SLUGS UAV: A Flexible and Versatile Hardware/Software Platform for Guidance Navigation and Control Research

Mariano Lizarraga[1],Gabriel Hugh Elkaim[2], and Renwick Curry[3]

***Abstract*— This paper presents the Santa Cruz Low-Cost UAV GNC Subsystem (SLUGS) developed at the University of California Santa Cruz. It is a versatile and flexible autopilot capable of controlling a small unmanned system. It is tightly integrated with MATLAB/Simulink, and allows for a simple and easy transition from pure simulation to HIL simulation to flight code. The hardware's main processing units are two low-cost dsPIC33F DSCs, one handling sensor input and the other managing the navigation and control loops. The *sensor DSC* implements a complementary attitude and position filter. An interprocessor communication protocol delivers fused attitude and position estimates to the *control DSC*. The *control* DSC implements low-level platform stabilization using PID loops, and higher level waypoint following based on a line-of-sight guidance law. Data is logged and available for replay post flight on both the ground station software, and also within MATLAB/Simulink. The hardware was installed on a low-cost single engine electric RC aircraft, and has been demonstrated to be capable of sustained autonomous flight. Several estimation topologies have been tested and developed using the system. The SLUGS is general, and can be adapted to multiple autonomous platforms such as helicopters, quadrotors, twin engine aircraft, ground, and marine surface vehicles.**

I. Introduction

Unmanned Aerial Vehicle (UAV) usage has exploded in the recent years. Fueled by multiple factors, but mainly due to the commoditization of the onboard components, UAVs are becoming ubiquitous. Examples of UAV uses, besides the typical law enforcement and military applications, are forest fire monitoring, data gathering for marine uses, search and rescue missions in disaster areas, shipboard oil pollution monitoring and detection, agricultural multi-spectral imaging, soil erosion monitoring, and even as off-the-shelf high-end toys.

Motivated by these, and many other applications, researchers have become interested in these platforms. Currently UAVs are being actively used to conduct research in fault-tolerance[1], adaptive control [2], trajectory tracking[3], path following[4], and cooperative control[5].

These factors have allowed many universities to establish well-equipped UAV laboratories e.g., [6][7] to conduct research in diverse topics in image processing and controls.

Sec. II presents the motivations that lead the UC Santa Cruz' Autonomous Systems Laboratory to embark in the development of a UAV autopilot that facilitated R&D. Sec. III describes the workflow when using SLUGS as a research platform. Sec. IV presents, at a block diagram level, the hardware architecture in the SLUGS autopilot. Sec. VI presents the Hardware-in-the-loop simulator architecture and discusses its advantages over other existing solutions. The paper concludes by showing flight test results, in Sec. VIII, for navigating a waypoint array and orbiting over a point of interest, and concluding remarks are in Sec. IX.

Department of Computer Engineering, University of California Santa Cruz, Santa Cruz, CA 95064, USA
[1]malife@soe.ucsc.edu, [2]elkaim@soe.ucsc.edu, [3]rcurry@ucsc.edu

II. Motivation for an R&D Autopilot

There exists a wide variety of commercial miniature autopilots; Piccolo[8], MicroPilot[9] and Kestrel[10] are the most visible in the US market. Instrumented with a set of sensors all of them deliver a complete, ready-to-use–but very difficult to modify–package. There are also many Open Source options: the Paparazzi autopilot[11] uses a set of thermopyles (infrared sensors) for attitude determination, while DiyDrones' ArduPilot[12] and the GentleNav/UDB[13] implement a traditional AHRS.

While the systems available today (commercial and open-source) are very capable and provide a useful platform for UAV flights, they are all difficult to modify. This is the driving force behind the autopilot design discussed in this paper. Research and Development (R&D) labs seeking to advance the state of the art require an autopilot that is easy to modify to suit their research objectives. Currently, these labs face two main options: **(i)** Augment the fixed set of autopilot commands available to the end user with their proposed control system[14], or **(ii)** invest a considerable amount of time and resources in modifying the autopilot's source code, hardware, or both (if able) to suit their needs.

