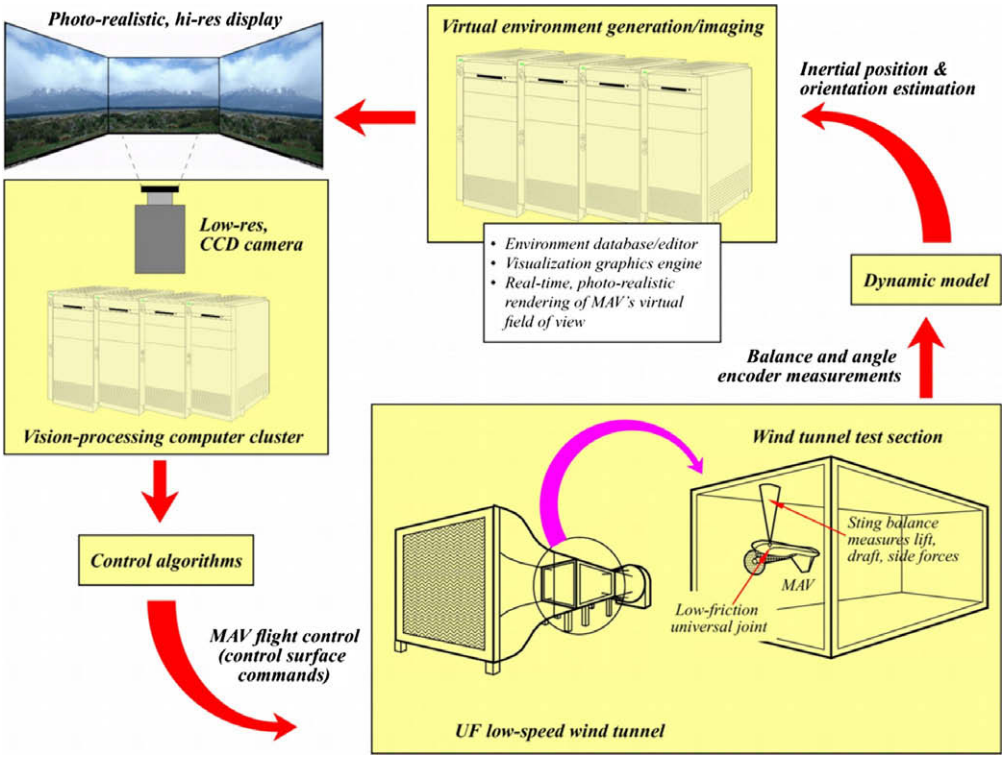


Fig. 1. The vision-based hardware-in-the-loop simulation environment.

autopilot to receive simulated flight data [10]. The authors of [11,12] measured the aerodynamic forces on an airfoil in a wind tunnel while simulating inertia of the wing, creating a closed loop "wing-in-the-loop" simulation. They recently extended this strategy to study behavior in unsteady regimes, such as a pitching airfoil at a high angle of attack and compared the results with established LPV models [13]. In [14] and [15], HILS systems are developed for vision-based UAV navigation, which incorporate flight avionics, ground station, and video processing hardware, however the images are generated by software without the use of cameras, and in both cases the aircraft is simulated using a nonlinear model. The authors of [16] designed a HILS system for vision-based UAV control involving a camera mounted on a robot manipulator that moved through a physical diorama.

Sections 2–4 discuss the three stages of the HILS platform in detail. Section 5 presents examples of the work being done at each stage.

2. Stage one – the virtual reality simulator

The first component is the virtual reality simulator which generates and displays the virtual environment. The virtual reality system is composed of multiple 3.6 GHz Xeon workstations running virtual environment (VE) rendering software by MultiGen-Paradigm. The VE software networks the workstations, allowing multiple instances of the VE to run on multiple workstations. The Gainesville facility has a cluster of five computers and a database server, currently capable of displaying on three simultaneous displays, while the REEF facility can has a cluster of eleven computers and five displays. Control routines can be written in C++ or Matlab/Simulink, and make use of the GNU Scientific Library (GSL) [17]. Running Matlab/Simulink simultaneously with image processing routines essentially creates a multithreaded application. There is no native way to pass information between Simulink and C++. It was necessary to create functions called by Simulink to read and write shared memory locations (in RAM) known to both C++ and Simulink. Sharing memory between C++ and Simulink requires the use of memory synchronization tools such as mutexes. Such tools are not complicated [18], but discussion is beyond the scope of this paper. The communication scheme is illustrated in Fig. 2.

The virtual reality simulator utilizes MultiGen–Paradigm's Vega Prime, a commercial software package for Windows. There are currently two virtual environments available. The first is a scale accurate model of the University of Florida (UF) campus, which

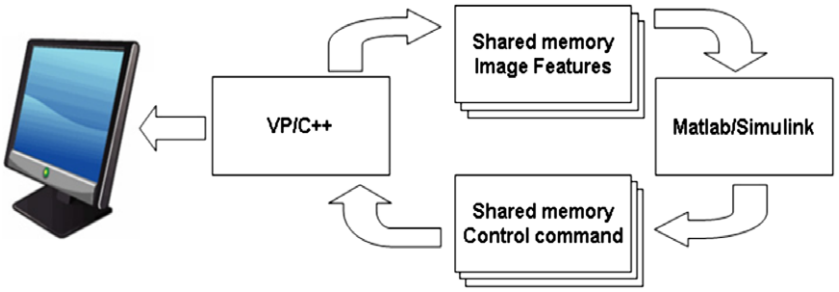


Fig. 2. Communication between VE software, image processing, and control for no camera in the loop simulations.

(a) Gainesville HILS facility (b)REEF HILS facility

Fig. 3. Modular displays at the two HILS facilities.

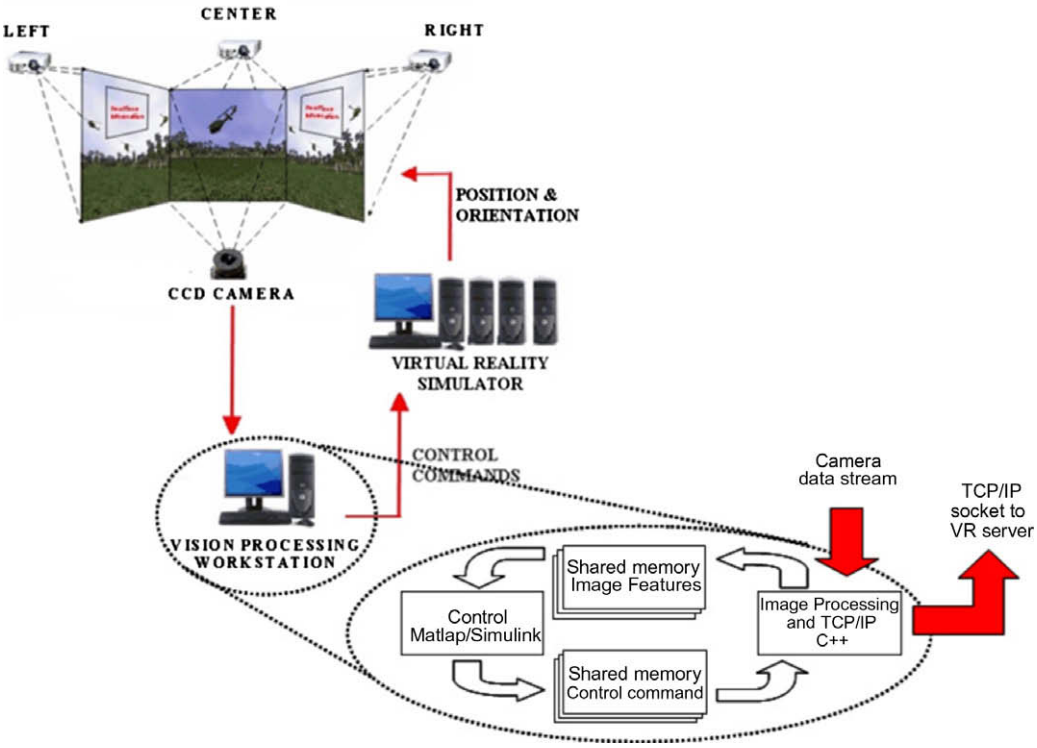


Fig. 4. Communication between VE software, camera, image processing, and control for camera in the loop simulations.

provides a large, urban environment. The second environment is a recreation of the US Army's urban warfare training ground at Fort Benning. While smaller in size, the second environment has a much denser polygon count, more detailed textures, and the effects of soft shadows, resulting in realistic images.

2.1. Camera projection model

At the heart of image based estimation and control is the camera projection model. The camera projection model describes how 3D points, lines, etc. in the environment are projected to 2D image features in digital image. A popular projection model is the pinhole camera model. The pinhole camera model is mathematically simple and accurate when using well focused lenses of reasonable quality.

In the pinhole camera model, an orthogonal coordinate system $\mathscr{F}$ is attached to a camera. Conventional orientation of $\mathscr{F}$ places the z-axis pointing ahead from the camera, and is often called the optical axis. Consider a collection of 3D feature points in front of the camera, denoted by $O_i, i \in \{1 \dots n\}, n \in \mathbb{R}^+$. The Euclidean coordinates of the feature points O_i, expressed in the frame $\mathscr{F}$, is denoted by $\bar{m}_i(t) \in \mathbb{R}^3$ and given by

$$\bar{m}_i \triangleq [x_i \; y_i \; z_i]^T. \tag{1}$$

The Euclidean-space is projected onto the image-space, giving the normalized coordinates of the targets points $\bar{m}_i(t)$, defined as

$$m_i = \frac{\bar{m}_i}{z_i} = \begin{bmatrix} \frac{x_i}{z_i} & \frac{y_i}{z_i} & 1 \end{bmatrix}^T. \tag{2}$$

In a digital camera, each target point has quantized pixel coordinates and homogeneous[1] pixel coordinate $p_i(t) \in \mathbb{R}^3$ expressed in the image coordinate frame

[1] Homogeneous pixel coordinates refers to the fact that the coordinates are intrinsically elements of $\mathbb{R}^2$, but extended to $\mathbb{R}^3$ by concatenating a 1 as a third element.

