Autonomous Spacecraft Inspection with Free-Flying Drones

Sami Mian*, Tyler Garrett*, Alexander Glandon†, Christopher Manderino*,
Swee Balachandran‡, César A. Muñoz§, and Chester V. Dolph§
Email: {sam415, tmg61, clm199}@pitt.edu, aglan001@odu.edu,
{sweewarman.balachandran, cesar.a.munoz, chester.v.dolph}@nasa.gov

*NSF SHREC Center, University of Pittsburgh, Pittsburgh, PA, USA
†Old Dominion University, Norfolk, VA, USA
‡National Institute of Aerospace, Hampton, VA, USA
§NASA Langley Research Center, Hampton, VA, USA

***Abstract*—This paper describes a proof-of-concept mission demonstrating a multi-agent system performing visual inspection of damage sustained by a spacecraft. Free-flying satellites, simulated by unmanned aerial vehicles (UAVs), autonomously fly around a mock space module maximizing the search space for damage detection. The free-flyers are responsible for independently coordinating their flights to avoid collision with the space module and each other, while executing mission tasks. Damage analysis on the surface of the mock space module is performed in real-time using video from each free-flyer. Three-dimensional modeling is deployed offline to supplement and improve damage detection. This approach demonstrates the feasibility of deploying real space systems for damage detection, where 2D analysis can quickly determine region of interest and 3D visualization can produce a human-navigable virtual environment with depth perspective for further investigation.**

***Index Terms*—unmanned aerial vehicle (UAV), multi-agent cooperation, computer vision, autonomous systems, free-flyer spacecraft, in-space assembly**

I. Introduction

In the past 20 years, space technology has rapidly evolved, presenting new challenges to a growing number of spacecraft in Earth's orbit. A major hazard for spacecraft is structural damage from collision with orbital debris or ablation. Structural damage may degrade performance and, in the worst case, cause catastrophic failure. According to [1], debris larger than 1cm can cause significant damage to a satellite endangering the spacecraft or its mission. As of January 2019, there are nearly one million pieces of space debris greater than 1cm in length estimated to be orbiting around Earth. The number of pieces less than 1cm that could still cause sensor damage is estimated to be over 128 million. As the United States and other countries take aim for new, large vessels like the Lunar Orbital Platform, these problems will continue to threaten the next generation of spacecraft.

The works in [2], [3] consider detecting damage on satellites prior to being launched into space. In contrast to those works, the research in this paper considers the problem of damage detection for a spacecraft after launch and in-orbit. This effort simulates a swarm of autonomous free-flyer satellites surveying a larger spacecraft for possible damage. In order to test the feasibility of the proposed damage detection system, a testbed is developed to simulate several free-flyer satellites working in unison to scan and inspect a simulated larger satellite body. Multiple unmanned aerial vehicles (UAVs) are used to simulate a free-flyer swarm. Each UAV is controlled using NASA's Independent Configurable Architecture for Reliable Operations of Unmanned Systems (ICAROUS) [4]. ICAROUS is an onboard software architecture intended to enable the development of autonomous UAV operations. ICAROUS consists of several distributed applications communicating over a software bus provided by NASA's core Flight System (cFS) [5]. This work extends ICAROUS to autonomous spaceflight for in-orbit systems. In particular, a 2D damage analysis application is developed for real-time damage detection via video feed. Additionally, several post-processing techniques were used to create a 3D reconstruction of the object-of-interest, including visible damage, for further post-mission analysis. This work demonstrates the viability of using ICAROUS on a swarm of free-flyers for detecting external damage on spacecraft in orbit.

II. Background

From the SPHERES project to Astrobees and Int-Ball, over the past decade, many teams have engaged long running experiments with free-flying small satellites. This work approaches mock missions for these maneuverable space robots using UAVs running ICAROUS.

A. Using UAVs for Damage Detection

Several studies have been conducted on the use of a UAV mounted camera for visual inspection of a building [6], [7]. More recent work has focused on sensor fusion techniques, combining sensor data from multiple UAVs to create more accurate analysis of damage to building structures [8]. Further

work has taken advantage of a UAV's capabilities to move in three dimensions using LIDAR and IR systems to create highly accurate 3D maps [9], [10], as well as perform group sensing tasks, such as search and rescue operations [11]. Usually referred to as a "swarm," a large group of UAVs is capable of cooperating to achieve a common goal or collective behavior [12]–[14]. Swarms also provide greater robustness against mission failures, via redundancy and error checking [15], [16]. Often, all agents in a swarm use a common communication platform for coordination. Some swarms are organized by a single leader; decentralized swarm models may require each UAV retain individual autonomy to make its own choices but share data. This work develops a scalable multi-agent system to utilize as many agents as possible to conduct inspections, drawing on work from the field of swarm robotics.

B. Overview of cFS

cFS is a mission framework for flight software applications developed at NASA Goddard Space Flight Center (GSFC). It consists of a dynamic runtime environment, layered software systems, and a component-based design [5]. cFS has a layered architecture that supports a variety of software and hardware platforms. cFS also provides a standardized application programming interface (API) for easier application development. The cFS software has been designed for spaceflight systems and is bundled with a variety of tools that help develop robust, safety-critical code for mission success.

C. Overview of ICAROUS

ICAROUS is an onboard software capability for UAVs developed at NASA Langley Research Center [4]. ICAROUS is intended to enable autonomous decision making and to provide functionalities needed for beyond visual line of sight UAS operations. ICAROUS consists of several applications communicating over a software bus provided by cFS. ICAROUS runs on an onboard companion computer, receiving data from various sensors and sending commands to an autopilot to maneuver around obstacles, to enforce adherence to a predetermined flight path, or to avoid intruders in the airspace. ICAROUS provides path planning [17], sense and avoid [18], and merging and spacing [19] for cFS-based systems. This work uses ICAROUS as the primary onboard mission planning software for free-flyers in-orbit. A diagram of the ICAROUS system is shown in Figure 1.

