AIAA SciTech Forum
23-27 January 2023, National Harbor, MD & Online
AIAA SCITECH 2023 Forum

10.2514/6.2023-2666

GPS-Denied State Estimation for Blue/NDAA Unmanned Multi-Rotor Vehicles

Grant Phillips* and Justin Bradley†
University of Nebraska-Lincoln, Lincoln, NE 68588, USA

Prashant Ganesh‡
University of Florida, Gainesville, FL 32611, USA

Unmanned Aircraft Vehicle (UAV) state estimation and navigation in GPS-denied environments has received a great deal of attention, with several researchers exploring a variety of compensating estimation methods. These methods vary in capability, and usually trade off estimation accuracy for simplicity and fewer resource requirements. More advanced estimation schemes, while capable of providing good state estimates for longer periods of time, may not be suitable for small, limited resource vehicles such as UAVs. Simpler and less-accurate estimation methods, while less capable, are useful for introducing the topic to students as well as helping researchers establish flight capabilities, and may be more suitable on limited hardware. The Autonomous Vehicle Laboratory's (AVL) REEF Estimator was designed to expedite the development of a group's GPS-denied flight capabilities through its simple and modular design. This work seeks to extend the application of the REEF Estimator by adapting it to fit the Ardupilot flight stack so that the estimator may be used on a readily available and NDAA-compliant flight controller, specifically, a Pixhawk Cube Blue. In addition, the REEF Estimator has been containerized to further facilitate its deployment between different vehicle architectures with minimal need for reconfiguration or setup.

I. Introduction

Feedback control of Unmanned Aircraft Vehicles (UAVs) requires information about the vehicle's states, most traditionally, attitude and position. Most directly, a suite of sensors make up an attitude and heading reference system (AHRS) and, together with GPS, provide periodic observations of the vehicle's state which is then utilized to compute a control input. Traditionally, large aerospace systems devote extensive sensor suites (e.g, IMUs, pitot systems) to this effort. However in small UAVs, space is limited, and these sensors may exceed weight/cost constraints. As such, lightweight MEMS suites are often utilized in these "micro" systems, usually at the cost of measurement accuracy. Additionally, in contested, or indoor environments, GPS signals may be degraded or spoofed resulting in vehicles departing from commanded behavior. Lightweight and inexpensive AHRS cannot accurately measure the vehicle's pose over extended periods of time in GPS-denied scenarios without the use of state estimation or filtering.

As a result, the community has devoted significant effort into advanced estimation schemes able to account for more noise in low-cost, lightweight AHRS, as well as methods for improved performance in GPS-denied environments. While many estimation schemes have been devised, they are often highly complex, requiring significant computational effort or additional sensors [1]. Unfortunately, often the more advanced estimation schemes may exceed available computing resources. The need for simple, but effective estimators that are compatible with COTS hardware and autopilot software has led to a recent effort by the University of Florida, in conjunction with AFRL to develop a computationally lightweight estimator for GPS-denied environments with relatively minimal sensor requirements known as the REEF Estimator [2].

The REEF Estimator was intended to be used as a starting point for students and researchers beginning to integrate autonomy into real UAVs by providing a simple velocity controller and velocity estimator. Although the REEF Estimator does not necessarily demonstrate state-of-the-art control and state estimation, it does provide reliable GPS-denied flight at a reduced computational cost. Beyond emerging research labs, the REEF Estimator may also be of value to researchers interested in adequate GPS-denied flight for applications with other computationally heavy tasks such as image processing or trajectory planning. The REEF Estimator has potential to provide tractable GPS-denied navigation

*PhD Student, School of Computing, AIAA Student Member
†Associate Professor, School of Computing, AIAA Associate Fellow
‡Research Assistant Engineer, Department of Mechanical and Aerospace Engineering. AIAA Member

capabilities to UAVs at a lower computational cost, however, its implementation is currently limited to a small class of vehicles due to hardware-software compatibility.

The REEF Estimator was initially released as a ROS package to be implemented alongside a flip32 autopilot running ROSflight flight stack. ROSflight caters to the research community by providing a simple and highly-configurable flight stack, however, it is not intended to be a fully functional autopilot, but rather a minimal base to build research applications upon. UAV researchers in need of more autopilot functionality often utilize the open-source Ardupilot* or PX4† flight stacks, which are frequently used with hardware that conforms to the Pixhawk standard‡.

Although the REEF Estimator stands to benefit UAV researchers, hardware-software compatibility may be acting as a barrier between the REEF Estimator and projects that require the level of functionality provided by Ardupilot or PX4. This work expands upon the REEF Estimator's compatibility by adapting it to fit the Ardupilot flight stack on an NDAA-compliant autopilot (e.g, the Pixhawk Cube Blue). We then evaluate the new Ardupilot-REEF Estimator by comparing in-flight estimator data against "truth" motion capture data. We also incorporate the REEF Estimator within a closed control loop to further evaluate its utility. Finally, we have containerized the REEF Estimator packages into a Docker§ image to further facilitate arbitrary vehicle deployment and minimize configuration and setup.

II. Related Work

GPS-denied navigation has received a great deal of attention, particularly as autonomous flight has become capable on small, low-cost hardware platforms. Flight under such circumstances is difficult since bias and noise in AHRSs and IMUs tends to grow rapidly unless supplemented with a more accurate, and regular GPS signal [3]. This difficulty is exacerbated on constrained Size, Weight, and Power (SWaP) vehicles due to the often lower-resolution hardware, and limited computing power. Although several methods have been presented and tested in the literature, they are typically not generalizable solutions. Most estimation methods depend on environmental features (e.g walls, trees) that must be sensed, the environment plays a vital role in the estimation method, and environmental features can vary widely, making a general solution to GPS-denied estimation difficult. In this section, we discuss inertial sensing and its shortcomings along with sensors that are typically used as supplements to inertial sensing. Next, we review state-of-the-art sensor fusion methods and the trade-offs between them and the REEF Estimator.