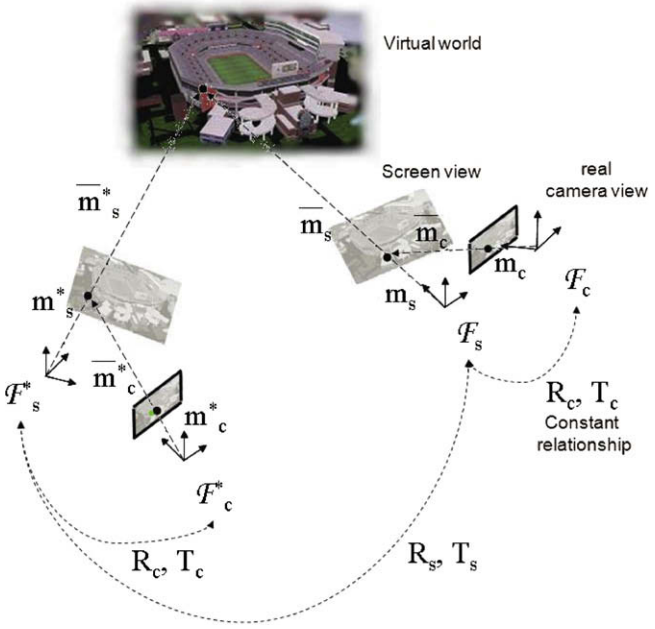


Fig. 5. Camera, projector plane and virtual scene geometry.

$$p_i \triangleq [u_i \;\; v_i \;\; 1]^T. \tag{3}$$

The pixel coordinates $p_i(t)$ are related by a global, invertible transformation to the normalized task-space coordinates $m_i(t)$

$$p_i = Am_i$$

where $A \in \mathbb{R}^{3\times 3}$ is the *intrinsic camera calibration matrix* given by

$$A = \begin{bmatrix} fs_x & s_\theta & u_x \\ 0 & fs_y & u_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

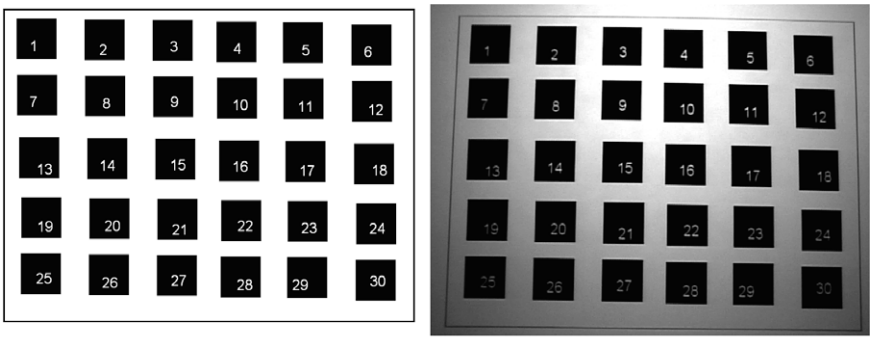
In (4), f is the camera focal length, s_x and s_y are related to the size and dimensions of the pixels elements, s_θ is a skew parameter for nonrectangular pixels, and u_x and u_y define the center of projection, in pixels. Collectively, these are referred to as the camera intrinsic parameters. In the case of a physical camera, the intrinsic parameters are often reported by the manufacturer, though a formal camera calibration should be performed to account for manufacturer defects. In the case of an image from a VE, the intrinsic parameters can be deduced from software settings. Given the coordinates p_i of a set of feature points at different times, there are numerous methods to solve for the relative displacement of the camera to the points, the motion of the points over time, the motion of the camera over time, etc. These methods include multi-view (epipolar) geometry [19–21], Kalman filtering [22–24] and nonlinear estimation [25,26]. Using this data in the feedback loop of a control system constitutes vision-based controls.

3. Stage two – camera in the loop

The second component of the HILS platform is the modular display and physical cameras. Since the VE can be instanced by several workstations simultaneously, multiple views can be rigidly connected in a mosaic for a large field of view. Alternately, multiple, independent camera views can pursue their own tasks, such as coordinated control of multiple vehicles. The display at Gainesville consists of a set of three rear projection displays as depicted in Fig. 3a, which shows a mosaiced view. The REEF has four plasma screens capable of mosaiced images and one projection display, as seen in Fig. 3b. Each camera feeds a video stream to a dedicated computer, which process the images and create control signals. Control signals are relayed to the VE rendering cluster over a network connection using TCP/IP sockets. Several cameras have been successfully fielded, including digital video cameras using Firewire and USB 2.0 interfaces, and analog video cameras in conjunction with D/A converters. Camera interface, image processing, VE rendering and socket communication among computers are implemented in C++. Control routines can be implemented in C++ or Matlab/Simulink, requiring the use of shared memory. This is illustrated in Fig. 4.

There is a notable complication when incorporating cameras into the HILS platform. Vision-based control algorithms assume that the camera is capturing an image of a 3D scene. This assumption holds for images captured from the frame buffer in a pure simulation, as described in Section 2. However, this assumption does not hold when a physical camera captures images of the VE rendered on a display. Clearly, the camera does not look at the 3D scene directly. The camera looks at a 3D scene that is projected onto a 2D display. Thus, the camera projection model is not valid in this case.

There exists an additional transform between the points on the screen and the points in the camera image. This transform must be incorporated in any vision-based control simulations with a camera in the loop. Geometry between the display, camera and virtual scene is illustrated in Fig. 5. The camera is fixed with respect to the projection plane, and any change of scene on the projector plane can be modeled as the display screen plane moving through the virtual environment, with the camera rigidly attached to it. The means to account for the *camera-to-screen geometry*, which



(a) Screen image of 1280×1024 resolution

(b) Camera image of 640×480 resolution

Fig. 6. Images used in camera-to-screen calibration.

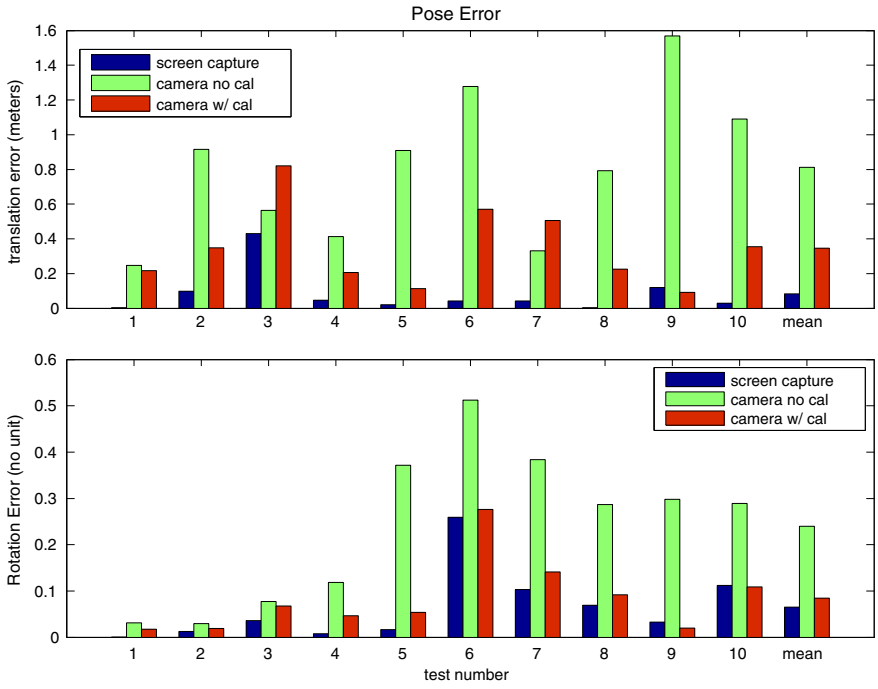


Fig. 7. Results of the camera-to-screen calibration test.

describes how features on the screen map to features in the camera image, are presented in this section.

3.1. Camera-to-screen geometry

The virtual environment is projected onto a virtual image which is reproduced on a physical display screen. This projection of the virtual world onto the screen is modeled by the pinhole projection model as described in Section 2.1. As seen in Fig. 5, consider the virtual camera frame $\mathcal{F}_s^*$ within the VE. The virtual camera undergoes a rotation R_s and translation x_s to a new pose in the VE denoted by frame $\mathcal{F}_s$. In the case of camera in the loop, a physical camera is looking at a screen onto which a 2D image is projected from 3D virtual scene. Consider a camera, with attached reference frame $\mathcal{F}_c$, viewing the display screen. $\mathcal{F}_c$ is separated from $\mathcal{F}_s$ by a constant rotation R_c and constant translation T_c.