To overcome these limitations this work presents an autopilot that has been designed from the beginning to be easily reprogrammable using MATLAB/Simulink code generation tools. By harnessing the power of automatic code generation, anything from a simple change in controller architecture to a complete redesign can be achieved without the need of directly writing source code. The embedded target blocks to access the hardware peripherals for the DSCs are from Ref. [15].

III. The SLUGS Workflow

The fundamental advantage of the SLUGS autopilot is its tight integration with MATLAB/Simulink. The overall process starts with software simulation, where the guidance, navigation, and control (GNC) algorithms interact with a full 6DOF flight model of the aircraft. The next step is Hardware-in-the-Loop (HIL) simulation (discussed in Sec. VI) where the algorithms are now running on the autopilot hardware in real time. The final step is is actual flight where the autopilot

now uses its onboard sensors to control the aircraft. This process is represented in Fig. 1 which shows the differences in simulation, HIL simulation and flight stages.

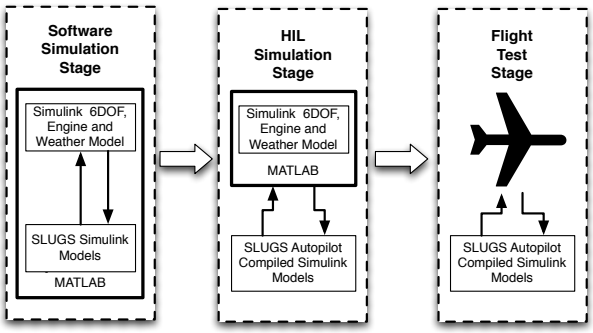


Fig. 1: The SLUGS Workflow

This workflow has several key advantages: (i) most GNC researchers are already familiar with the MATLAB/Simulink environment; (ii) flight code is produced without rewriting or adapting the algorithms (a source of many bugs); (iii) flight telemetry data can be replayed post-flight through the MATLAB models of the attitude and control system to debug the algorithms and fine tune results; and (iv) HIL simulation which matches the flight code exactly.

A. Software Simulation Stage

The software simulation is developed in Simulink, and drives a 6DOF model of our aircraft. Here, truth is always known, and full control over the environment can be exerted. Algorithms are developed and tested, architecture is modified, and different computational methods are benchmarked against each other. With full flexibility, perfect attitude can be assumed, or the sensor data can be corrupted and fed to the attitude estimation as required for system development. By using the same Simulink model for flight there also exists the option to bring in raw flight data from a telemetry file previously recorded so that algorithms can be checked against real flight data.

B. Hardware-in-the-loop Stage

Once the algorithms are developed, the blocks can be targeted to the embedded processors (control and sensor) using the dsPIC blockset[15] and MATLAB's embedded coder. While some low-level C code is required (for such things as circular buffers and GPS parsing), these too appear as blocks within the Simulink. The Embedded Target calls the C compiler and creates the code that is loaded onto each DSC. For HIL testing, the SLUGS controls a 6DOF simulation running on a PC while the flight code is executing on the embedded DSCs in real time. (In essence, the SLUGS does not know that it is not flying a real aircraft.) HIL testing allows testing of flight hardware (e.g., communications protocols and computational load on the DSCs);

C. Fight Test Stage

With HIL testing complete, the SLUGS is physically mounted in the UAV, and flight tests are conducted. Here, differing levels of control can be passed from the safety pilot to the autopilot (detailed in Sec. V). Flight telemetry is monitored from the ground station and recorded for post-flight analysis. This telemetry data can then be used as inputs to the MATLAB/Simulink simulation of attitude and navigation modules to verify and validate the model. Based on flight performance, improvements are proposed and the design cycle is iterated.

While the flight code generated by the Embedded Target for the SLUGS is clearly not as efficient as hand coded C, it is, however, bug free[1]. This is an enormous advantage as a great deal of time in flight testing is spent looking for bugs in the flight code. This is especially true with flight vehicles where bugs can cause crashes that destroy the vehicle.