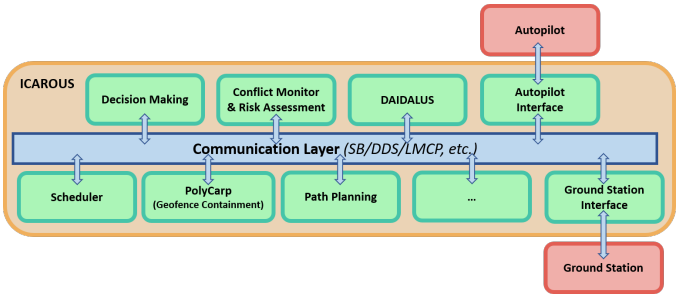


Fig. 1. ICAROUS Architecture

D. Computer Vision Background

The 2D damage analysis application performs image segmentation for the mock satellite and highlights of detected damage. Image segmentation for objects is a well-studied computer vision problem. Some techniques used in this research effort are described below.

Otsu thresholding is a technique for extracting a binary mask from an image. A foreground object can be segmented from a background object if the intensities are different. A threshold intensity level that separates low and high intensity regions with minimum intra class variance is used. Otsu can also be applied adaptively, i.e., as image region dependent [20]. Template matching is another technique for object segmentation. Section 1.2 of [21] describes basic template matching as searching the image for a subregion (or vector) with the smallest distance to the template vector. Color matching is used to segment an object of a particular color. Global thresholding generalizes intensity based thresholding [22]. In the methods section, the global thresholding method is extended to a color ratio based thresholding, which better suits this application.

After segmentation, highlighting of damage on the mock satellite is posed as a filtering problem. For each window, the goal is to give a binary result of normal or damaged. Sobel edge detection is a filtering technique based on gradient calculation in the vertical and horizontal image directions. The sobel method involves 2D filtering with a kernel representing a directional derivative [23].

Alternatively, convolutional neural networks (CNNs) are a deep learning technique that takes image input and can return classification (or detection) output. CNNs employ weight sharing to enable effective training for a given function on high-dimensional image input [24].

The 2D damage detection software is developed as a cFS application for use in ICAROUS during flight. The 3D modeling is performed for an object from a gallery of 2D images. A technique called photogrammetry is used to perform this function. The toolkit used in this work is AliceVision Meshroom™ [25].

III. System Design

This work deploys ICAROUS with cFS in UAV mission computers to simulate free-flyers operating in orbit. New modules for ICAROUS provide high-level mission management and multi-agent coordination. This work also develops a cFS Vicon™ interface for indoor localization during research and testing. Furthermore, a novel computer vision module is implemented in cFS for accurately detecting damage in-situ.

A. System Software Architecture

Three new modules are added to ICAROUS for the free-flyer damage-detection mission: Cognition, Guidance, and Coordination. Figure 2 shows the various software modules that comprise the flight software system used for controlling the free-flyers.

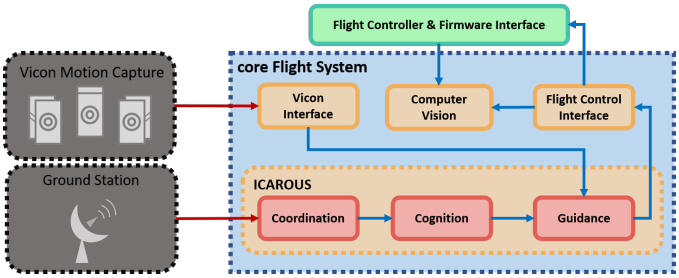


Fig. 2. Flight Software Architecture

Cognition determines various levels of mission tasks for each of the free-flyer including takeoff/land, assigning specific waypoints for each free-flyer, and positioning for capturing data with available sensors.

Guidance issues low-level commands to each free-flyer based on their allocated tasks, such as changes in directional velocity, position estimation, and local trajectory planning functions.

Coordination manages the multi-agent aspects of the mission. This application accepts mission input from the ground station, determines how to split up the mission parameters/tasks based on requirements and number of agents available to deploy. The Coordination application also handles the dynamic addition or loss platforms of free-flyer workers at any time during the mission.

B. Platform Support Applications

1) Hardware Interface: In addition to ICAROUS suite applications, a firmware interface module allows ICAROUS to interface with the free-flyer firmware.

2) Positioning System: As autonomous systems, the free-flyers require the ability to accurately determine position in orbit with respect to themselves and an object of interest. To use traditional GPS localization indoors, a cFS Vicon interface application is developed. This application translates local area positioning to GPS coordinates for real-time autonomous navigation using a Vicon motion capture (mocap) setup. The Vicon system is a commercially available indoor mocap system. The flight space in this work utilized 16 HD mocap cameras. Each free-flyer platform is marked with several tracking tags and individually registered in the system.

3) Inter-craft Communication: Each system uses a specialized cFS application, namely the Software Bus Network (SBN), for communication. SBN enables each instance of cFS to receive messages published to the software bus by any member of the swarm. For example, if telemetry received from one free-flyer indicates its position is too close another, modification can be made to flight paths to avoid potential collisions while still progressing to its next waypoint.

C. Inspection Protocol Using Computer Vision

The inspection protocol uses computer vision (CV) techniques on video streams provided by each free-flyer to identify potential damage or anomalies. There are two subtasks for the computer vision protocol: first, the object-of-interest (the mock satellite) is segmented from the background; second, damage is detected within a windowed area that corresponds to the segmented satellite. The 2D algorithm isolates regions of interest for autonomous operations and damage is highlighted and visualized for the operation team in real-time. As a complementary feature to 2D damage detection, 3D reconstruction for visualization is also implemented for human-in-the-loop post-mission analysis. The CV has been integrated into the flight software system as a cFS application.