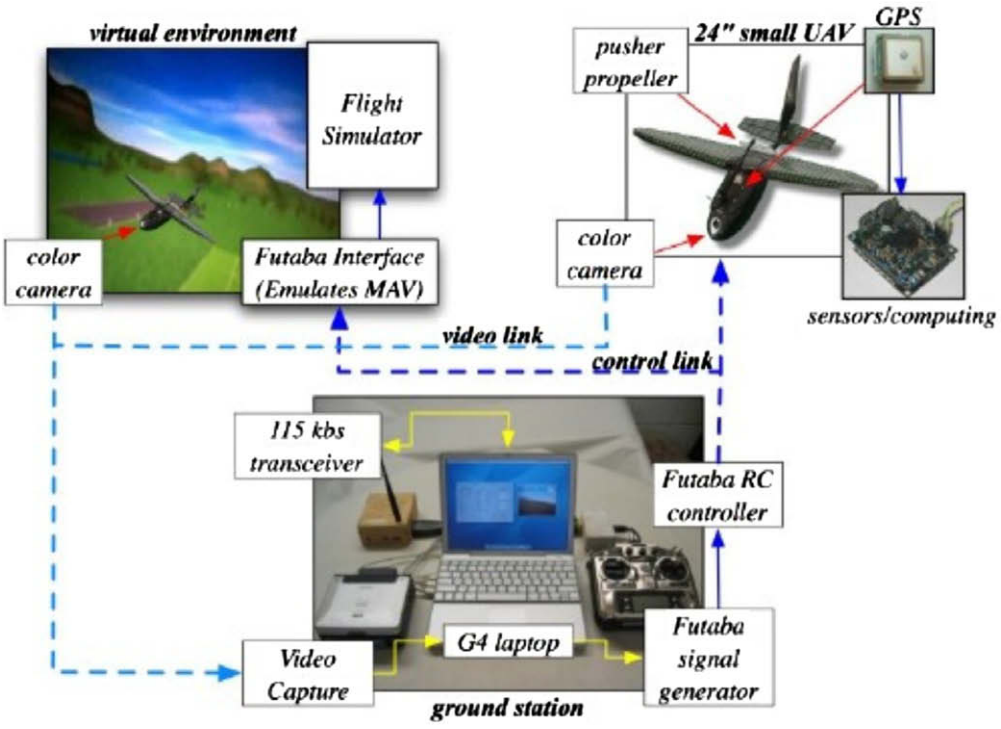


Fig. 8. Flight hardware configurations.

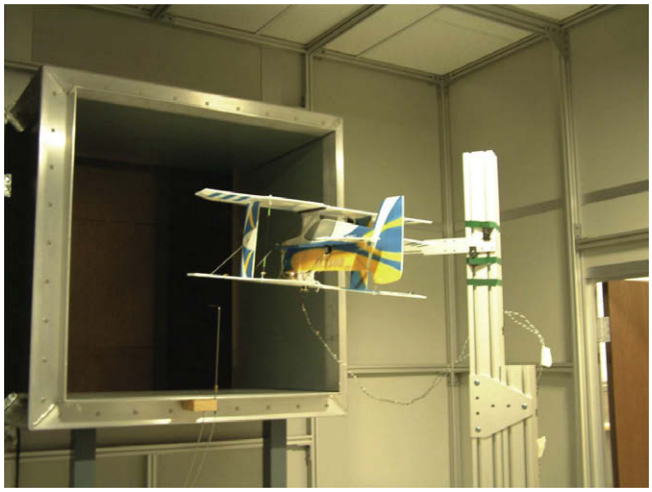


Fig. 9. E-flite aircraft in the wind tunnel.

A set of points $O_i, i \in \{1 \dots n\}, n \in \mathbb{R}^+$ on the display screen have homogeneous pixel coordinates in $\mathscr{F}_s$ given by

$$p_{si} = A_s m_{si}. \tag{5}$$

A_s is the intrinsic calibration matrix of the virtual camera, and can be determined from the settings of the VE rendering software. The same points have Euclidean coordinates $\overline{m}_{ci} \in \mathbb{R}^3$ in $\mathscr{F}_c$. The normalized coordinates of p_{ci} in the camera frame are given by

$$p_{ci} = A_c m_{ci} = A_c \frac{\overline{m}_{ci}}{z_{ci}}. \tag{6}$$

where A_c is the camera calibration matrix.

When the camera views the screen, every point on the screen corresponds to one and only one point in the camara image plane. Thus, there is a homography relationship between p_{ci} and p_{si} given by

$$p_{ci} = G_{csN} p_{si} i \in \{1 \dots n\}. \tag{7}$$

where $G_{csN} \in \mathbb{R}^{3\times3}$ is the *camera-to-screen homography matrix.*

The matrix G_{csN} can be solved for with a calibration method. A known calibration grid pattern is projected on the screen and a picture is taken with the camera. This is shown in Fig. 6. Extracting the corners of the grids gives a set of points p_{ci} in the image, and by matching them to the known p_{si} a set of linear equations can be solved for G_{csN}.

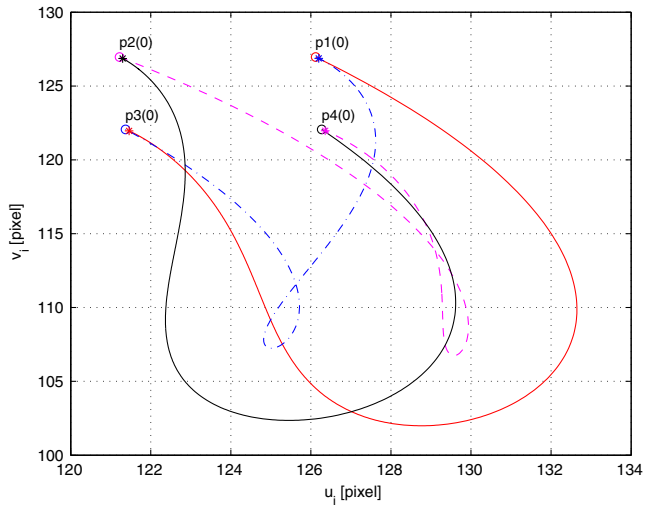
Measuring the coordinates of the points p_{ci} is subject to sensor noise and is thus random process. To eliminate the effects of noise, G_{csN} is estimated as the solution to linear equations using RANSAC [27]. RANSAC is an algorithm for robust fitting of models in the presence of uncertainty and data outliers. It returns a best fit of G_{csN} to p_{ci} and p_{si}, but eliminates any points that appear to be corrupted or inconsistent with the obtained solution (i.e., outliers). RANSAC is governed by a random process, so results are obtained for several runs and averaged.

A typical result for the mean and standard deviation for G_{csN} is seen below

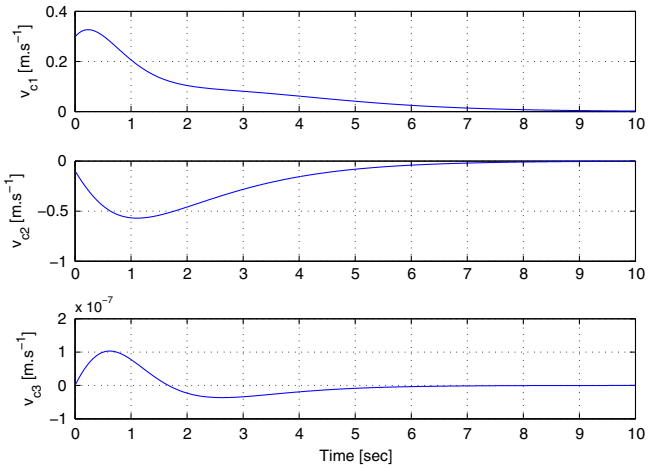
$$\mu(G_{csN}) = \begin{bmatrix} 0.6500 & -0.0296 & -83.2262 \\ 0.0072 & 0.6366 & -61.7221 \\ 0.0000 & -0.0001 & 1.0000 \end{bmatrix},$$

$$\sigma(G_{csN}) = \begin{bmatrix} 0.0016 & 0.0007 & 0.6298 \\ 0.0004 & 0.0013 & 0.3714 \\ 0.0000 & 0.0000 & 0.0000 \end{bmatrix}.$$

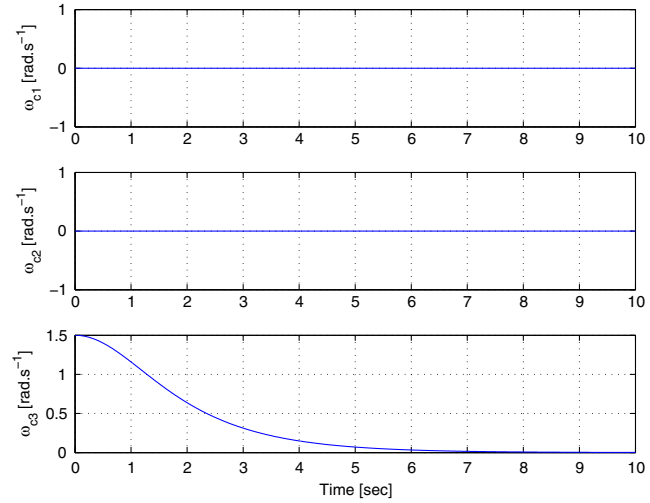
An experiment was carried out to demonstrate the need to solve the camera-to-screen homography. The virtual camera was placed at a known reference position in the VE, looking at a recognizable target (a dark window on a lighter building) and ten, known, reference positions with the same window visible. Images were captured at each pose, directly from the virtual environment display