IV. Autopilot Hardware

The development of the SLUGS autopilot was a direct result of the desire to have an autopilot that was easy to reprogram using MATLAB/Simulink for UAV research. The goal was to develop a complete architecture for an autopilot for small UAVs that had enough processing power for moderately complicated control tasks and at the same time was easily and rapidly reprogrammable using the advanced capabilities of Simulink. The SLUGS architecture physically decouples sensor integration and attitude estimation/navigation from the control algorithm and communications by using two separate dsPIC33 DSCs interconnected via a high-speed Serial Peripheral Interface (SPI) bus using the MAVlink protocol[16]. The autopilot has been designed to be modular and extendable in order to accommodate a rich sensor and peripheral suite as the need arises.

The physical hardware of the SLUGS is pictured in Fig. 2, and the engineering block diagram in Fig. 3. The basic architecture uses two separate DSCs, one to handle sensor inputs and vehicle state estimation, and the other to implement guidance, navigation, and control tasks. This decoupling allows researchers to explicitly work on each broad area separately without the danger of corrupting the other functionality. Details of the design and implementation can be found in [17], however a brief overview is provided here for completeness.

A. Sensor DSC

The *sensor* DSC handles the incoming data from the various sensors (gyros, accelerometers, magnetometers, GPS, etc.) as well as monitoring the incoming RC commands from the RC receiver. It implements a complementary filter (CF) for attitude estimation based on a quaternion integration of the rate gyros, and a feedback from the accelerometers and GPS course over ground to determine gyro biases. This attitude filter has been benchmarked against a traditional 15-state Extended Kalman Filter (*EKF*) with comparable results,

[1]Bug free here means that the code faithfully replicates the Simulink blocks. It is of course possible to create an erroneous Simulink model.

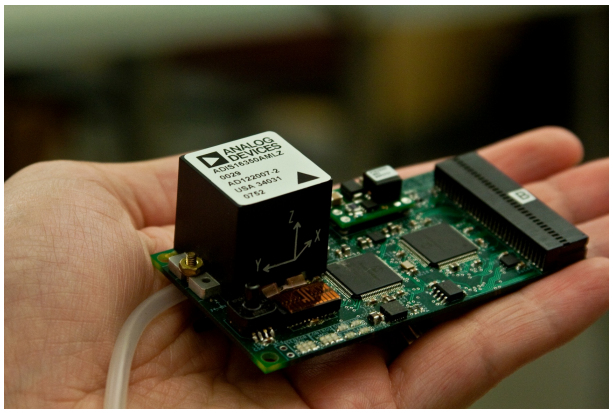


Fig. 2: The UCSC SLUGS autopilot

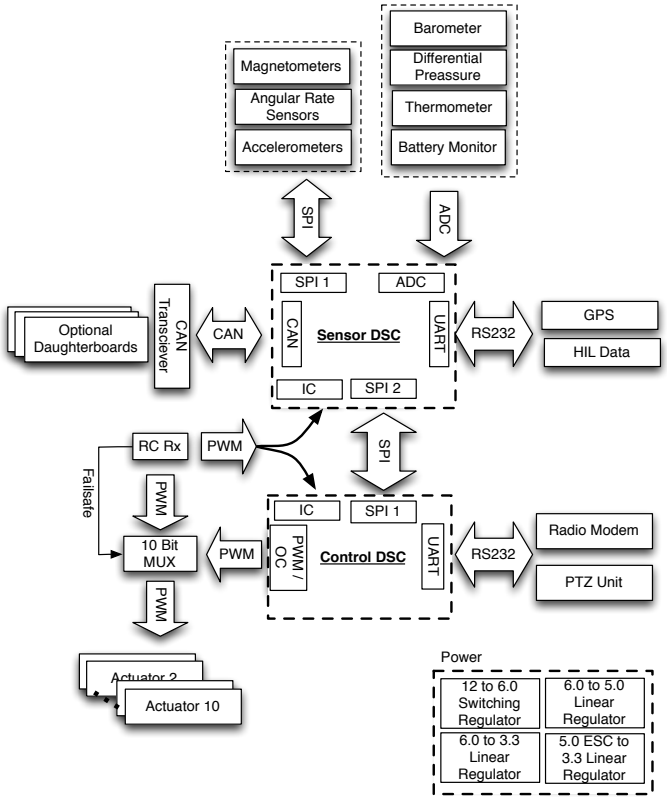


Fig. 3: Block Diagram of the SLUGS Autopilot

at a much lower computational burden[18]. As anecdotal evidence of the advantage of the SLUGS workflow, the attitude estimation filter was originally designed to use the magnetometers for feedback, but was unusable due to excessive noise from the electric motor. The filter configuration was redesigned, simulated, tested in HIL, and then flight tested all within 24 hours.