IV. Implementation

New ICAROUS modules and cFS applications are used for swarm coordination and control. In particular, the following modules were developed: a mission coordinator for decentralized task distribution, custom flight planner for multiple agents, networking module that enables free-flyers to share flight plans and mission objectives, visual damage inspector, Vicon Tracker interface for providing vehicle telemetry to enable the damage detection and analysis, and custom flight controller based on a Proportional-integral-differential (PID) architecture to achieve the demonstration mission objectives. Several libraries are also created to autogenerate nominal flight plans for optimized video stability and field of view.

A. UAV Hardware Platform

In this demonstration mission, free-flyers are simulated with the Parrot™ AR 2.0 Drone equipped with an ARM™ Cortex A8 processor, 1Gb of RAM, and a barebones version of Linux 2.6 [26]. These platforms come equipped with a built-in WIFI b/g/n chip for both establishing and connecting to wireless networks. The sensors onboard each platform included a 3-axis gyro, a 3-axis accelerometer, magnetometer, ultrasonic sensor (for altitude measurements), and two cameras. A 720p 30 FPS camera faces forward on the UAV and is used to collect video for the damage analysis in this demonstration. The other camera is a downward facing wide angle lens sensor. This camera is used for optical flow tracking, which allows for smoother movement and hover.

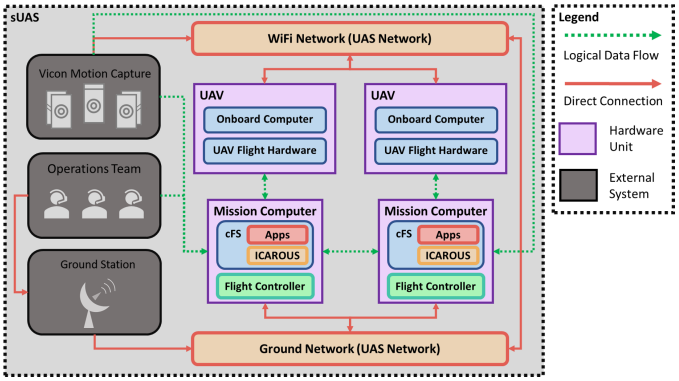


Fig. 3. Hardware Architecture for Simulating Free-Flyer Swarms

Due to a limit of 100g payload and insufficient computing power, the secondary mission computer payload communicates

to the onboard computer (OBC) remotely, as depicted in Figure 3. The Intel™ NUC miniature PC is chosen due to its small form factor. Each NUC is connected to one OBC via the UAV network. Each mission computer runs cFS and ICAROUS and issues low-level actuation commands. The live video from the forward-facing camera is streamed to the NUCs, where the cFS CV application would analyze the video for damage patterns.

B. Drone Control Software

The Flight Control module provides two high-level functions: convert velocity commands from cFS to low-level commands for the AR 2.0 Drone and serve as a flight controller to maintain trajectories with minimal error. The Parrot AR 2.0 Drone Software Development Kit (SDK) provides a standard API to support for takeoff and land, hover in place, activate emergency mode, and modify the roll, pitch, yaw, and gaz (vertical thrust). Several control systems are implemented for precise movement control, seen in Figure 4. A PID controller is used for managing 2D grid-based navigation, bang-bang controller for altitude, double setpoint controller for yaw and field of view, and normalized proportional controller for ground speed.

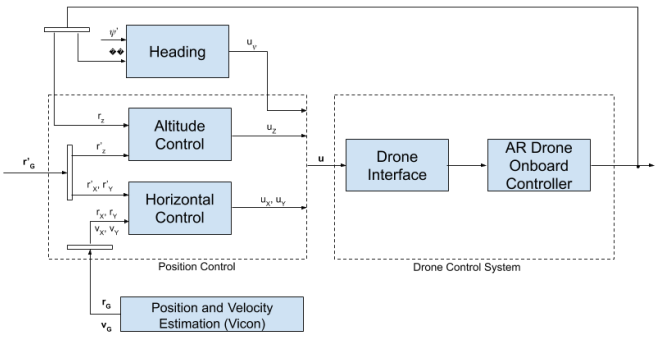


Fig. 4. High-level Control Overview

The resulting velocity output matrices were multiplied with three sets of transformation matrices, to convert the values from the local frame to the global frame of reference. Several experiments were run to tune these controllers and determine their effectiveness in comparison to off-the-shelf solutions. The equations used for each controller are listed below. For the UAV flight controller, Equation 1 is used to determine a desired viewing angle for the object-of-interest. X and Y are the Cartesian coordinates for the UAV and object-of-interest, in the local frame.

$$\theta = arctan2\frac{Y_{drone} - Y_{object}}{X_{drone} - X_{object}} \tag{1}$$

Equation 2 is used to calculate the yaw velocity of the drone to change its camera orientation. Here, ψ is the current UAV heading, θ is the desired heading, and max is the maximum UAV angular velocity.

$$V_{YAW} = \begin{cases} 0.25\omega_{max}, & \theta - \psi > 5^\circ \\ 0.75\omega_{max}, & \theta - \psi > 15^\circ \\ 0.75\omega_{max}, & \theta - \psi < 15^\circ \\ 0.25\omega_{max}, & \theta - \psi < 5^\circ \end{cases} \tag{2}$$

Equation 3 is used to calculate the thrust needed to change the UAV's current altitude. Velocity input for UAV thrust, where Δ_{Alt} is the required change in altitude, V_x is the current velocity in the X direction, τ is the yaw scaling factor, and V_{Alt} is the vertical velocity required to stabilize the UAV.