(a)Feature point trajectories

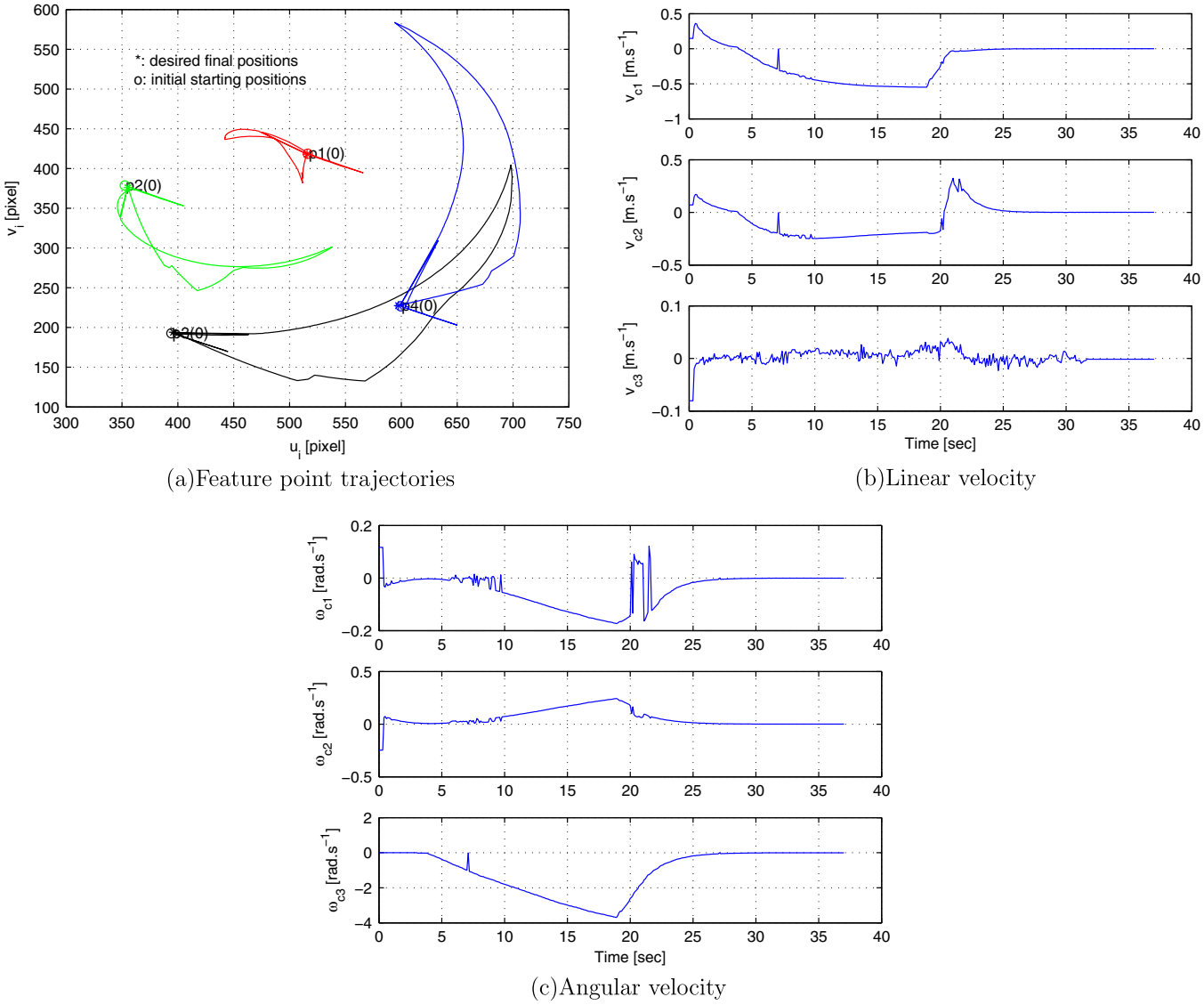


(b)Linear velocity



(c)Angular velocity

Fig. 10. Results of Simulink simulation of quaternion-based controller.



(a)Feature point trajectories

(b)Linear velocity

(c)Angular velocity

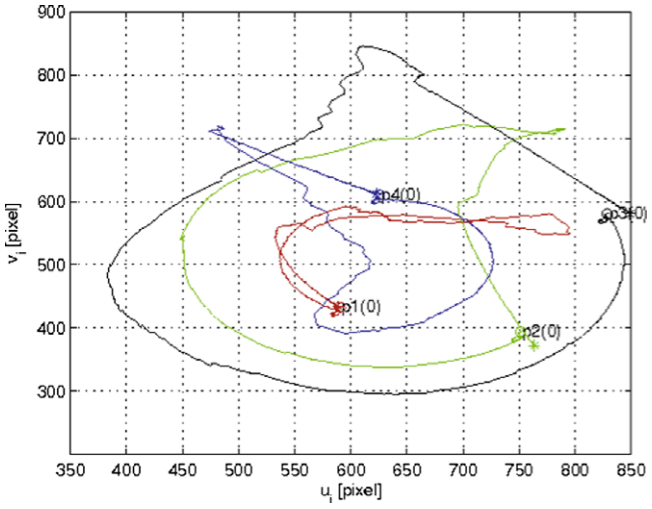
Fig. 11. Results of VE simulation of quaternion-based controller.

and the camera. A standard, vision-based motion estimation technique (see [21]) was used to estimate the displacement of the camera from the reference position to the test positions. Thus the estimation error for the vision-based estimation can be tested for pure simulations and camera-in-the-loop when G_{csN} is known and unknown. Given a known translation $T_s \in \mathbb{R}^3$ and an estimated translation $\hat{T}_s \in \mathbb{R}^3$, the translation estimation error is given by $\|T_s - \hat{T}_s\|$. Given a known rotation $R_s \in SO(3) \subset \mathbb{R}^{3\times3}$ and an estimated rotation $\hat{R}_s \in SO(3) \subset \mathbb{R}^{3\times3}$, the rotation estimation error is given by $\|I - R_s^{-1}\hat{R}_s\|$, where I is a 3×3 identity matrix and the matrix norm is given by the largest singular value of the matrix. Note that while the translation error has the appropriate unit of distance, the rotation error does not have an appropriate unit (in many cases, $\|I - R_s^{-1}\hat{R}_s\|$ is approximately the norm of the vector given by the roll, pitch, yaw angle parameterization of $R_s^{-1}\hat{R}_s$).

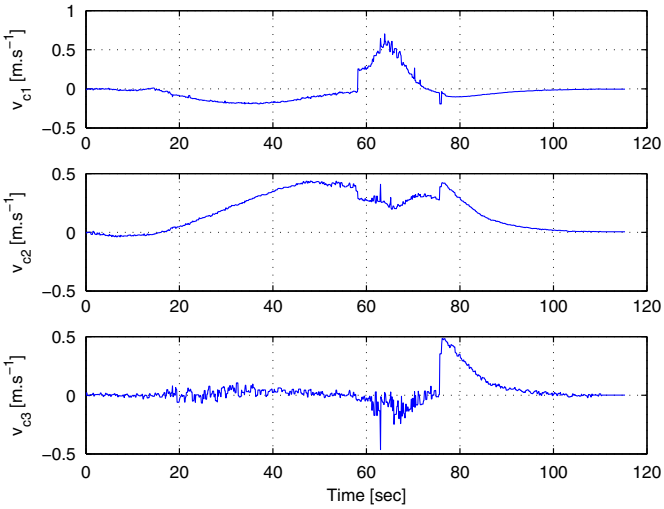
Results are given in Fig. 7. The first four poses were pure translation, while the next six were general motions involving translation and rotation. The mean of the errors is given as well. As expected, the estimations from pure simulation were better than camera-in-the-loop, reflecting the presence of noise and distortion unique to the camera. Knowledge of the camera-to-screen callibration improved the performance. In all but one case, the translation error was greater, often more than twice as large, when the camera-to-screen homography was not accounted for. Rotation error was always much larger when the calibration was not performed. Indeed, the rotation estimation error when G_{csN} is known is not generally much larger than the pure simulation case.

4. Stage three – UAV and avionics in the loop

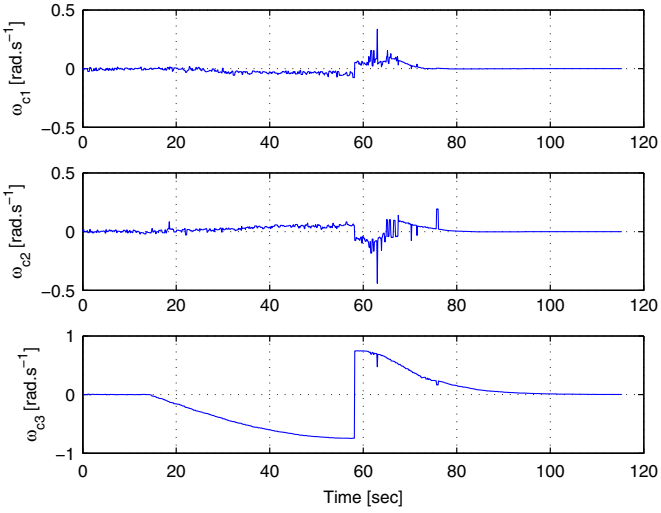
Active efforts focus on extending the amount of hardware in the control loop. Additional hardware includes flight hardware such as an actuated UAV, flight camera, transceivers, radio control, ground station PC, etc. This is illustrated in Fig. 8. The additional hardware allows for accurate testing of signal noise, signal dropout, lag, actuator saturation, etc. The UAV is mounted on a sting balance in a low-turbulence wind tunnel. Actuation of airfoils in the airflow of a wind tunnel will change the attitude of the UAV. The attitude change is relayed to the VE software, which alters the viewpoint accordingly. In turn, the vision-based control routines send velocity commands to the UAV autopilot. This is illustrated in Fig. 1.