A. Inertial Navigation and UAV Sensors

Inertial or acceleration measurements are utilized in a majority GPS-denied estimators as most vehicles have inertial measurement units (IMUs) as part of their AHRS. While in theory the UAV's acceleration could be integrated over time to produce a pose estimate – a method known as dead reckoning – in practice, the integration drifts over long time periods due to the limitations of lightweight MEMS hardware. In order to mitigate estimation drift, additional sensor readings such as visual odometry (VO) and optical flow are fused with IMU measurements [4].

Range finders are frequently used to complement IMU readings in estimators and filters [5], [6]. Sonar and stationary LiDAR can provide direct above ground level altitude measurements up to 100 m. 2D and 3D LiDAR scanners are able to determine the distance between a vehicle and the surrounding obstacles and surfaces. While powerful, such range finders often require costly scanning time or computation to process the results [7].

Computer vision-based sensing for GPS-denied navigation has seen an increase in usage and performance as the processing capability of embedded systems improves [8]. The popularity of visual methods is due to the high level of detail captured by the sensors and subsequent robustness of state estimation algorithms [7]. While computer vision has been implemented in a number of unique ways, these methods tend to require significant computational resources and degrade in performance in featureless environments [9].

Similar to tracking large aircraft, RADAR has been explored as a possible solution to GPS-denied navigation for small UAVs [10]. While the technology is improving, the hardware in its current state is large enough to be taxing on the available space, weight, and power constraints of the vehicle. This, in turn, reduces flight/mission time and may reduce the utility of the UAV.

*https://ardupilot.org/

†https://px4.io/

‡https://pixhawk.org

§https://www.docker.com/

B. Sensor Fusion Methods

Several implementations in the literature mix a variety of sensors with different estimation algorithms via sensor fusion (Kalman or otherwise). Most often, the more accurate and robust pose estimation schemes utilize larger sensor arrays that combine measurements from several of the sensing methods mentioned. For example, the relative navigation framework outlined by Koch et al. [1] utilizes IMU, RGB-D camera, laser scan, and altimeter sensor data. This framework has been demonstrated as a useful state provider in both autonomous indoor (GPS-denied) and outdoor (GPS-degraded) environments, making it one of the more capable estimation schemes designed. However, this particular scheme faces the same hardware-compatibility issues REEF faces with the additional hurdle that no code base is explicitly provided. However, even the most accurate and robust methods cannot account for all possible environments. As is often the case, new sensing methods are often devised to overcome troublesome obstacles, such as glass structures in [11], or to approach a problem differently, similar to the multi-agent cave exploration in [12]. Furthermore, artificial neural networks are being used to fuse data from sensor arrays in order to estimate remote UAV poses [13] [14], bringing state estimation into the deep learning age.

Relative navigation has also been implemented for small UAV with minimal sensor requirements [15], which has been expanded to multi-agent estimation most recently in [16]. In this cooperative relative navigation scheme, IMU, visual odometry, and inter-vehicle measurements are leveraged to enable decentralized, multi-agent control suitable for fixed-wing outdoor navigation in GPS-degraded situations.

LiDAR range finding is often implemented in simultaneous localization and mapping (SLAM) techniques. SLAM methods and their variations are used to estimate a vehicle's pose by constructing a map of the immediate observable surroundings via LIDAR, acoustic, or tactile sensing. Since its introduction in 1995, SLAM continues to evolve as shown by the recent semantic SLAM approach presented by Liu et al. [17] which demonstrates great performance in forested areas.

While the methods mentioned represent the state-of-the art in estimation techniques in terms of accuracy, they also require multiple additional sensors and powerful on-board computers. Battery resources are strained through sensor/computer power consumption along with the additional weight on the frame, which ultimately cuts into flight/mission time. In the interest of preserving vehicle resources, both cyber and physical, simpler estimation methods may be preferred at the cost of estimation performance.

The interest of this work, the REEF Estimator, is a simplified GPS-denied estimator for multicopter UAVs with a relatively light sensor suite. The REEF Estimator is based on the square root formulation for the Partial-Update Kalman filter [18]. This filter provides a more robust version of the Partial-Update Kalman filter with minimal impact on the computation needed. This results in a low-cost estimation scheme that can be accommodated by most companion computers. The REEF Estimator's simplicity makes it ideal for emerging UAV researchers and applications that can benefit from "tractable" GPS-denied navigation.

Although the REEF Estimator may be an appealing tool to researchers, hardware compatibility may be acting as a barrier to a larger user base. Specifically, the REEF Estimator has been tested and designed on a ROSflight platform [19], a low-level flight controller, which lacks some of the upper level functionality provided by more mainstream open source flight stacks such as Ardupilot and PX4. The focus of this work centers on adapting the REEF Estimator to interface with the Ardupilot flight stack, specifically on a Pixhawk-based, NDAA-compliant Cube Blue autopilot. This adaptation will expand the REEF's potential user base, including applications that require NDAA-compliant hardware.

Additionally, in the interest of simplifying the REEF Estimator's deployment, the entire working REEF Estimator has been containerized in a Docker image with associated "docker-compose" scripts that can be easily edited and deployed. Containerization mitigates the overhead of installing support packages and configuration/setup while also protecting against backwards-compatibility issues. Specifically this work contributes:

- Adapting the REEF Estimator to be compatible with Pixhawk-based flight controllers
- Containerizing the REEF Estimator following the Docker paradigm, and releasing publicly the Docker images and associated docker-compose scripts
- Evaluation of the REEF Estimator's performance following the platform changes
- Evaluation of the REEF Estimator's performance in a closed-loop flight controller

III. Adaptation and Containerization of REEF Estimator

In this section, we will discuss the steps taken to adapt the REEF Estimator to Pixhawk flight controllers. This adaptation includes changes to the vehicle's hardware, changes to the ROS structure of the REEF Estimator, and slight changes to the REEF Estimator source code. Additionally, this section outlines the containerization of the REEF

Estimator and the benefits of containerization for robotics.