The position filter uses GPS position and velocity, with integration of the accelerometers for high bandwidth updates. This was also implemented as a complementary filter, and the position will diverge quickly upon loss of GPS. A more robust dead reckoning solution based on the airspeed will be implemented if longer GPS outages occur. The altitude (Z-position) is also derived from a complementary filter that fuses high rate barometer readings with GPS. There is switch logic in the altitude CF to ignore GPS vertical jumps (usually due to satellites coming in and out of the solution) that are not physically possible for the aircraft.

In the case of Hardware-in-the-Loop (HIL) testing, sensor data from the simulation enters the sensor DSC directly through its second serial port from the computer that runs the 6DOF simulation (see Sec. VI).

B. Control DSC

The *control* DSC implements the low-level inner loop control and stabilization, as well as the higher level guidance. It is also the main processor which delivers telemetry to, and receives commands from, the ground control station through an RF link. The SLUGS autopilot implements a PID based inner loop controls that individually control the lateral directional modes and the longitudinal modes of the aircraft. The lateral control consists of two loops: a commanded roll to aileron loop, and a lateral acceleration to rudder loop (which coordinates turns). Logic is provided to translate turn rate commands to roll commands in an open loop fashion. The longitudinal control consists of an airspeed controlled by a PID loop commanding the throttle, and a cascaded altitude to pitch and pitch to elevator loops for altitude control. The first part of the altitude control loop is an altitude command to pitch PI loop with hard limits on acceptable pitch ($\pm 15^\circ$). The output of the PI loop is commanded pitch which is used as the input of the PID loop to output elevator commands.

Given that a banking aircraft will lose altitude, there is a feedforward term from roll to elevator command to keep the aircraft from descending when banking into a turn. Every pilot learns to pull back on the stick when turning, and the feedforward gain simply implements a mathematically correct logic. Details of the inner loop design can be found in [17].

The higher level waypoint following is based on a improved line of sight guidance law (L_2^+) that extends previous work to account for roll dynamics and the effects of groundspeed on the navigation. Switch logic handles transitions from legs and line acquisition when far away from the waypoint. A *return to base* functionality is included as a safety feature, and the L_2^+ controller is capable of following a target or moving base station with no additional changes. A complete treatment of the L_2^+ controller can be found in [19].

C. Failsafe

The SLUGS contains a robust failsafe system to return control to the human safety pilot in case of any failures. The system monitors the signal coming out of the RC receiver onboard the airplane[2], and uses this signal to pass the RC signals directly to the control surfaces or allows SLUGS to control them. The failsafe is completely self contained; it does not share power or other components with the SLUGS board, but rather runs off of the critical systems power bus

[2]Used by the safety pilot for takeoff and landing.

which powers the main motor and the RC servos. A small microcontroller monitors the failsafe signal and implements a state machine to hand over (or take control) during the dead time of the RC pulses.

V. Ground Station Software

The groundstation hardware and the groundstation software (GSS) are a necessary parts of every unmanned system to allow the users need to monitor and command the vehicle. In the case of SLUGS, we have a modified version of the popular open source QGroundControl software[20] to suit our needs. The software interprets the MAVLink messages from the RF communications link and allows the operator to control the aircraft in autonomous mode, as well as monitoring the telemetry in real time. Note that this is fundamentally different from the safety pilot who simply controls the aircraft and can turn control over to the SLUGS autopilot through the "gear" switch on the RC transmitter.