(a)Feature point trajectories



(b)Linear velocity



(c)Angular velocity

Fig. 12. Results of camera-in-the-loop simulation of quaternion-based controller.

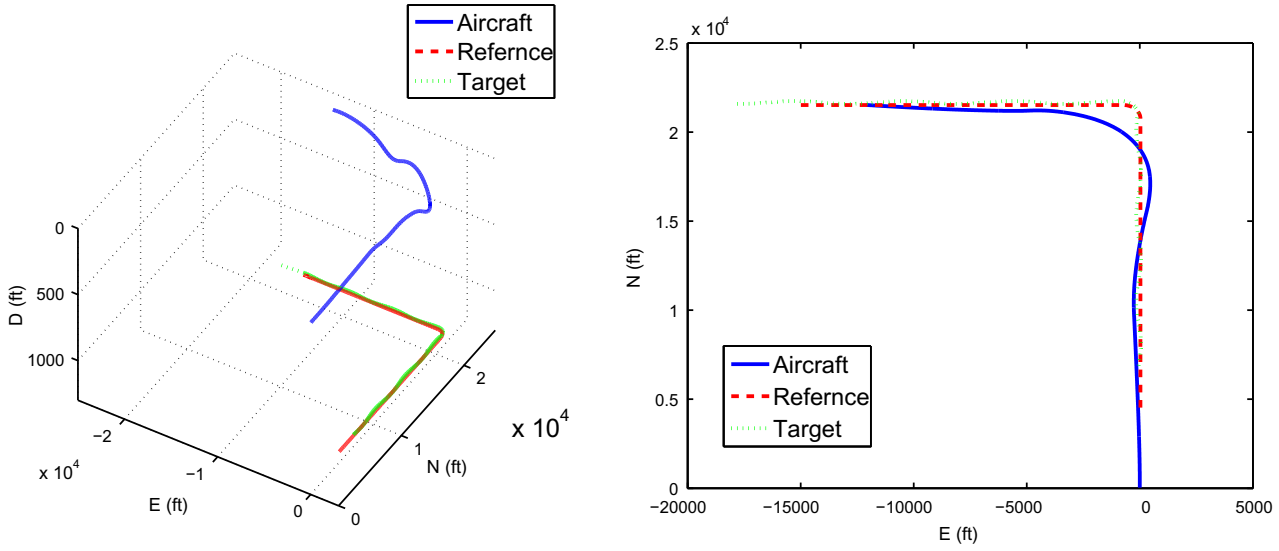


Fig. 13. Vehicle trajectories.

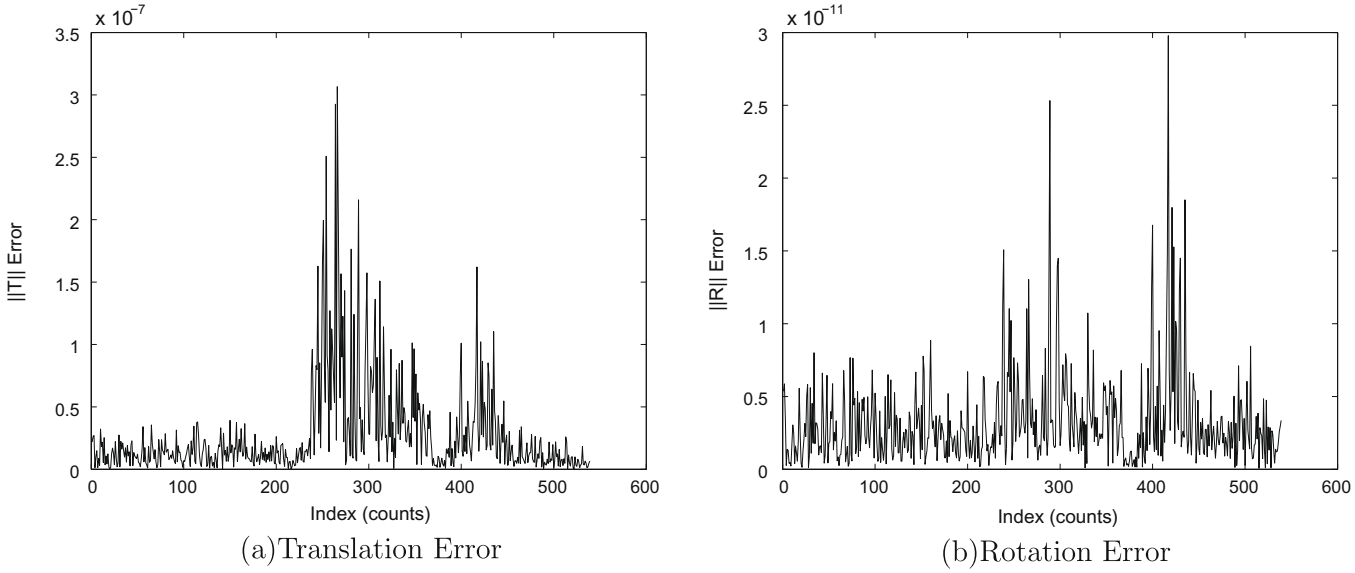


Fig. 14. Relative pose estimation error.

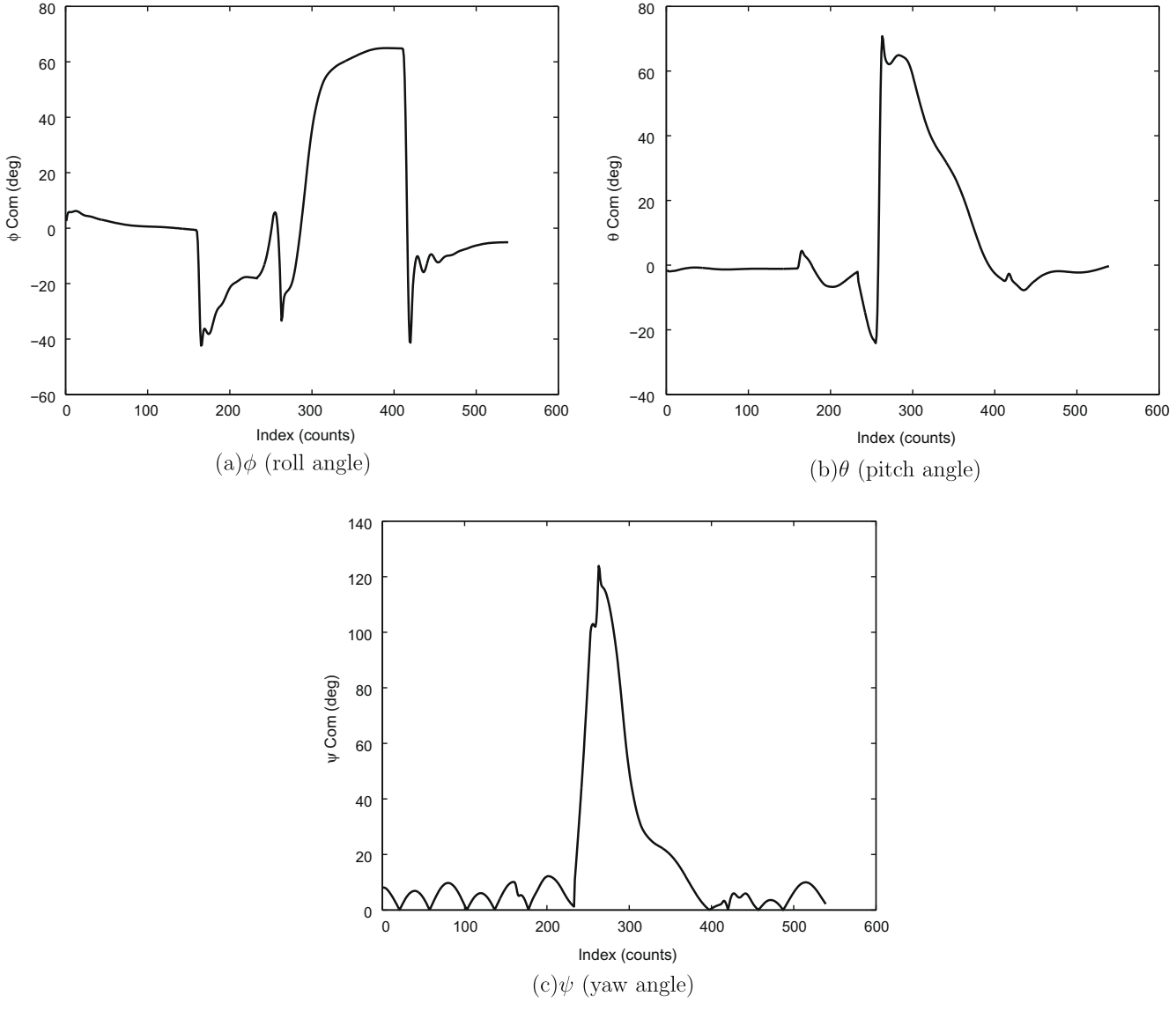


Fig. 15. Body axis angle commands.