A. UAV Hardware

The REEF Estimator and controller evaluations were performed on a custom 550-size hex-copter with a ProFiCNC Cube Blue flight controller. The Cube Blue houses a set of sensors that make up a light AHRS (3 IMUs, 3 gyroscopes, a barometer, and magnetometer). An auxillary sonar sensor is used to supplement the barometer and IMU in altitude estimation. The Cube also handles the low-level flight controls for the UAV, and it exchanges it's sensor data and flight controls with an on-board Raspberry Pi 4 Model B (RPi). The RPi uses sensor data from the Cube as well as a forward-facing Intel realsense D435i RGB-D camera as inputs to a set of processes that make up the REEF Estimator. The test vehicle and sensors are presented in Figure 1

Fig. 1 Custom NIMBUS 550-size hexcopter outfitted with sensors used for REEF estimator evaluation.

B. REEF-Pixhawk Adaptation

1. Sensors

The first step in adapting the REEF Estimator was to build the hardware platform and mount the external sensors on the vehicle. The build materials are listed in Table 1. The Cube Blue utilizes an Extended Kalman Filter (EKF) to fuse readings from the internal IMUs, barometer, and external sonar sensor to produce attitude and altitude estimates. The Cube Blue also communicates IMU, attitude, and sonar information to the RPi over a hard serial connection via MAVLink messages. MAVLink[¶] (Micro Air Vehicle Link) is a lightweight messaging protocol specifically designed for sending messages and commands between drones and drone components.

Table 1 Vehicle Sensors

Sensor Type	Detail
Autopilot	Pixhawk: Cube Blue
Companion Computer	Raspberry Pi 4, 8 GB
RGB-D Camera	Intel Realsense 435i
Sonar Sensor	Maxbotics MaxSonar EZ4

[¶]MAVLink.io

On the RPi side, the Robot Operating System (ROS) provides the communication structure between the different processes that make up the REEF Estimator (see Figure 2). Each process (referred to as a "node") has information (referred to as a "topic") that it receives, or "subscribes" to, and other information/topics that it sends, or "publishes". For the REEF Estimator, MAVLink messages that are exchanged on the RPi are handled by a special node: MAVROS [‖]. MAVROS reads an incoming MAVLink message stream and converts the messages to ROS topics for IMU, sonar, and attitude readings. Additionally, MAVROS is capable of sending commands (such as roll, pitch, yaw, and thrust) to the flight controller. The RGB-D camera is handled by a ROS wrapper for the realsense camera (`realsense2_camera` [**]); This makes the RGB-D data readily available as ROS topics for other nodes to subscribe to and use. The nodes that make up the REEF Estimator use the sensor-data topics to eventually produce an estimate for the UAV's velocity.

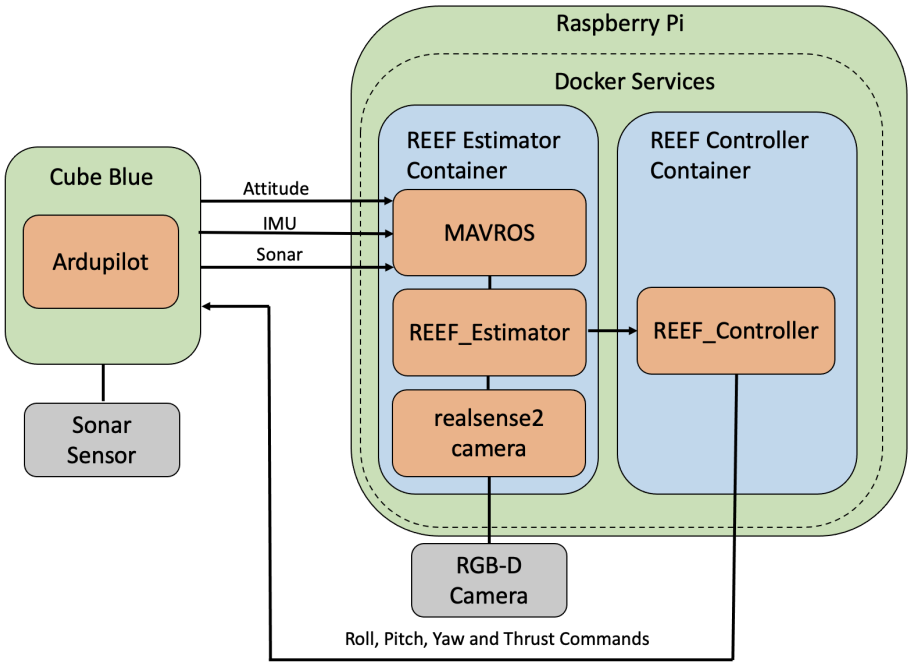

Fig. 2 Autopilot, companion computer, and software configuration. The names of the software packages in this figure are presented as they are provided.