From the ground station software (GSS), the operator is able to verify control parameters and change any control gains on the fly. Waypoints can be added and deleted from the mission and uploaded to the aircraft, and return to base or point of interest orbiting can be initiated. Return to base is a special function that automatically executes if the SLUGS loses contact with the ground station for more than 30 seconds; it will interrupt its current mission and return to the home position (also entered from the ground station preflight).

There are three main modes to control the UAV: (i) manual; (ii) mid-level commands; and (iii) autonomous. *Manual* control simply responds to the traditional RC inputs from the safety pilot, while logging all sensor and control inputs. *Mid-Level* control commands the basic inner loop with airspeed, turn rate, and altitude commands. *Autonomous* control implements fully autonomous flight which can be switched between mission waypoints, point of interest orbiting, and return to base. Note that SLUGS also implements a *selective passthrough* mode which allows the autopilot to command only certain servos while leaving others to the safety pilot (e.g., throttle controlled by the AP, all others actuators controlled by the pilot). This is very useful when tuning the gains of the inner loop, as the safety pilot can keep the aircraft within a safe zone while changing the gains.

Telemetry can be displayed in real time, and graphs of any of the incoming variables are available on the engineering display of the GSS. Telemetry is logged to a file which can be replayed through the ground station or exported for Simulink replay and/or MATLAB post-flight analysis.

VI. Hardware-in-the-Loop Simulation

An important resource for any autopilot research is a Hardware-In-the-Loop (HIL) simulator. This software allows the end user to conduct overall system check, training, and, in the R&D case, algorithm verification and validation without the need to fly. It allows the autopilot to fly a mathematical simulation of the aircraft. This is extremely important for verification and validation of the flight hardware, and allows for aggressive experiments (crashes simply require a reset, rather than rebuilding the UAV). While most autopilots come with HIL capability, most do not have the flexibility to completely change the flight model and environment. By using a 6DOF Simulink model to drive the simulation, *any* aircraft or environment or failure can be simulated.

The HIL architecture in Fig. 4 consists of a PC running the 6DOF simulation in Simulink, connected via a serial port directly to the *sensor* DSC. The Simulink 6DOF model (and its engine and weather model) communicates in UDP packets which are translated by an intermediate program to serial. A second computer runs the GSS, which can be connected either via a direct serial link or via an RF link.

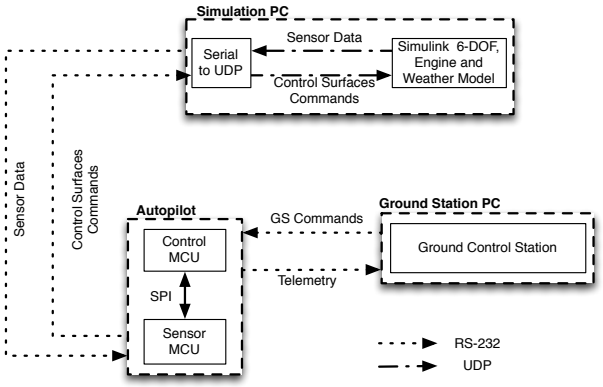


Fig. 4: Hardware-in-the-loop Architecture

When set into HIL mode from the GSS, the *sensor* DSC overwrites its sensor data from the incoming data on its secondary serial port. Likewise, the control surfaces commands are packaged and sent back to the 6DOF model. The simulation PC receives this commands and flies the simulated aircraft. The roundtrip delay is approximately $30msec$ and is well within the capability of the control system to handle. Failures can be introduced, and truth is available from the 6DOF simulation to check the algorithms for correctness (something not available in actual flight data).

The vast bulk of the development cycle occurs between the pure simulation and the HIL, with actual flights done to verify and validate the algorithms. HIL is a low-risk method of exercising the autopilot and testing it against harsh failures and difficult scenarios.

VII. Aircraft Integration

The SLUGS was installed in an off-the-shelf commercial hobby electric model aircraft, the Multiplex Mentor shown in Fig. 5. It is a very benign aircraft with an endurance of approximately 15 minutes on its battery. The autopilot was installed in a small balsa wood "cage" aft of the main wing, along with its GPS receiver and RF modem. This location was determined to be the most crash resistant on the aircraft.