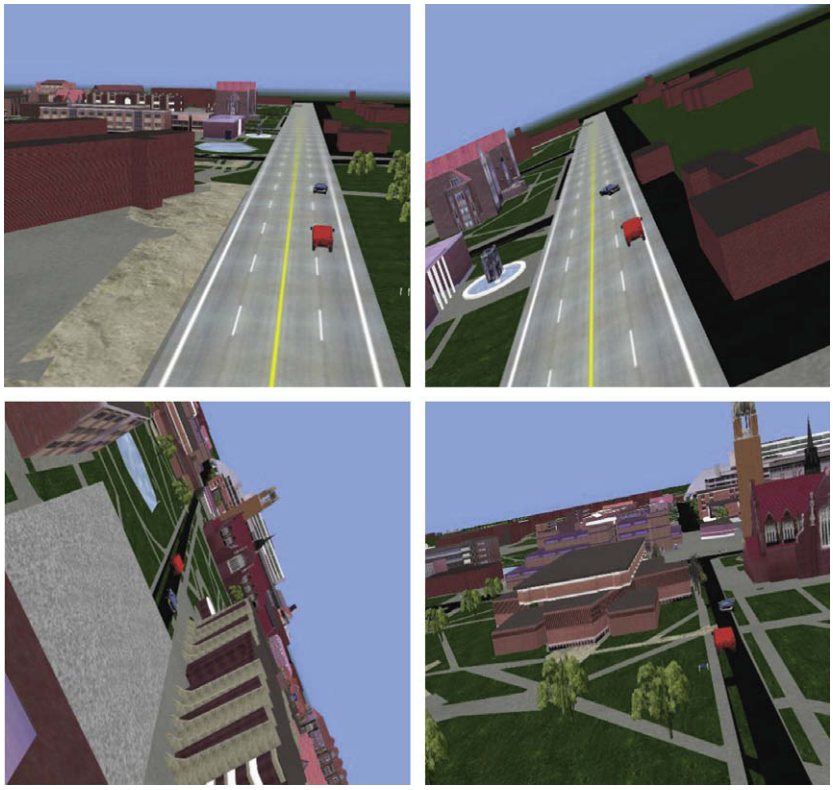


Fig. 16. Snapshots from vehicle pursuit experiment.

An E-flite Tensor 4D was chosen as the UAV for this stage of development and can be seen in a wind tunnel flight in Fig. 9. The Tensor 4D is a pull-prop biplane with a wingspan of 27 inches (68.5 cm) and length of 30 inches (76 cm). A Kestrel Autopilot by Procerus Technologies measures motion of the plane through an onboard Inertial Measurement Unit (IMU), rate-gyro, pressure sensor and electric compass. The autopilot can control the airfoil servos to regulate the attitude and velocity of the plane. A ground station runs Procerus Virtual Cockpit software to interface with the autopilot. The communication between the ground station and VE rendering cluster is accommodated via TCP/IP sockets. The wind tunnel facility is a low-speed low-turbulence facility with a test section of 107 by 300 cm. The free stream velocity can be finely adjusted in a range between 2 and 22 m/s. Typical testing Reynolds numbers, based on wind chord geometry, range between 50,000 to 150,000.

5. Example research

This section details the results of recently completed experiments in the HILS.

5.1. Visual servo control camera-in-the-loop experiment

Hu et al. [28] recently designed a novel visual servo pose controller that incorporates quaternions in the measurement of rotation error. Typical controllers map the rotation matrix to a vector in $\mathbb{R}^3$ (e.g. Euler angles, angle/axis, etc.). Any such mapping will have singularities which limit the region of convergence. The quaternion mapping has no such singularity. The quaternion controller was initially designed and simulated in Simulink. Results are shown in Fig. 10 for a task of rotating a camera by 180°. These figures show good system performance but do not give an accurate view of how a real system can be expected to perform. There is no signal noise and all points are perfectly tracked with subpixel accuracy. Furthermore, the simulation runs at 1 kHz, which is much faster than most cameras.

The control method in [28] was then implemented with the virtual reality environment without cameras, and the results are shown in Fig. 11. While there is no injected sensor noise, there is tracking error, quantization noise, no a priori knowledge of the feature point locations, and the camera rate is 30 Hz (the typical frame rate for a video camera). Finally, the experiment was repeated using a camera-in-the-loop, with results given in Fig. 12. As expected, the camera results show the effects of sensor noise, but the controller was still successful.

5.2. Pursuit of moving objects

A simulation to demonstrate navigation relative to moving objects is performed in the HILS. The setup consisted of three vehicles: an UAV with a mounted camera, a reference ground vehicle and a target vehicle. The camera setup considered in this problem consists of a single camera attached to the UAV with fixed position and orientation. While in flight, the camera measures and tracks feature points on both the target vehicle and the reference vehicle. This simulation assumes perfect camera calibration, feature point extraction, and tracking so that the state estimation algorithm can be verified.

The motion of the vehicles were generated to cover a vast range of situations. The UAV's motion was generated from a nonlinear simulation of flight dynamics using a high-fidelity model of an F-16 fighter with full 6 degrees-of-freedom [29]. Meanwhile, the reference vehicle and the target vehicle exhibited a standard car dynamic model with similar velocities. Sinusoidal disturbances were added to the target's position and heading to create some complexity in its motion. The three trajectories are plotted in Fig. 13 for illustration.

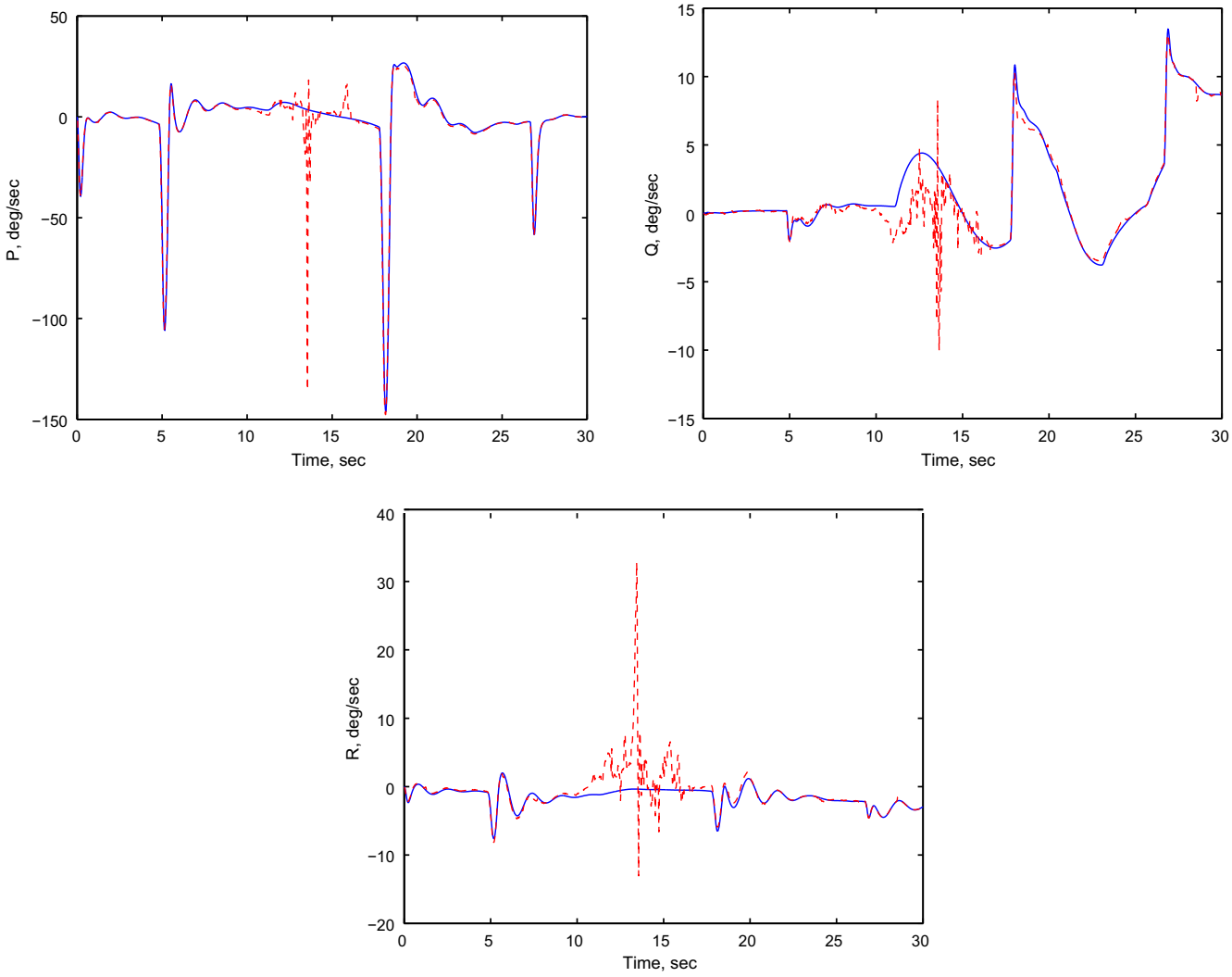


Fig. 17. True (—) and Estimated (- - -) values of rotational rates for roll (left), pitch (middle), and yaw (right).