2. Software Integration

The REEF Estimator was designed as a general purpose estimation tool that, in theory, can be used on a variety of UAV classes (e.g., fixed wing, quad-rotor, hex-rotor) as long as the sensor data is sent to the REEF Estimator at an appropriate rate (~200 Hz). A majority of the software changes needed to adapt the REEF Estimator to the Pixhwak/Cube Blue occur at the ROS level. That is, most of the adaptation is done by re-mapping ROS topics and changing ROS launch files and *not* modifying the REEF Estimator source code; this approach reduces the risk of introducing bugs to the original REEF Estimator, and could be considered "configuration" issues that can easily be solved via containerization. Software changes that could not be accomplished at the ROS level (e.g., changing coordinate frames) are handled by a simple transformation node between MAVROS and the main REEF Estimator node.

C. Containerization

Containerization has gained popularity as a tool for developing and distributing software applications between host machines with different processors and operating systems. Applications are built in layers of containers, starting with a "base" layer (often, but not always, a bare-bones Linux distribution). A single script, known as a Dockerfile, contains all of the instructions needed to build a Docker image (a non-running container). The Dockerfile may compile the vehicle

[‖] http://wiki.ros.org/mavros

[**] http://wiki.ros.org/realsense2_camera

software into binaries which can be easily configured to match the vehicle's unique characteristics. The Docker image can be easily shared and instantiated by machines with similar hardware resulting in a common configuration setup across many machines. Running an image will create an instance of the container described by the Dockerfile.

Containers, specifically Docker, have seen increasing use in robotics [20]. Perhaps most obviously, containers significantly enhance the ability to reuse software on slightly different platforms [21, 22], or adjacently, ease the ability to deploy the same software onto many identical agents simultaneously. In addition, containers have been used as a mechanism for sharing robotics-based data [23], deploying robotics-based applications [24], and even deploying robotics-based education platforms [25].

The Dockerfiles for the REEF Estimator are based on the Ubuntu 18.04 image (see Figure 3). From the base, basic dependencies and support packages are installed (e.g., python3, git, OpenCV, ROS Melodic); then, the ROS packages for the REEF Estimator are copied and compiled, completing the REEF Estimator image.

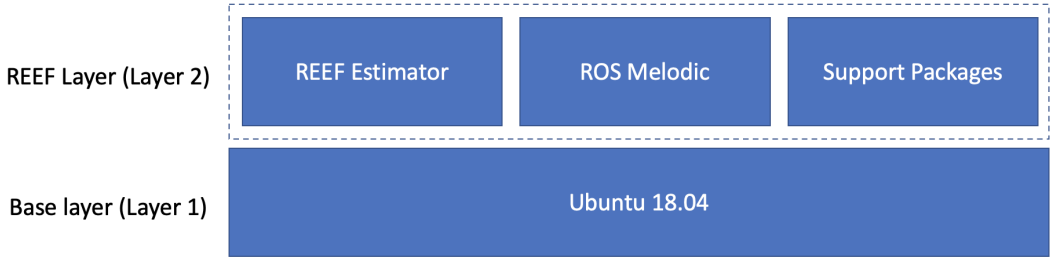

Fig. 3 Structure of Docker layers. The image consists of two layers, "Ubuntu 18.04", serving as a base layer, upon which the next layer is built. The top layer consists of the "REEF Estimator" code, "ROS Melodic," and various "Support Packages."

The REEF Estimator packages were divided into two containers: a container for the estimator only and an optional container that adds the controller. The estimator-only container is reserved for the REEF Estimator, MAVROS, and camera driver (see Figure 2); this serves the basic functionality of the REEF Estimator: generate pose estimates. This separation allows the REEF Estimator to be used independently of the controller.

Docker also provides several features that further improve image deployment. Docker images can be built manually on a local host machine or a pre-built image can be pulled from an online repository (e.g., Dockerhub or GitHub). When running an image from an online repository, Docker will automatically download the latest versions of the image layers before running, keeping container management simple. Furthermore, docker-compose is a tool that can configure and launch several containers at once according to instructions in a docker-compose YAML script. Docker-compose is capable of pulling images from online repositories, executing commands once the container is running (e.g., roslaunch), restarting the containers when the host reboots. Docker-compose further automates deployment of larger containerized applications.

To illustrate the power of containerization for UAVs, we are currently developing a trajectory planner for a swarm of 10-20 quadcopters using the REEF Estimator. Each time a new update to the planner or estimator needs to be deployed (without containerization) a developer must manually pull and compile new code to each companion computer. Additionally, each time a vehicle is rebooted, the planner and estimator must also be manually restarted. In the context of the swarm, this process is time-consuming and vulnerable to human error (e.g., a single compile, pull, or startup can be missed). With containerization and docker-compose, each vehicle in the swarm only needs the docker-compose file, which will automatically pull the planner and estimator images as needed, restart them on reboot, and contains the configuration parameters needed for each agent. This not only potentially saves significant deployment time that can now be used towards development and testing, but also lowers the risk of mis-configuration and missed steps in a deployment sequence.

IV. Evaluation

Testing and evaluation were done in stages in order to ensure that each component of the REEF Estimator bundle was functioning properly before adding more sensors or controller functionality. The testing was conducted in an indoor facility on a custom F550 hexcopter we dub a "NIMBUS 550," with the sensor suite described in Section III.A and seen in Figure 2. "Ground truth" position and velocity information are provided by a Vicon motion capture system. Data