$$V_{GAZ} = \begin{cases} V_x * \tau + V_{ALT} & \Delta_{ALT} > 0.5m \\ V_x * \tau + V_{ALT} & -\Delta_{ALT} > 0.2m \\ 0.1m/s & otherwise \end{cases} \tag{3}$$

C. Multi-Agent Coordination

In the most basic implementation of an inspection, the ground station uploads a single flight plan to one free-flyer, which then travels to each waypoint. The mission concludes once all points have been reached. The Cognition and Guidance applications guide the free-flyer effectively and safely. These applications rely on receiving an initial flight plan to carry out a mission. This becomes further complicated as more than one free-flyer is introduced into the system. The complexity increases with dynamic swarm sizes. The Coordination application monitors the swarm and dynamically allocates and distributes mission plans according to swarm size and remaining waypoints from the flight plan that is uploaded initially to each single free-flyer. Coordination evaluates, computes, and distributes the mission tasks for its own free-flyer and all other swarm members based on their spacecraft ID. If a free-flyer is added to or removed from the system, a reassessment of remaining waypoints and free-flyer positions occurs, remaining tasks are redistributed. Coordination can handle several scenarios, including:

1) Mission starts with one or more available agents
2) A new agent is added to the available group of platforms
3) An existing agent is no longer able to perform a mission (loss of platform, communication, etc.)
4) A discrepancy in data is detected and new mission tasks need to be added for robustness
5) The object undergoing inspection has moved and new mission waypoints need to be determined

D. Automated Waypoint Generation

The goal of this mission is to use free-flyers to inspect a spacecraft for damage using a computer vision approach. To acquire sufficient visual data and ensure the detection of all simulated damage, several tools are created that auto generate various flight plans to obtain images at various angles and distances. The tool requires a number of parameters, including

the satellite's size and GPS coordinates, number of images desired, resolution of the images, desired yaw and pitch angles of the photos, and any unique flight patterns (helix, circle, raster photos, etc.). The planner first calculates all requested waypoints in a 3D cartesian coordinate system, placing the satellite at the origin. Then, these coordinates are converted into geodesic coordinates using an open source UTM library, which simulates the projection of 3D space onto a sphere (the Earth). Lastly, these new coordinates are formatted and combined into a mission input file.

E. Localization System

The Vicon motion capture is integrated as a cFS application using the Vicon SDK. The application opens a socket connection to received telemetry as the free-flyers' motions are tracked in real-time. Capturing 3D frames of the flight space at up to 200Hz, the free-flyers' position and rotation are recorded relative to the global center. The Vicon application performs several calculations to derive velocity (taking the difference in position between frames) and heading. The Vicon application also translates the coordinates from the local frame (North-East-Down) to the spoofed global frame (geodesic). Geodesic position data are piped to the Guidance application where, based on the current location and the assigned destination, adjustments are made to the velocities in the local coordinate frame to keep on course. These adjustments are then passed to the Firmware Interface to be translated to the raw commands accepted by the firmware.

F. Computer Vision Implementation

Figure 5 shows the 2D damage detection pipeline, beginning with reading the live-stream video feed into image segmentation with color ratios.

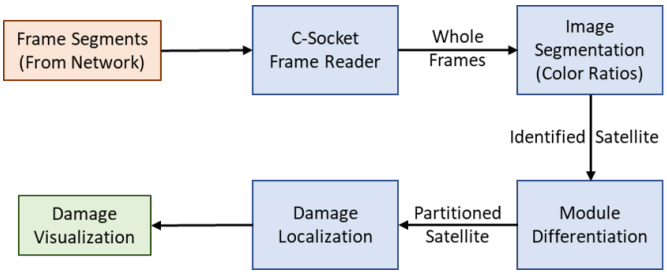
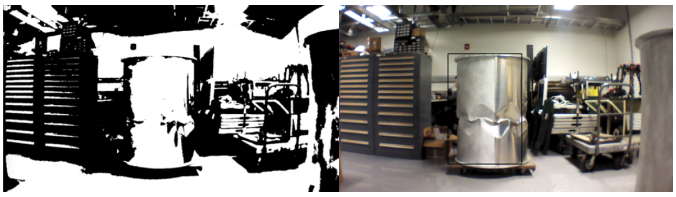


Fig. 5. 2D Online Damage Detection and Visualization

For background subtraction, adaptive Otsu thresholding is considered, shown in Figure 6a. Otsu is fast, but imprecise because intensity between the cylinders and the background is insufficient. Template matching is considered, shown in Figure 6b. Template matching works when the mock satellite is at a fixed distance (or image size). However, when the mock satellite is too close or too far, the predefined template will not match. This can be remedied by performing multiple searches with different-sized templates. Speed is drastically reduced with multiple template searches. Color-based thresholding is used to extract the location of the spacecraft module. The mock satellite is a gold color. Gold was chosen because of its similarity to that of the polyimide-based insulation usually found on the outside of satellites.



(a) Otsu Thresholding (b) Template Matching

Fig. 6. Initial Segmentation Experiments

The color based segmentation is based on ratio matching. Standard color matching is based on color channel vector distance as in equation 4. R refers to the red color channel intensity, and so on for G and B. R_{ref} refers to the target channel intensity and so on.

$$\sqrt{(R-R_{ref})^2+(G-G_{ref})^2+(B-B_{ref})^2} \tag{4}$$

This gives stable results invariant to lighting and background noise using ratio based matching, where distance is now described in equation 5. Reference RG_{ref} refers to a target red to green ratio, and likewise for green to blue ratio and blue to red ratio.

$$\sqrt{(\frac{R-G}{RG_{ref}})^2+(\frac{G-B}{GB_{ref}})^2+(\frac{B-R}{BR_{ref}})^2} \tag{5}$$

Once the cylinder image is segmented, CNN and Sobel edge detection are compared experimentally. The first CNN model is trained for a 3-class decision problem of "background", "damage", and "no damage". Including "background" made the classification problem more complex. Next, the CNN is trained to return a binary decision, representing damage "present" or "absent". This required building a training set of many example views of normal and damaged cylinders, with a variety of lighting and distance conditions. The CNN is applied as a sliding window operation to detect damage in regions of interest.