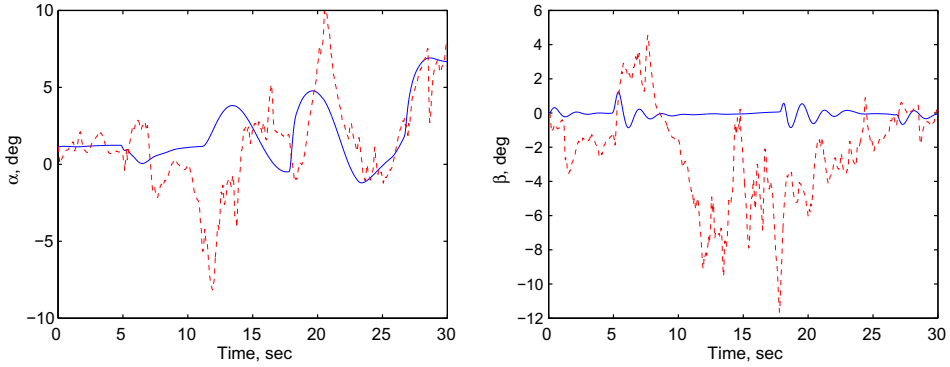


Fig. 18. True (—) and Estimated (- - -) values of angle of attack (left) and angle of sideslip (right).

Homography methods [21] were used in this simulation to find the relative rotation and translation between the ground vehicles. These results are then used to find the relative motion from the UAV to the target of interest. The error of this motion for translation and rotation are depicted in Fig. 14a and b. These results indicate that with synthetic images and perfect tracking of the target, accurate motion estimates can be extracted. The control signals used for tracking the target vehicle are shown in Fig. 15.

Fig. 16 is composed of snapshots from the camera view that depict the surrounding scene and the two vehicles, red designating the reference vehicle and gray for the target vehicle.

5.3. State estimation of an aircraft

State estimates are obtained using the HILS for a simulated flight. Specifically, a high-fidelity nonlinear model of an aircraft is

Fig. 19. Horizon extraction in the HILS. The camera view with horizon extracted can be seen in the lower left display.

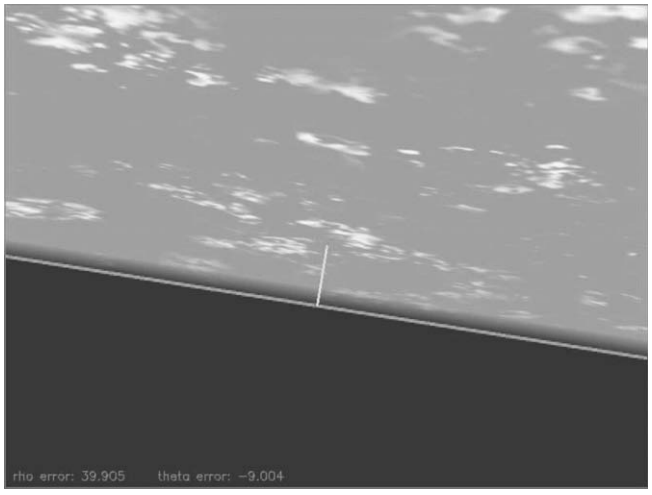


Fig. 20. Horizon extraction with no foreground objects.

simulated to fly through the VE of the University of Florida campus. The vehicle received a set of predetermined waypoints that determine a flight path that is tracked by an autopilot. In this way, the state estimates are not used for feedback but rather generated to validate the method.

The state estimates are based on optic flow. The actual measurements of optic flow, which can be simplified as the rate of change in the image plane of the feature points with time, are decomposed into rotational and translational components [30]. Such a decomposition is directly related to a focus of expansion such that all rotations are about this focus while all translations radiate from this focus. The movement of feature points relate to the focus of expansion are direct indicators of the states.

The rotational velocities of the vehicle are shown in Fig. 17 for both true values and estimated values. The translational velocities are also estimated, however, the scale-factor ambiguity precludes the absolute velocities from being computed. Instead, the relative velocities are computed in ratio. The angle of attack is the ratio of vertical velocity over forward velocity while the angle of sideslip is the ratio of side velocity over forward velocity. These angles are shown in Fig. 18.

The disparity in accuracy is a critical feature that was learned using the HILS. Initial noise-free feature-rich simulations using Matlab showed nearly perfect estimates throughout trajectories; however, the realistic imagery of the HILS created more problems for the feature point tracking and thus the state estimates. The rotation causes a much larger affect on the image than the translation, so this sensitivity was clearly highlighted using the high-resolution imagery. Another feature of interest is the lack of quality for any estimate during the middle portion of the flight. During this portion of the flight, the vehicle passed close to a sculpture composed of packed, yellow beams. The net effect was that the view was dominated by uniformly colored surfaces, and feature points could not be extracted. This caused difficulties in estimating any states. The HILS was instrumental in showing this common-sense occurrence that is difficult to demonstrate using simpler imagery, demonstrating the need for such a simulation system.

5.4. State estimation in the wind tunnel

Early testing of vision-based estimation and control using flight hardware in the loop focuses on vision-based attitude estimation.

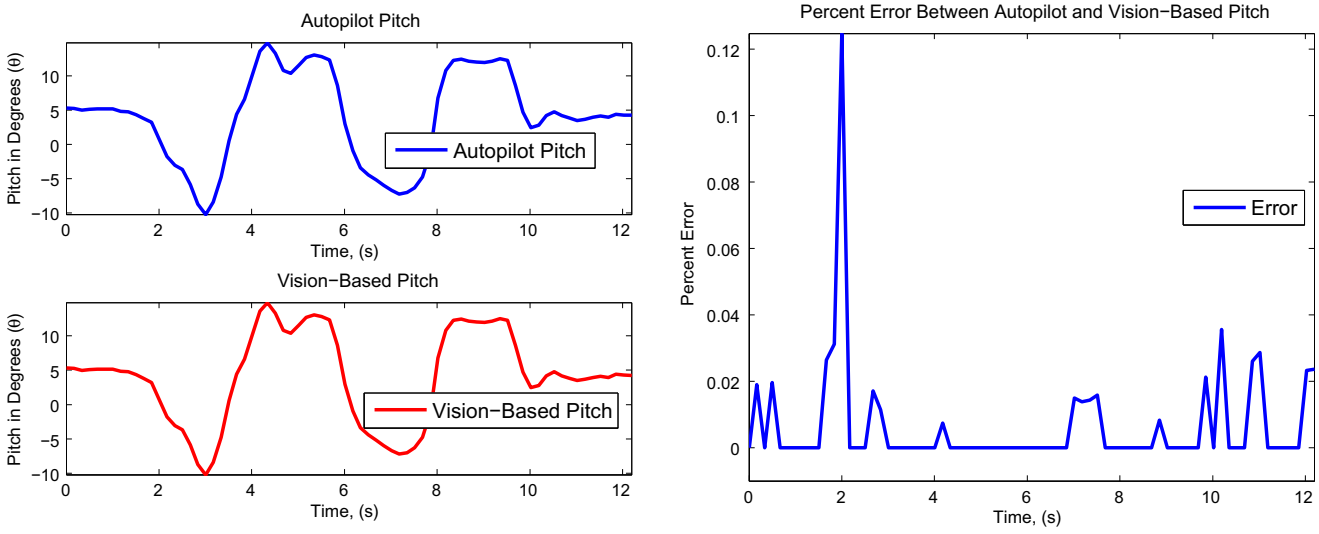


Fig. 21. Pitch angle measured by telemetry sensors and vision-based estimate, and percent error between the vision and telemetry.

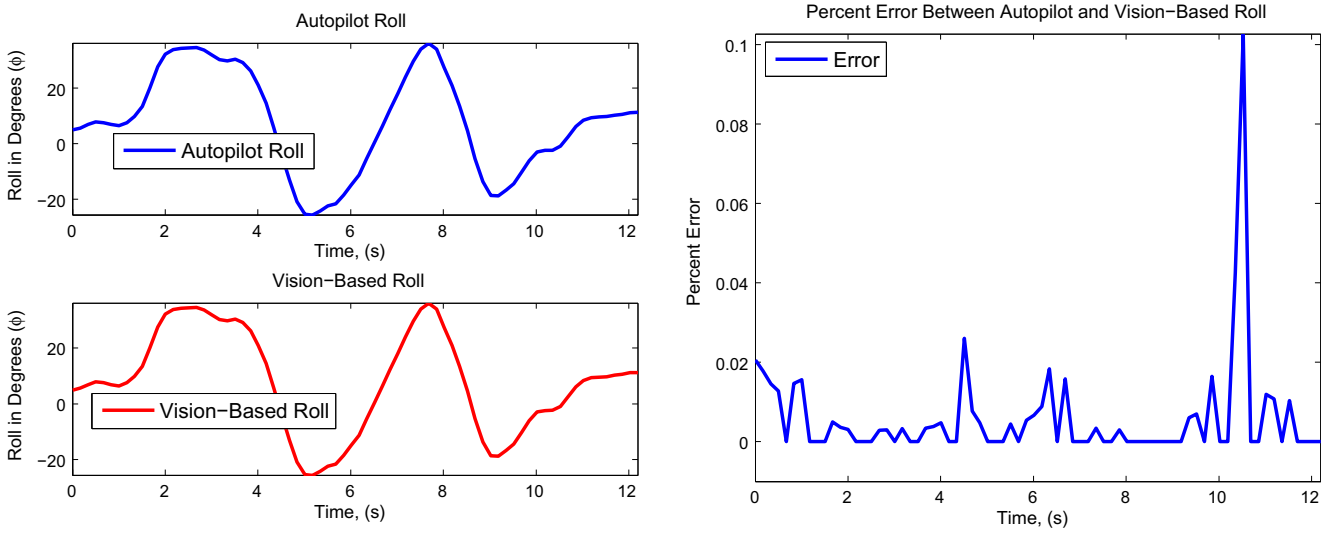


Fig. 22. Roll angle measured by telemetry sensors and vision-based estimate, and percent error between the vision and telemetry.