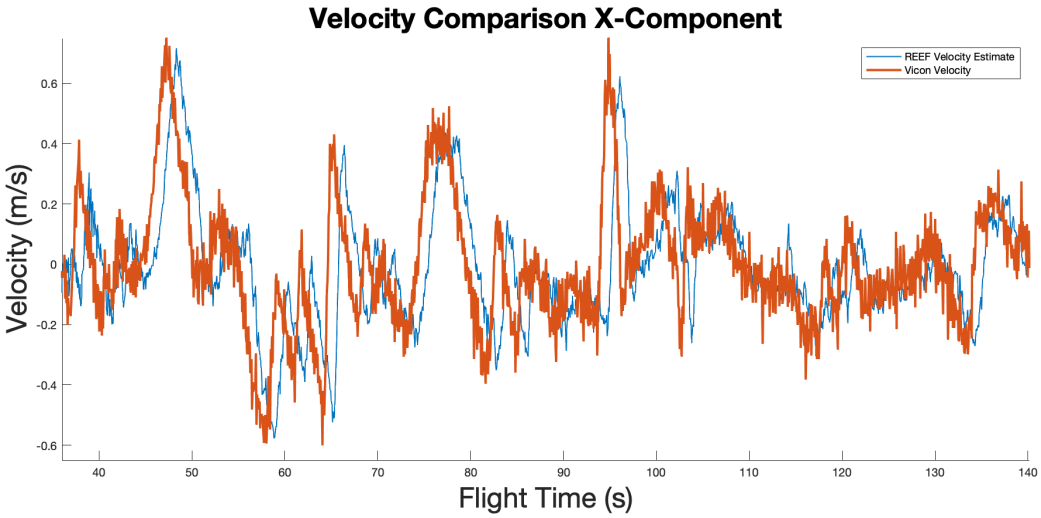

Fig. 4 In this Figure, the REEF estimates for the vehicle's X-axis velocity are compared to the truth velocity from the Vicon motion capture system. Although the velocity measurements and ground truth are the same data, it can be seen that the REEF estimates closely mimic the Vicon velocity. This indicates that the coordinate frames between REEF and Vicon are properly transformed and that the IMU is refreshing the correct rate ($\approx$ 200 Hz).

gathered from testing was captured using ROS bags recorded from flights.

A. REEF Estimator and REEF Control Evaluation

The REEF Estimator requires three streams of information in order to produce velocity estimates: X (east) and Y (north)-axis velocity measurements, altitude updates derived from sonar, and linear acceleration updates from the IMU. For clarity: a velocity "estimate" refers to a filtered/derived state estimate and a "measurement" refers to an output from a sensor. The estimator executes its velocity calculations every time an IMU message is received. Velocity measurements are generated by a visual odometry (VO) process (DEMO RGBD) at the camera's frame rate: 30 frames per second in the realsense's case. The entire estimator consists of several processes that are either driving sensors or filtering data. As such, the evaluation was broken down into 3 targeted stages in order to help isolate sources of bugs and ensure good performance of different parts of the system: 1) Estimator only and Vicon velocity measurements 2) REEF Controller with estimator feedback and Vicon velocity measurements 3) Estimator only with VO velocity measurements.

B. Estimation with Vicon Measurements

The flight tests for the first evaluation stage confirmed that the REEF Estimator receives IMU data and generates velocity estimates. At this stage, odometry information is provided by Vicon at 200 Hz in place of the RGB-D camera; using Vicon isolates the IMU and main estimation node as the variables in this stage, which is helpful for aligning frames and evaluating the estimator in ideal conditions. The velocity estimates are compared against the velocity information provided by Vicon. This stage of evaluation is mostly qualitative since the odometry data and ground truth data are the same. Figure 4 shows an example of the REEF velocity estimates closely following the truth velocities from Vicon. From Figure 4 the X-axis velocity plot indicates that the REEF Estimator is responding to the IMU data stream and tracking the Vicon velocity measurements.

C. Control with Vicon Measurements

The second evaluation stage closed the control loop between the estimator and a PID controller provided by the UF Autonomous Vehicle Lab (AVL REEF Controller) that tracks a X, Y velocity reference and maintains a constant altitude. The controller uses REEF Estimator feedback to generate roll, pitch, yaw, and thrust commands that are sent via MAVROS to the Cube Blue. Similar to the previous stage, Vicon odometry is used in place of the RGB-D camera. Velocity reference commands are sent to the REEF Controller from an Xbox 360 wireless controller. Flight tests were conducted to confirm that the REEF Controller + Estimator are capable of maintaining a stable hover and move the

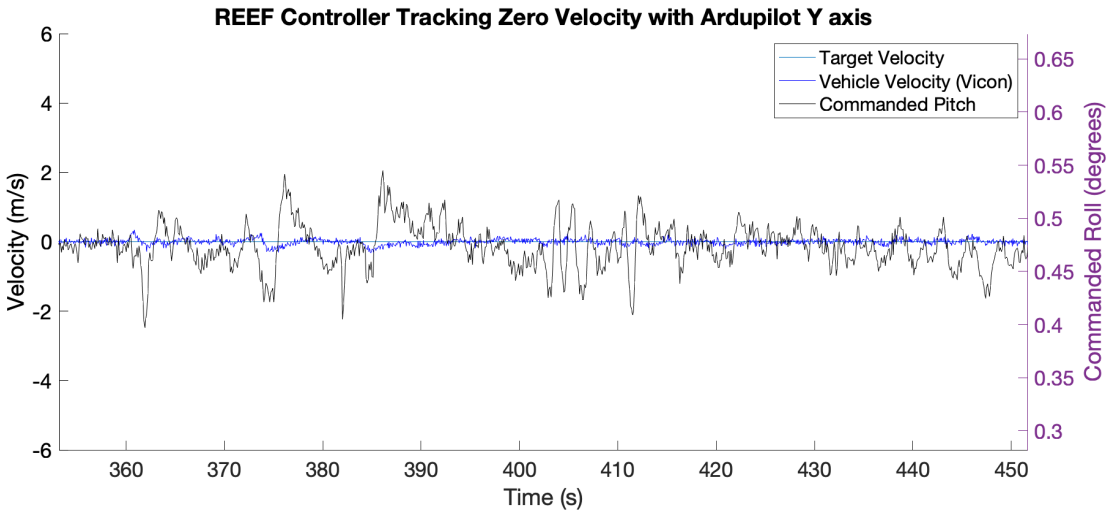

Fig. 5 In this figure, the REEF Controller is shown regulating the vehicle's velocity through roll, pitch, yaw, and thrust commands (only pitch is shown in this figure). The vehicle's velocity can be seen to start drifting from 0 m/s around t =375, which in turn, triggers a response from the controller and returns the vehicle back to its reference velocity.

vehicle in an altitude-hold fashion. The REEF Controller with REEF Estimator feedback can be seen regulating the vehicle's X-axis velocity in Figure 5.