Sobel edge detection returns a filtered image, where edges are highlighted. Two damage detection algorithms are applied to the Sobel output. The Sobel output is integrated over windows of interest to determine regions of damage. The Sobel output is also visualized at the granularity of pixel level. For post-mission processing, 3D damage visualization using photogrammetry is performed. Figure 7 shows the 3D virtual reconstruction pipeline.

A necessary condition for fidelity in 3D object rendering is the presence of discriminatory image features at each location of the object. Unlike the expectations for a spacecraft, the clean surfaces of the gold painted cylinders are feature poor. To correct this deficiency, a speckle paint pattern is applied to the surface of the damaged cylinder. The photogrammetry pipeline uses an algorithm called "Structure from Motion" to

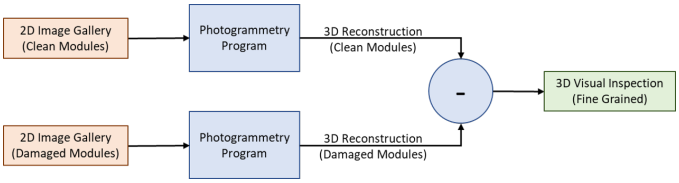


Fig. 7. 3D Offline Reconstruction

generate the 3D representation giving a gallery of images from different perspectives [25].

V. Experimentation & Results

For this demonstration, a damaged aluminum cylinder is used as a mockup of a damaged satellite. Multiple free-flyers use ICAROUS to autonomously navigate around a satellite while keeping it within their cameras' field of view for detecting damage at a high resolution. The free-flyers cooperate and maintain a safe distance between each other vehicle and the satellite while capturing damage at a high resolution. The satellite's location is represented by GPS geofencing. The Coordination module creates a unique flight plan for each free-flyer based on the shared mission plan. Coordination also allows dynamic task reallocation when the number of free-flyers available for the mission changes (though addition or loss). The free-flyers complete a full successful scan of the spacecraft, highlighting the damaged surfaces in real-time and providing a video visualization during mission execution. A publicly released video of the demo and the project overview is available at [27].

A. Full System Demo

The mission success demonstrates that multiple agents can cooperatively inspect an object for damage in real-time. Two UAVs are used to simulate two free-flyers in space. The mock satellite sections were stacked together, with the damaged cylinder on top of the pristine cylinder, to mimic the large cylindrical body of a fuselage. A single flight plan was autogenerated for inspecting the cylinders, with three distinct parts: An orbit of the top cylinder, a downward spiral in the pattern of a helix with two full revolutions, and a full orbit of the bottom cylinder. With this flight plan, each portion of the object's surface would be viewed at least twice by one of the cameras. In this demonstration mission, one free-flyer initiates the mission and a second free-flyer joins halfway through the mission. Three-quarters through the orbit, the second free-flyer is abruptly removed from the space, simulating a loss of an agent. The Coordination application detects the loss and dynamically reallocates the remaining tasks.

The demo progressed with the following events:

1) Free-flyer 1 enters flight space
2) Coordination module detects one freeflyer in swarm and distributes full flight plan from ground station
3) Free-flyer 1 begins orbit of top half of mock satellite, in accordance with flight plan, and processes video stream for real-time damage analysis
4) Free-flyer 2 is dispatched
5) Coordination module detects new free-flyer in swarm and dynamically updates mission into two subtask lists: one for orbiting top-half, one for orbiting bottom-half
6) Subtask lists are distributed between Free-flyer 1 and 2
7) Free-flyer 1 and 2 begin executing mission subtasks
8) Free-flyer 2 abruptly departs
9) Coordination module detects Free-flyer 2 loss
10) Coordination module calculates new mission task list for Free-flyer 1 with remaining waypoints
11) Free-flyer 1 completes visual analysis of top-half
12) Free-flyer 1 enters downward helix pattern in order to complete analysis of bottom-half using waypoints inherited after loss of Free-flyer 2
13) Free-flyer 1 completes remainder of mission

Figure 8 shows the original mission flight plan path and represents the independent flight patterns for both free-flyers. Each track point is generated from the flight log of their respective free-flyer from this demonstration mission.

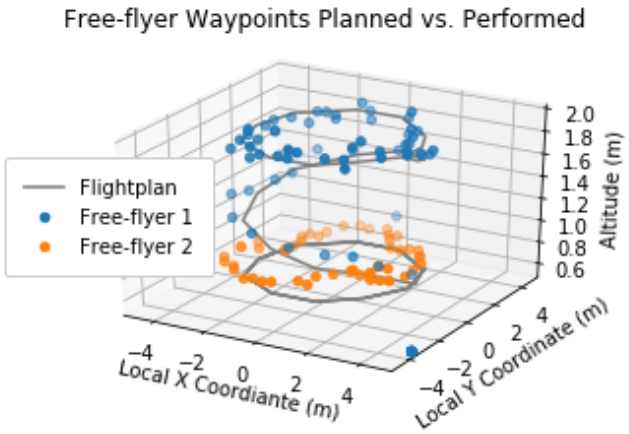
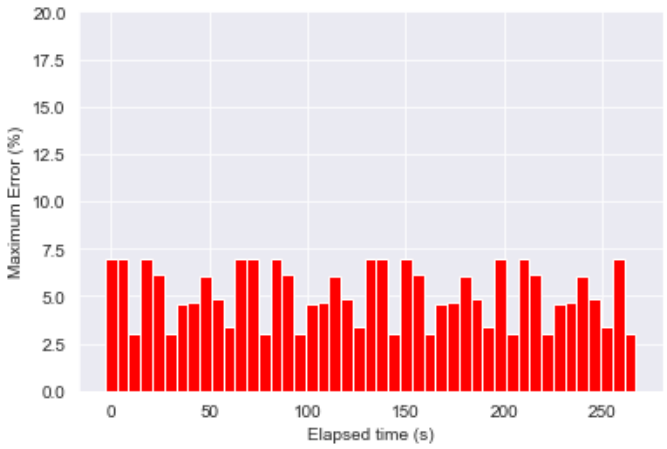