Including the SLUGS and all flight hardware, the Mentor has an all up weight of $\sim 2.1Kg$. The aircraft has a speed range of approximately $10 - 20m/s$ and is always flown within sight of the safety pilot. Early flight experiments

Fig. 5: UCSC's UAV based off a Multiplex Mentor.

were for tuning the individual low-level control PID gains, followed by waypoint tracking, circular navigation and return to base.

VIII. Flight Tests

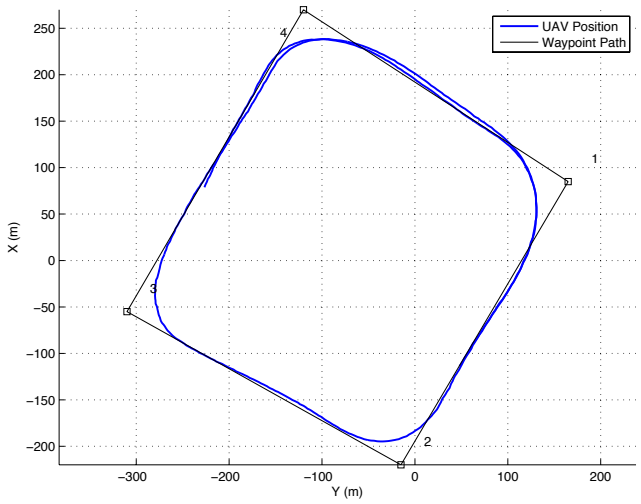
As a research platform, the SLUGS/Mentor combination provides a low-cost method of validating various control strategies and verifying that they work in practice as well as in theory. Flight tests were conducted to validate both inner loop and waypoint guidance control[3]. The SLUGS UAV was flown at UCSC in order to validate the complete system. On all flights, takeoff and landing was performed by the safety pilot who would switch the SLUGS into autonomous mode to begin the mission. To test waypoint navigation, a waypoint array of four approximately equal legs describing a rough square was used. Fig. 6a shows SLUGS performance after flying the waypoint array. The next test consisted on orbiting around a point of interest. Fig. 6b shows the performance when orbiting such point.

As a demonstration of capability, an adaptive control architecture (based on Hovakimyan [21]) was implemented on SLUGS along with a sideslip compensator (SSC)[17] and tested under various control surface failures. The flight was conducted with an in-flight hard-over rudder failure of 8° deflection, and the resulting trajectories are shown in Fig. 7a (though the aircraft was flying with a 19° sideslip angle to maintain its trajectory). With the full adaptive control running, the control DSC had a nominal load of less than 50% as shown on Fig. 7b. Again, a complete description of the adaptive control experiments can be found in Ref. [17].

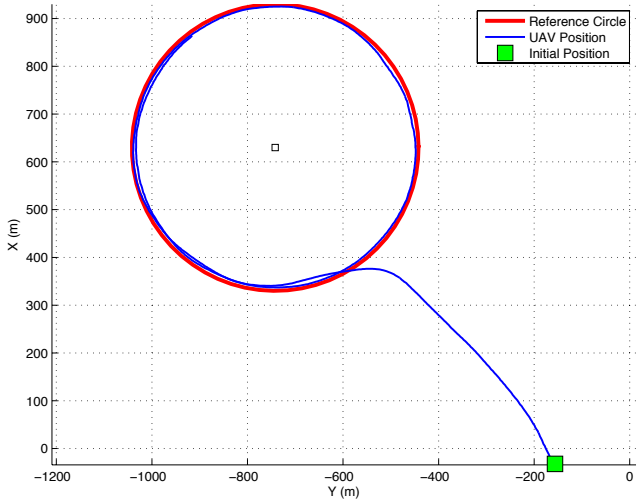
IX. Conclusions

This paper has presented an overview of the SLUGS autopilot, developed at UC Santa Cruz' Autonomous Systems Lab. A low-cost yet capable autopilot especially suited for research into Guidance, Navigation, and Control (GNC) algorithms. While other commercial and open source autopilots exist, none offer the versatility, performance, or ease of change that SLUGS provides. The SLUGS is being

[3]Detailed results of the initial validation and performance metrics of these flights are presented in Ref. [17].