D. Estimation with VO Measurements

The third evaluation stage aimed to replace the Vicon velocity provider with the RGB-D camera measurements; this configuration embodies the main goal of the REEF Estimator: GPS-denied state estimation. Preliminary experiments indicated that the full estimator application overloads the RPi's capabilities. In this state, we found that the closed loop controller with the REEF estimator can only provide reasonable control authority in non-aggressive flight and gentle hovers with few disturbances; additionally, camera frame rates become inconsistent and jump between 7-15 FPS vs. the default 30 FPS. In order to continue evaluating the adapted estimator, all computation was moved from the RPi to an off-board desktop computer outfitted with a full-sized processor and GPU. On the desktop, the REEF Estimator is able to run at the recommended 200 Hz with VO executing at the nominal 30 Hz. Figure 6 shows the estimator running at 200 Hz with 30 FPS camera streams. In this figure, the estimates are able to closely follow an oscillating motion.

E. Discussion of Results

Although the entire estimator running on a single RPi overloads its processor, it was found that the REEF Estimator is able to reasonably estimate the vehicle's velocity in non-aggressive flights and simple hovers. Figure 7 shows that estimates generated from an RPi in overload can reasonably follow the actual vehicle velocity, but as the vehicle's path becomes more aggressive, the estimates drift rather quickly (this can be seen in Figure 7 after 500 s). This is most likely due to stale measurements being input to the estimation algorithm. This happens most often during dynamically fast portions of the flight envelope. That is, if there are no disturbances and no movement, then little estimation is needed because the state has not appreciably changed. However, as the vehicle's state begins to change faster than estimates can be generated, errors due to lost state information propagate through the estimation process and cause the estimate to critically drift from the actual vehicle state.

One proposed solution to overcome overloading the RPi, as is often the case (and even implemented in this work), is to use a more powerful computer. While this option may be simple to implement, a more-capable onboard computer will likely further burden the vehicles scarce SWaP resources, ultimately resulting in shorter flight times. Additionally, as we've recently seen, upgrading to a better onboard processor may be infeasible given current supply chain issues, with product lead times up from a couple of weeks to several months or years.

Another solution would leverage the distributed computing capability provided by ROS. ROS allows for two or more

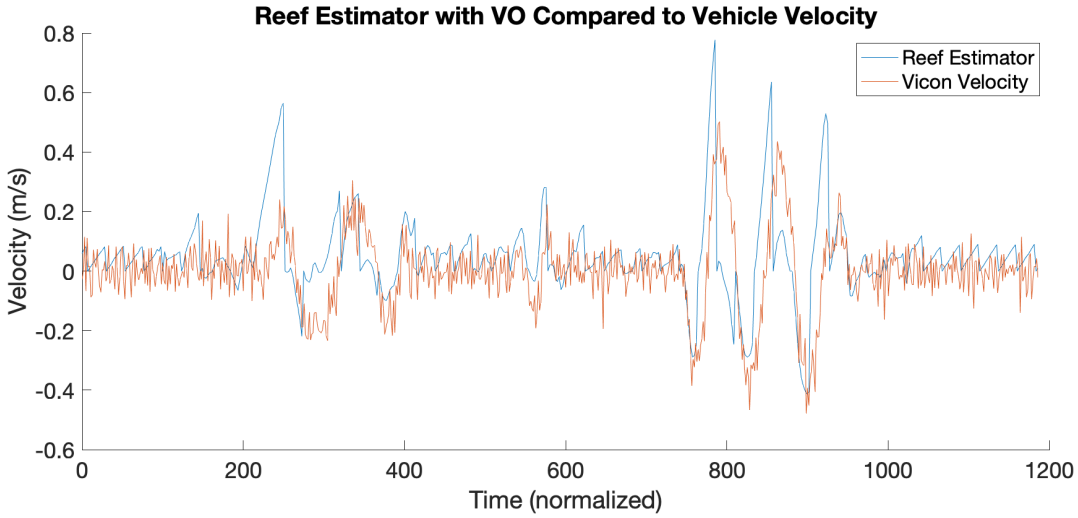

Fig. 6 In this figure, the REEF Estimator with Visual odomentry measurements is shown to replicate the Vicon-measured velocities. Near t=740, the vehicle is excited along the X axis and the REEF Estimator velocities can be seen to follow the vehicle's oscillating motion .

networked computers to execute different nodes locally, but share topics and messages as if the nodes were run on a single machine. In this case, the realsense camera driver and MAVROS would be executed by the onboard RPi and the image processing and estimation would be offloaded to a ground station computer. This distributed configuration may be feasible as the RPi was able to run the camera driver alone at 30 FPS. However, this configuration creates a network "tether" between the vehicle and ground station for even basic GPS-denied control of the vehicle, and it is likely to also introduce latency or packet loss between estimation calculations and flight command execution, resulting in a different set of challenges.