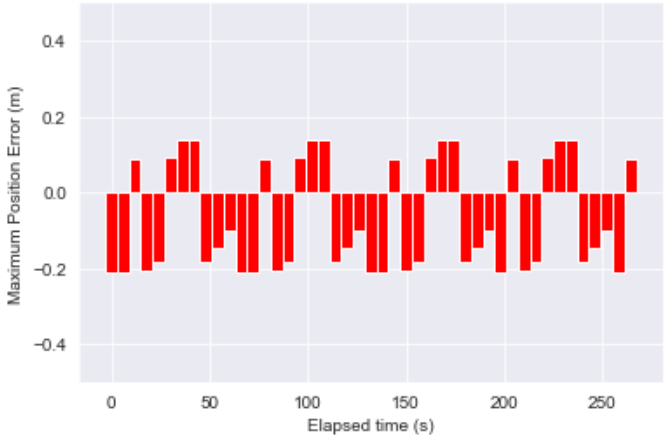
Fig. 8. Free-flyer Waypoint Visualization

B. Flight Controller Performance

Several experiments are performed during the development of the UAV control software to assess the impact of the different control approaches and tuning methods. Figure 9 shows the maximum error of the drone's position across five trials. The PID controller for the 2D trajectory provides the AR 2.0 Drone acceptably precise movement. During flight testing, each UAV's actual position is recorded and compared to the intended position determined by ICAROUS. Figure 9a shows the error in terms of absolute distance (meters). and Figure 9b shows the error in terms of relative distance (percentage). For the yaw control, both a proportional controller and a double setpoint controller were considered. Tradeoffs between these controllers are discussed in Section VI-A. For the altitude controller, a simple bang-bang controller is used for thrust control. The system reaches the targeted altitude within 1.5 seconds, based on the ground speed of the platform, while maintaining smooth motion.



(a) UAV Position: Absolute Error Over Time



(b) UAV Position: Relative Error Over Time

Fig. 9. UAV Position Error

C. Computer Vision Results

The CNN is trained for damage detection on multiple surfaces. In comparison to Sobel filtering, the CNN presents as over-engineered for a simple spacecraft surface. Figure 10 shows the output of the CNN on a sliding window damage detection where green represents normal and red represents damage.

Based on performance, the final system incorporated Sobel filtering for damage detection. Given that the surfaces are rather smooth and have low inter class variance, the CNN would yield false negatives, where the Sobel filter would not. False negatives qualify as mission failure for detecting damage. The final system incorporates a tunable Sobel filter threshold which enables the false negatives to be minimized by increasing false positives. This parameter is fine-tuned and found that the damage can be robustly isolated with very low false negatives, and at the same time the false positives are very low. The color ratio segmentation algorithm performs satellite segmentation, shown below in Figure 11 with the blue and green windows. The overlaid final damaged detection is shown using window-based damage detection in the left and edge-based highlighting in the right. The final segmentation and damage detection algorithms are written in OpenCV, which requires C++. However, it is wrapped into a C application, so the CV algorithms are self-contained in an ICAROUS module.

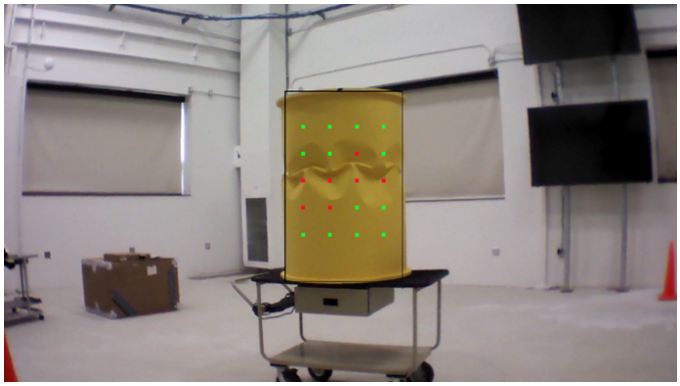


Fig. 10. CNN Sliding Window Output

Fig. 11. CV Output of Segmented Satellites with Damage Highlighting

Additionally, 3D modeling is demonstrated to supplement and improve perception of damage to the spacecraft. Three-dimensional visualization shows a human-navigable depth perspective for further investigation. Given a spacecraft surface with arbitrary lines (features that the Sobel would pick up but do not represent damage), the original surface can be modeled for comparison with an after-damage 3D reconstruction to detect discrepancies due to damage. Figure 12 shows where the speckle paint is applied to the cylinder, and 3D-reconstruction.

VI. Discussion

The objective of this work is to demonstrate the utility of free-flyers to future space missions. Using simple, cost-effective UAVs as free-flyer substitutes, a software framework for onboard computer vision for in-situ inspections of objects-of-interest demonstrates the feasibility of such a system. The mission and experiments carried out have shown that the ICAROUS framework allows for simulating free-flyers with UAVs platforms and may be a suitable candidate for space-flight missions. The following is a discussion on the outcomes, implementation results, and challenges for each major part of this project.