(a)

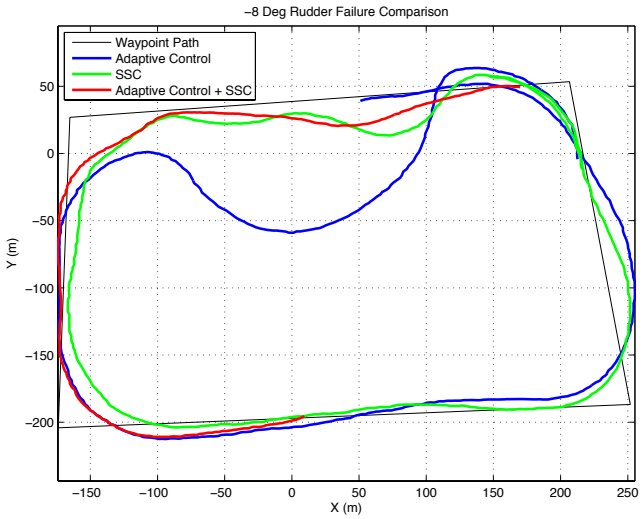


(b)

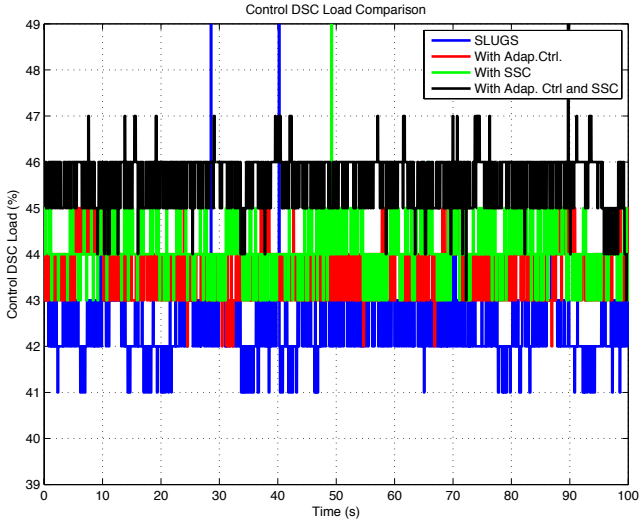
Fig. 6: Flight data for the L_2^+ guidance law[19] on the Mentor platform, showing waypoint array tracking performance in (a) and orbiting around a point of interest in (b)

continually improved, with better attitude estimation algorithms currently under development, as well as migrating the SLUGS to other platforms. Currently, a SLUGS-based autonomous marine surface vessel is being developed, and a SLUGS-based unmanned ground vehicle has already been deployed.

Due to its tight integration into MATLAB/Simulink, it provides a seamless transition from software simulation to Hardware-in-the-Loop (HIL) simulation to flight code. By dividing up the sensor fusion and estimation tasks onto one DSC and the control and navigation tasks on to the other, the SLUGS retains computational horsepower sufficient to fly modern algorithms on small aircraft. Flight tests have demonstrated good performance in its base form flying on a small electric RC aircraft, and experience in attitude estimation has demonstrated the rapid design cycle achievable by having MATLAB/Simulink as the main development toolchain.



(a)



(b)

Fig. 7: SLUGS/Mentor flight data for (a) adaptive control with rudder failure and (b) *control* DSC load during tests.

REFERENCES

[1] V. Dobrokhodov, I. Kitsios, I. Kaminer, K. Jones, E. Xargay, N. Hovakimyan, C. Cao, M. Lizarraga, I. Gregory, "Flight validation of metrics driven $\mathcal{L}_1$ adaptive control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008.

[2] V. Dobrokhodov, I. Kaminer, I. Kitsios, E. Xargay, N. Hovakimyan, C. Cao, I. Gregory, and L. Valavani, "Experimental validation of l1 adaptive control: The rohrs counterexample in flight," *Journal of Guidance Control and Dynamics*, vol. 34, no. 5, p. 1311, 2011.