As mentioned in section IV.D, the overloaded RPi was able to run the estimator with a lower camera FPS when the vehicle was stable and the state was relatively constant, but failed when the vehicle began to speed up, because the cameras were not updating fast enough. This leads us to believe that there were times during these flights where updating the sensor measurements at 7 Hz was sufficient for the required control needed under the existing conditions. But those conditions are possibly brief in real-world conditions and environmental disturbances where requested control maneuvers require higher estimator rates. As built and configured, the REEF estimator is not dynamic in its execution and therefore simply cannot be executed on Raspberry Pi 4 hardware for a UAV of this size. As such, another possible solution to overcoming the resource limitations of the RPi would be to implement the REEF Estimator in a resource-aware, co-regulated manner.

Co-regulation has been used in controllers to dynamically adjust the rate at which control inputs are sent to a system based on the performance of the system under given conditions [26]. In brief, co-regulated systems increase the frequency of control calculations and efforts as the error between the system state and its reference increases. As the system moves closer to the reference, the control frequency decreases, freeing up resources for other processes. The work we demonstrate here shows that a similar dynamic applies to estimation, wherein during some portions of the flight envelope fewer estimates are needed since little control authority is needed, while at other times increased estimation is required to combat environment disturbances and complete controlled maneuvers. Such a strategy would be readily applicable to estimation and could allows for more complex estimation schemes to be accomplished with fewer onboard computational resources.

V. Conclusions

In this work, we have outlined the steps taken to adapt a simple, novel GPS-denied estimator to a class of NDAA-compliant flight controllers. This adaption will allow the computationally less expensive REEF Estimator to be used as a state provider for projects and missions that require NDAA-comliant hardware (or "blue") hardware. The evaluation results create confidence that the estimator is capable of adequate feedback for velocity control. Additionally, we outline

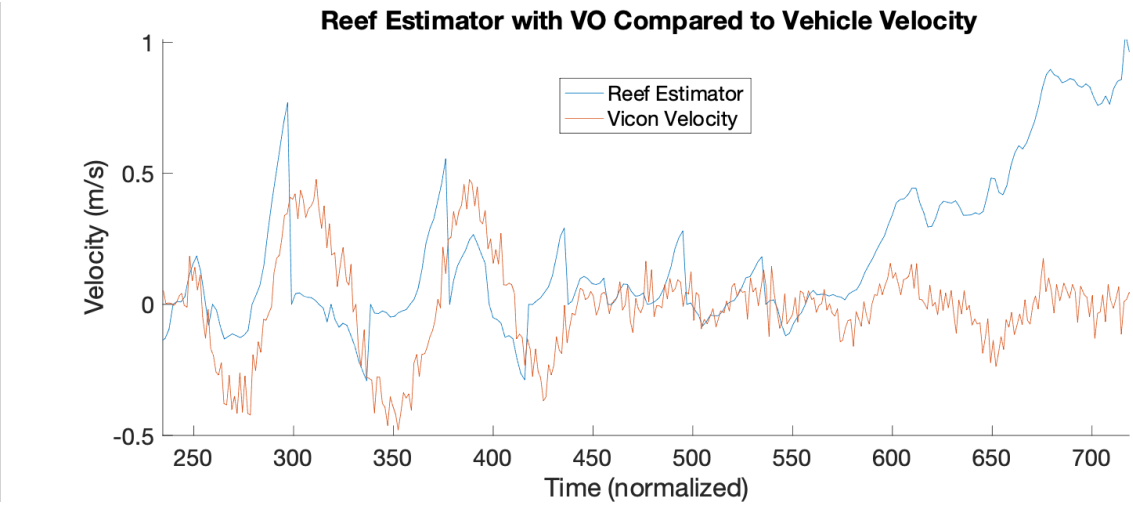

Fig. 7 REEF Estimator X-axis velocity estimates running on an overloaded RPi. The RPi manages to mimic the Vicon velocity measurements until Time≈600, at which point an aggressive maneuver "shakes" the estimated state from the actual velocity.

the benefits of containerization for robotics deployments and how the REEF Estimator was containerized to facilitate its deployment in future applications. Our results highlight the computational complexity of even simplified UAV estimation schemes underscoring the need for estimator designs that consider computation and communication from the beginning. Finally, we propose future work to further improve the REEF Estimator's performance on resource-constrained cyber systems.

VI. Acknowledgements

This work was partially supported by the National Science Foundation under grants NSF-2047971 and NSF-2221648.

References

[1] Koch, D. P., Wheeler, D. O., Beard, R. W., McLain, T. W., and Brink, K. M., "Relative multiplicative extended Kalman filter for observable GPS-denied navigation," *The International Journal of Robotics Research*, Vol. 39, No. 9, 2020, pp. 1085–1121. https://doi.org/10.1177/0278364920903094, URL https://doi.org/10.1177/0278364920903094.

[2] Ramos, J. H., Ganesh, P., Warke, W., Volle, K., and Brink, K., "REEF Estimator: A Simplified Open Source Estimator and Controller for Multirotors," *2019 IEEE National Aerospace and Electronics Conference (NAECON)*, 2019, pp. 606–613. https://doi.org/10.1109/NAECON46414.2019.9058099.

[3] Anbu, N. A., and Jayaprasanth, D., "Integration of Inertial Navigation System with Global Positioning System using Extended Kalman Filter," *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2019, pp. 789–794. https://doi.org/10.1109/ICSSIT46314.2019.8987949.

[4] Scaramuzza, D., and Zhang, Z., "Visual-Inertial Odometry of Aerial Robots," , 2019. https://doi.org/10.48550/ARXIV.1906.03289, URL https://arxiv.org/abs/1906.03289.

[5] Bachrach, A., Prentice, S., He, R., and Roy, N., "RANGE–Robust autonomous navigation in GPS-denied environments," *Journal of Field Robotics*, Vol. 28, No. 5, 2011, pp. 644–666.