A. Individual UAV Control

The AR 2.0 Drone is adequate for the requirements of this mission simulator. The SDK enables precision control where the off-the-shelf control is insufficient. The PID control allows

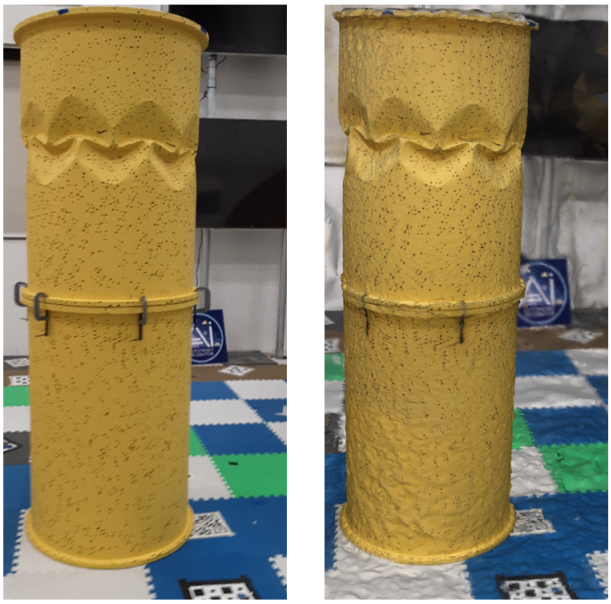


Fig. 12. Speckled Painting (left) and 3D Reconstruction (right)

the UAV to follow the directed velocity commands and improve power efficiency by avoiding unnecessary acceleration and jerk. The original design of the yaw controller tended to overdampen, causing the UAV to jerk back and forth while focusing on the cylinder. This caused noise in the video data, which inhibited the damage analysis. The double setpoint control scheme for the yaw allows for smooth control of the UAV's angular velocity, and the small margin of error allowed the video to stabilize sufficiently for video data analysis. The upgraded control system improved stability while hovering. The simple bang-bang controller for movement in the Z-axis proved sufficient, as the system would always have to provide a non-zero thrust to ensure stability. This allowed the UAV to reach the desired altitude quickly, while using the UAV's current directional momentum and improving gaz. It was shown that this scaled approach to altitude hold allowed for the smoothest transition between altitudes, during UAV movement and position hold.

B. Localization System

The indoor localization system successfully integrates into ICAROUS, allowing for pseudo-GPS data to be provided for any of the tracked objects within the flight space's vicinity. The GPS location was dynamic, allowing for simulated testing in numerous real-world locations. The use of the indoor localization system allows for a high degree of accuracy with respect to tracking the UAVs' movement. This enables the cooperation of multiple platforms in a highly confined space. The Vicon system is designed to provide localization data for over one hundred unique agents. Each agent's position and orientation data are provided at a rate of 200Hz. This rate allows for the tracking of vehicle acceleration and jerk at an interval measured in milliseconds. This is comparable to the tracking necessary of any free-flyers in orbit, where the distance traveled in this short time frame is significant.

C. Multi-Agent Implementation

The main challenge of using multiple agents is determining how to distribute the mission tasks between available free-flyers and coordinate their motion planning. This is challenging as the available free-flyers are treated as a dynamic resource, i.e., free-flyers may be added or removed from service at any time. In order to ensure all mission waypoints and tasks are achieved, the Coordination application tracks all assigned subtasks, and verified task completion with each agent at every milestone. At the beginning of each time step, the free-flyers check to see if any other agents were added to or removed from the swarm. Only upon additional or loss would the Coordination module re-distribute mission tasks. This method of free-flyer coordination and mission allocation proves successful in all the testing scenarios. In addition to tests with variable number of free-flyers, some tests are run where platforms are suddenly removed from the system (to simulate sudden damage or system failure). Each agent's data are marked with a timestamp including the platform's spacecraft ID. This way, during 3D reconstruction, the CV module would be able to use each free-flyer's position/velocity data to match with relevant video data in relative space. One challenge of joint motion planning is preventing any well-clear violations (making sure the free-flyers do not fly too close to each other or the object). This is accomplished by having each free-flyer broadcast its current and next waypoint throughout the mission; a free-flyer's path planner checks bands to make sure there is no intersection or adjacency of paths during movement. Another major challenge is verifying that each waypoint is visited the correct number of times by the swarm. To solve this, each platform kept track of the list of waypoints all the other free-flyers visited, based on the previously mentioned broadcasts; if there was a discrepancy, the affected waypoints would be visited again to verify data redundancy.

D. Computer Vision System

The CNN-based algorithm is trained as demonstrated for damage detection. This is less preferred for a simple spacecraft surface than Sobel. However, if the damage is more diverse in character, and the spacecraft has a complicated surface with equipment and junctions, a CNN can be trained, given an existing dataset of labeled damage. The final system incorporated Sobel filtering for damage detection. Advantageously, Sobel filters may be tuned to minimize false negatives.

Adding 3D modeling offers two benefits: first, comparative before and after models to detect damage; second, 3D models for visualizing damage offline in an interactive virtual environment. Using 3D modeling, human-in-the-loop operators may employ damage detection techniques that are not feasible on a free-flyer's OBC.

VII. Future Work

For future work the new ICAROUS modules can be improved to allow for more complex methods of mission allocation and coordination. As part of this project, preliminary tests with distributed communication systems were conducted, e.g., Data Distribution Service (DDS) incorporated into the default cFS software communication bus [19]. Using DDS in a multi-agent system would allow for other advanced networking and data sharing techniques when coordinating a large number of systems. In computer vision another step would be to implement 3D reconstruction capability in real time, as the free-flyers scan its surface.