[3] C. B. Low, "A trajectory tracking control design for fixed-wing unmanned aerial vehicles," in *IEEE International Conference on Control Applications*, Japan, September 2010, pp. 2118–2123.

[4] D. Jung, J. Ratti, and P. Tsiotras, "Real-time Implementation and Validation of a New Hierarchical Path Planning Scheme of UAVs via Hardware-in-the-Loop Simulation," *Journal of Intelligent and Robotic Systems*, vol. 54, no. 1-3, pp. 163–181, 2009.

[5] I. Kaminer, O. Yakimenko, V. Dobrokhodov, A. Pascoal, N. Hovakimyan, C. Cao, A. Young, and V. Patel, "Coordinated path following for time-critical missions of multiple UAVs via $\mathcal{L}_1$ adaptive output feedback controllers," *AIAA Guidance, Navigation and Control Conference and Exhibit*, August 2007.

[6] V. Dobrokhodov, O. Yakimenko, K. Jones, I. Kaminer, E. Bourakov, I. Kitsios, and M. I. Lizarraga, "New generation of rapid flight test prototyping system for small unmanned air vehicles," *AIAA Modeling and Simulation Technologies Conference Proceedings*, 2007.

[7] M. Lizarraga, G. H. Elkaim, G. Horn, R. Curry, V. Dobrokhodov, and I. Kaminer, "Low cost rapidly reconfigurable uav autopilot for research and development of guidance, navigation and control algorithms," in *ASME/IEEE MESA09*, International Conference on Mechatronic and Embedded Systems and Applications. San Diego, CA: International Conference on Mechatronic and Embedded Systems and Applications, 08/2009 2009.

[8] Cloud Cap Technology, *Piccolo Unmanned Avionics System*, http://www.cloudcaptech.com/, Goodrcih, Hood River, Oregon, 2012.

[9] MicroPilot, "Micropilot: World leader in miniature uav autopilots," http://www.micropilot.com, 2012.

[10] Lockheed Martin, "KESTREL Autopilot," http://www.procerus.com/productsKestrelAutopilot.php, 2012.

[11] P. Brisset, A. Drouin, M. Gorraz, P. Huard, and J. Tyler, "The Paparazzi Solution," *2nd US-European Competition and Workshop on Micro Air Vehicles*, November 2006.

[12] C. Anderson, "Diydrones ardupilot," http://diydrones.com/.

[13] W. Premerlani, "Uav dev board," http://code.google.com/p/gentlenav/.

[14] I. Kaminer, A. Pascoal, E. Xargay, C. Cao, N. Hovakimyan, and V. Dobrokhodov, "3D Path Following for Small UAVs using Commercial Autopilots augmented by $\mathcal{L}_1$ Adaptive Control," *Journal of Guidance Control and Dynamics*, vol. 33, no. 2, 2010.

[15] L. Kerhuel, "Matlab-simulink device driver blockset for PIC/dsPIC microcontrollers," http://www.kerhuel.eu/wiki/index.php5.

[16] L. Meier, "Mavlink: Micro air vehicle communication protocol," http://qgroundcontrol.org/mavlink/start.

[17] M. Lizarraga, "Design, implementation and flight verification of a versatile and rapidly reconfigurable uav gnc research platform," Ph.D. dissertation, Department of Computer Engineering, University of California Santa Cruz, Santa Cruz, CA, December 2009.

[18] R. Curry, M. Lizarraga, and G. Elkaim, "The design of rapidly recon gurable filters for attitude and position determination," *AIAA Infotech Conference*, April 2010.

[19] R. Curry, M. Lizarraga, G. Elkaim, and B. Mairs, "$\mathcal{L}_2^+$, an improved line of sight guidance law for uavs," in *American Control Conference, ACC*, Washington, D.C., June 2013.

[20] E. Zurich, "Qgroundcontrol: Ground control station for small air land water autonomous unmanned systems," http://qgroundcontrol.org/.

[21] N. Hovakimyan and C. Cao, *L1 adaptive control theory: guaranteed robustness with fast adaptation*. Siam, 2010.