[6] López, E., Barea, R., Gómez, A., Saltos, Á., Bergasa, L. M., Molinos, E. J., and Nemra, A., "Indoor SLAM for micro aerial vehicles using visual and laser sensor fusion," *Robot 2015: Second Iberian Robotics Conference*, Springer, 2016, pp. 531–542.

[7] Gyagenda, N., Hatilima, J. V., Roth, H., and Zhmud, V., "A review of GNSS-independent UAV navigation techniques," *Robotics and Autonomous Systems*, Vol. 152, 2022, p. 104069. https://doi.org/https://doi.org/10.1016/j.robot.2022.104069, URL https://www.sciencedirect.com/science/article/pii/S0921889022000343.

[8] Bengtsson, E., "Image Processing and Its Hardware Support Analysis vs Synthesis - Historical Trends," *Image Analysis*, edited by P. Sharma and F. M. Bianchi, Springer International Publishing, Cham, 2017, pp. 3–14.

[9] Yang, N., Wang, R., and Cremers, D., "Feature-based or direct: An evaluation of monocular visual odometry," *arXiv preprint arXiv:1705.04300*, 2017, pp. 1–12.

[10] Nguyen, T.-M., Nguyen, T. H., Cao, M., Qiu, Z., and Xie, L., "Integrated uwb-vision approach for autonomous docking of uavs in gps-denied environments," *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9603–9609.

[11] Zhang, T., Chong, Z. J., Qin, B., Fu, J. G. M., Pendleton, S., and Ang, M. H., "Sensor fusion for localization, mapping and navigation in an indoor environment," *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, 2014, pp. 1–6. https://doi.org/10.1109/HNICEM.2014.7016188.

[12] Best, G., Garg, R., Keller, J., Hollinger, G., and Scherer, S., "Resilient Multi-Sensor Exploration of Multifarious Environments with a Team of Aerial Robots," *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, 2022. https://doi.org/10.15607/RSS.2022.XVIII.004.

[13] Aledhari, M., Razzak, R., Parizi, R. M., and Srivastava, G., "Sensor Fusion for Drone Detection," *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, 2021, pp. 1–7. https://doi.org/10.1109/VTC2021-Spring51267.2021.9448699.

[14] Samaras, S., Diamantidou, E., Ataloglou, D., Sakellariou, N., Vafeiadis, A., Magoulianitis, V., Lalas, A., Dimou, A., Zarpalas, D., Votis, K., Daras, P., and Tzovaras, D., "Deep Learning on Multi Sensor Data for Counter UAV Applications—A Systematic Review," *Sensors*, Vol. 19, No. 22, 2019. https://doi.org/10.3390/s19224837, URL https://www.mdpi.com/1424-8220/19/22/4837.

[15] Ellingson, G., Brink, K., and McLain, T., "Relative navigation of fixed-wing aircraft in GPS-denied environments," *NAVIGATION: Journal of the Institute of Navigation*, Vol. 67, No. 2, 2020, pp. 255–273. https://doi.org/10.1002/navi.364, URL https://navi.ion.org/content/67/2/255.

[16] Ellingson, G., Brink, K., and McLain, T., "Cooperative Relative Navigation of Multiple Aircraft in Global Positioning System-Denied/Degraded Environments," *Journal of Aerospace Information Systems*, Vol. 17, No. 8, 2020, pp. 470–480. https://doi.org/10.2514/1.I010802, URL https://doi.org/10.2514/1.I010802.

[17] Liu, X., Nardari, G. V., Ojeda, F. C., Tao, Y., Zhou, A., Donnelly, T., Qu, C., Chen, S. W., Romero, R. A., Taylor, C. J., and Kumar, V., "Large-scale Autonomous Flight with Real-time Semantic SLAM under Dense Forest Canopy," *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[18] Ramos, J. H., Brink, K. M., and Hurtado, J. E., "Square Root Partial-Update Kalman Filter," *2019 22th International Conference on Information Fusion (FUSION)*, 2019, pp. 1–8.

[19] Jackson, J., Ellingson, G., and McLain, T., "ROSflight: A lightweight, inexpensive MAV research and development tool," *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2016, pp. 758–762. https://doi.org/10.1109/ICUAS.2016.7502584.

[20] White, R., and Christensen, H., "ROS and Docker," *Robot Operating System (ROS)*, Springer, 2017, pp. 285–307.

[21] Cervera, E., "Try to start it! the challenge of reusing code in robotics research," *IEEE Robotics and Automation Letters*, Vol. 4, No. 1, 2018, pp. 49–56.

[22] Mabry, R., Ardonne, J., Weaver, J. N., Lucas, D., and Bays, M. J., "Maritime autonomy in a box: Building a quickly-deployable autonomy solution using the docker container environment," *OCEANS 2016 MTS/IEEE Monterey*, IEEE, 2016, pp. 1–6.

[23] Pörtner, A., Hoffmann, M., Zug, S., and Knig, M., "SwarmRob: A Docker-Based Toolkit for Reproducibility and Sharing of Experimental Artifacts in Robotics Research," *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2018, pp. 325–332.

[24] Melo, P., Arrais, R., and Veiga, G., "Development and Deployment of Complex Robotic Applications using Containerized Infrastructures," *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, IEEE, 2021, pp. 1–8.

[25] Roldán-Álvarez, D., Mahna, S., and Cañas, J. M., "A ROS-based Open Web Platform for Intelligent Robotics Education," *International Conference on Robotics in Education (RiE)*, Springer, 2021, pp. 243–255.

[26] Bradley, J. M., and Atkins, E. M., "Optimization and Control of Cyber-Physical Vehicle Systems," *Sensors*, Vol. 15, No. 9, 2015, pp. 23020–23049. https://doi.org/10.3390/s150923020, URL https://www.mdpi.com/1424-8220/15/9/23020.