It has been established that horizon extraction can provide an accurate estimate of roll and pitch percentage (the ratio of image-space above the horizon to space below) [31,32] This provides an ideal test of the HILS. A vision program extracts the horizon line from the current view using color information and other queues. Many horizon extraction routines can ignore foreground clutter [31,33]. Fig. 19, shows the camera viewing a display of an urban scene. In the lower left, a monitor shows the camera view, along with the extracted horizon represented as a red line, which is able to ignore small foreground clutter. To ensure accurate testing, the simulation involved a high altitude without foreground clutter, as seen in Fig. 20, which shows a screen capture of the horizon extraction indicated in red[2]. The yellow line (orthogonal to the red line) represents the distance of the current horizon line from the set point of zero pitch, thus the length of the yellow line is proportional to the pitch angle. For simplicity, the camera reference frame was assumed to coincide with the aircraft reference frame, with the optical axis aligned with the major body axis of the aircraft.

The plane was remote piloted by a human in the wind tunnel, and performed a series of pitch, roll, and pitch/roll doublets. Doublets are a common testing procedure used in aerospace engineering, where the aircraft is made to quickly change the values being measured. In the case of the roll doublet, the UAV is made to roll right, left, right, left, and back to center with the target values of roll being 20° in either direction. Pitch doublets similarly alternated between pitch angles of approximately ±20° and pitch/roll doubles consisted of pitch and roll motions simultaneously.

While performing the maneuvers, roll and pitch angles were estimated by the autopilot telemetry as well as by the vision-based horizon extraction. Results comparing the telemetry data and vision estimation are shown in Figs. 21 and 22. The estimates matched extremely well for both pitch and roll, with error typically less that 0.1% of the total measurement. It is worth noting that generally only the pitch percentage can be estimated by horizon extraction, which is a function of the pitch, the altitude and the terrain. These experiments involved constant altitude maneuvers over flat terrain, so pitch percentage corresponds directly to pitch.

[2] For interpretation of color in Fig. 20, the reader is referred to the web version of this article.

6. Conclusion and future work

A novel platform has been developed for hardware in the loop simulation of vision-based control of UAV's. The need for such HILS platforms is strong, given the delicate and expensive nature of many UAV systems, the risk of damage to property during testing, and increasingly strict government regulations on flight testing. The HILS platform consists of virtual reality software to produce realistic images at a video rate of 30 Hz. Images can be processed directly in software, or projected on to a screen and viewed by a camera. The camera is the first level of hardware in the loop, and necessitated a novel solution for the camera-to-screen relationship, which can be modeled as a homography. Experiments demonstrate the efficacy of the HILS and highlight issues that may be encountered in flight but not seen in tradition simulations. Additional hardware is incorporated in the loop, including flight hardware consisting of an UAV with onboard autopilot interfaced to a ground station, which communicates with the virtual reality software. The UAV can be mounted in a wind tunnel, allowing attitude regulation through servoing the airfoils.

Acknowledgement

Many people contributed to the development of the HILS and conducting the experiments included in this paper. The authors would like to acknowledge the efforts of Dr. Chad Prazenica, Dr. Mujahid Abdulrahim, Dr. Guoqiang Hu, Dr. Roberto Albertani, Joe Kehoe, Adam Watkins, Ryan Causey, Siddhartha Mehta, Sumit Gupta, James Oliverios, Andy Quay, Yunjun Xu, Jayme Broadwater and Tullan Bennett.

This research was funded by AFOSR Grants F49620-03-1-0381 and FA9550-04-1-0262.

References

[1] Hanselmann H. Hardware-in-the-loop simulation as a standard approach for the development, customization, and production test of ECU's, Society of Automotive Engineers Paper Number 930207; 1993.
[2] Isermann R, Schaffnit J, Sinsel S. Hardware-in-the-loop simulation for the design and testing of engine-control systems. Control Eng Pract 1999;7:643–53.
[3] Duke E. V&V of flight and mission-critical software. Software IEEE 1989;6(3):39–45.
[4] Johnson E, Fontaine S. Use of flight simulation to complement flight testing of low-cost UAV's. In: Proc AIAA Conf Model Simul Technol; 2001.

[5] Williamson WR, Abdel-Hafez MF, Rhee I, Song E, Wolfe JD, Chichka DF, et al. An instrumentation system applied to formation flight. IEEE Trans Contr Syst Technol 2007;15(1):5–85.
[6] Liu Y, Steurer M, Ribeiro P. A novel approach to power quality assessment: real time hardware-in-the-loop test bed. IEEE Trans Power Delivery 2005;20(2):1200–1.
[7] Stoeppler G, Menzel T, Douglas S. Hardware-in-the-loop simulation of machine tools and manufacturing systems. Comput Control Eng J 2005;16(1):10–5.
[8] de Carufel J, Martin E, Piedboeuf J-C. Control strategies for hardware-in-the-loop simulation of flexible space robots. IEE Proc Control Theory Appl 2000;147(6):569–79.
[9] Martin A, Emami M. An architecture for robotic hardware-in-the-loop simulation. In: Proc IEEE Int Conf Mechatronics Automat; 2006. p. 2162–7.
[10] Jodeh N, Blue P, Waldron A. Development of small unmanned aerial vehicle research platform: modeling and simulating with flight test validation. In: Proc AIAA Conf Modeling Simulat Technol; 2006.
[11] Bacic M, Daniel R. Towards a low-cost hardware-in-the-loop simulator for free flight simulation of UAVs. In: Proc AIAA Conf Modeling Simul Technol; 2005.
[12] MacDiarmid M, Bacic M, Daniel R. Extension and application of a novel hardware-in-the-loop simulator design methodology. In: Proc IEEE Conf Decision Control; 2008. p. 5054–61.
[13] Bacic M. On prediction of aircraft trajectory at high angles of attack: preliminary results for a pitching aerofoil. In: Proc AIAA Model Simul Technol Conf; 2008.
[14] Frew E, McGee T, Kim Z, Xiao X, Jackson S, Morimoto M. et al. Vision-based road-following using a small autonomous aircraft. In: Proc IEEE Int Conf Robotics Automat, vol. 5, 2004. p. 3006–15.
[15] McGee T, Sengupta R, Hedrick K. Obstacle detection for small autonomous aircraft using sky segmentation. In: Proc IEEE Int Conf Robot Automat; 2005. p. 4679–84.
[16] Narli V, Oh P. Hardware-in-the-loop test rig for designing near-earth aerial robotics. In: Proc IEEE Int Conf Robot Automat; 2006. p. 2509–14.
[17] Galassi M, Davies J, Theiler J, Gough B, Jungman G, Booth M, et al. GNU scientific library: reference manual. Bristol, UK: Network Theory Ltd.; 2005.
[18] Courtois PJ, Heymans F, Parnas DL. Concurrent control with 'readers' and 'writers'. Commun ACM 1971;14(10):667–8.
[19] Longuet-Higgins H. A computer algorithm for reconstructing a scene from two projections. Nature 1981;293:133–5.
[20] Huang T, Faugeras O. Some properties of the E matrix in two-view motion estimation. IEEE Trans Pattern Anal Machine Intell 1989;11(12):1310–2.
[21] Faugeras OD, Lustman F. Motion and structure from motion in a piecewise planar environment. Int J Pattern Recog Artificial Intell 1988;2(3):485–508.
[22] Broida T, Chellappa R. Estimating the kinematics and structure a rigid object from a sequence monocular images. IEEE Trans Pattern Anal Machine Intell 1991;13(6):497–513.
[23] Soatto S, Frezza R, Perona P. Motion estimation via dynamic vision. IEEE Trans Automat Contr 1996;41(3):393–413.
[24] Chiuso A, Favaro P, Jin H, Soatto S. Structure from motion causally integrated over time. IEEE Trans Pattern Anal Machine Intell 2002;24(4):523–35.
[25] Dixon WE, Fang Y, Dawson DM, Flynn TJ. Range identification for perspective vision systems. IEEE Trans Automat Contr 2003;48(12):2232–8.
[26] Ma L, Chen Y, Moore KL. Range identification for perspective dynamic system with single homogeneous observation. In: Proc IEEE Int Conf Robot. Automat; 2004. p. 5207–12.
[27] Fischler M, Bolles R. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: Commun. ACM, 1981. p. 381–95.
[28] Hu G, Dixon WE, Gupta S, Fitz-coy N. A quaternion formulation for homography-based visual servo control. In: Proc IEEE Int Conf Robot Automat; 2006. p. 2391–6.
[29] Stevens B, Lewis F. Aircraft control and simulation. Hoboken, NJ: John Wiley and Sons; 2003.
[30] Kehoe J, Watkins A, Causey R, Lind R. State estimation using optical flow from parallax-weighted feature tracking. In: Proc AIAA Guidance Navigat Control Conf; 2006.
[31] Kurdila A, Nechyba M, Prazenica R, Dahmen W, Binev P, DeVore R, et al. Vision-based control of micro air vehicles: progress and problems in estimation. In: Proc IEEE Conf Decision Control; 2004. p. 1635–42.
[32] Kehoe J, Causey R, Lind R, Kurdila A. Maneuvering and tracking for a micro air vehicle using vision-based feedback. SAE Trans 2004;113(1):1694–703.
[33] Bao G, Xiong S, Zhou Z. Vision-based horizon extraction for micro air vehicle flight control. IEEE Trans Instrum Measure 2005;54:1067–72.