VIII. Conclusions

This work implements and demonstrates the feasibility of computer vision and navigation techniques to perform inspection of a large spacecraft using the ICAROUS framework. In this demonstration mission, UAVs were successfully used to simulate free-flyer spacecraft. Moreover, localized damage was successfully detected on a uniform metallic surface. The multi-agent coordination approach proved capable of supporting a dynamic number of agents, allowing for efficient mission planning and completion based on a variable number of resources. The results of this work suggest free-flyers are a viable platform for in-situ, real-time damage detection of spacecraft structures.

Acknowledgment

The authors of this paper would like to thank the Safety-Critical Avionics Branch and the Autonomy Incubator at NASA Langley Research Center as well as faculty and students of Old Dominion University and the NSF SHREC Center at the University of Pittsburgh (NSF SHREC Center and IUCRC Program of the NSF under Grant No. CNS-1738783), for their continued support, resources, and insights provided throughout the duration of this research project.

References

[1] N. L. Johnson, "Orbital debris: the growing threat to space operations," in *33rd Annual Guidance and Control Conference*, 2010.

[2] D. Doyle, A. Zagrai, B. Arritt, and H. Çakan, "Damage detection in bolted space structures," *Journal of Intelligent Material Systems and Structures*, vol. 21, no. 3, pp. 251–264, 2010.

[3] A. Zagrai, D. Doyle, and B. Arritt, "Embedded nonlinear ultrasonics for structural health monitoring of satellite joints," in *Health Monitoring of Structural and Biological Systems 2008*, vol. 6935. International Society for Optics and Photonics, 2008, p. 693505.

[4] S. Balachandran, C. A. Munoz, M. C. Consiglio, M. A. Feliú, and A. V. Patel, "Independent configurable architecture for reliable operation of unmanned systems with distributed onboard services," in *2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*. IEEE, 2018, pp. 1–6.

[5] D. McComas, "Nasa/gsfc's flight software core flight system," 2012.

[6] J. Fernandez Galarreta, N. Kerle, and M. Gerke, "Uav-based urban structural damage assessment using object-based image analysis and semantic reasoning." *Natural Hazards & Earth System Sciences*, vol. 15, no. 6, 2015.

[7] G. Morgenthal and N. Hallermann, "Quality assessment of unmanned aerial vehicle (uav) based visual inspection of structures," *Advances in Structural Engineering*, vol. 17, no. 3, pp. 289–302, 2014.

[8] H. Sui, J. Tu, Z. Song, G. Chen, and Q. Li, "A novel 3d building damage detection method using multiple overlapping uav images," *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 40, no. 7, p. 173, 2014.

[9] Q. Guo, Y. Su, T. Hu, X. Zhao, F. Wu, Y. Li, J. Liu, L. Chen, G. Xu, G. Lin *et al.*, "An integrated uav-borne lidar system for 3d habitat mapping in three forest ecosystems across china," *International journal of remote sensing*, vol. 38, no. 8-10, pp. 2954–2972, 2017.

[10] Z. Shang and Z. Shen, "Real-time 3d reconstruction on construction site using visual slam and uav," *arXiv preprint arXiv:1712.07122*, 2017.

[11] Y. Tian, K. Liu, K. Ok, L. Tran, D. Allen, N. Roy, and J. P. How, "Search and rescue under the forest canopy using multiple uas," in *International Symposium on Experimental Robotics*. Springer, 2018, pp. 140–152.

[12] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *International workshop on swarm robotics*. Springer, 2004, pp. 10–20.

[13] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for swarm robots, intelligent robots and systems' 93, iros'93," in *Proceedings of the 1993 IEEE/RSJ International Conference on*, vol. 1, 1993.

[14] Y. U. Cao, A. S. Fukunaga, and A. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous robots*, vol. 4, no. 1, pp. 7–27, 1997.

[15] P. Vincent and I. Rubin, "A framework and analysis for cooperative search using uav swarms," in *Proceedings of the 2004 ACM symposium on Applied computing*, 2004, pp. 79–86.

[16] K. Lerman, A. Martinoli, and A. Galstyan, "A review of probabilistic macroscopic models for swarm robotic systems," in *International workshop on swarm robotics*. Springer, 2004, pp. 143–152.

[17] S. Balachandran, A. Narkawicz, C. Muñoz, and M. Consiglio, "A path planning algorithm to enable well-clear low altitude uas operation beyond visual line of sight," in *Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM2017)*, 2017.

[18] M. Consiglio, B. J. Duffy, S. Balachandran, L. Glaab, and C. Munoz, "Sense and avoid characterization of the independent configurable architecture for reliable operations of unmanned systems," in *13th USA/Europe Air Traffic Management Research and Development Seminar*, 2019.

[19] S. Balachandran, C. Manderino, C. A. Munoz, and M. C. Consiglio, "A decentralized framework to support uas merging and spacing operations in urban canyons," in *2020 IEEE/AIAA 39th Digital Avionics Systems Conference (DASC)*. IEEE, 2020.

[20] H. J. Vala and A. Baxi, "A review on otsu image segmentation algorithm," *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, vol. 2, no. 2, pp. 387–389, 2013.

[21] R. Brunelli, *Template matching techniques in computer vision: theory and practice*. John Wiley & Sons, 2009.

[22] F. Garcia-Lamont, J. Cervantes, A. López, and L. Rodriguez, "Segmentation of images by color features: A survey," *Neurocomputing*, vol. 292, pp. 1–27, 2018.

[23] S. Gupta and S. G. Mazumdar, "Sobel edge detection algorithm," *International journal of computer science and management Research*, vol. 2, no. 2, pp. 1578–1583, 2013.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[25] I. Reljić, I. Dunder, and S. Seljan, "Photogrammetric 3d scanning of physical objects: Tools and workflow," *TEM Journal*, vol. 8, no. 2, p. 383, 2019.

[26] Parrot. Parrot ar.drone 2.0 power edition. [Online]. Available: https://www.parrot.com/global/drones/parrot-ardrone-20-power-edition

[27] A. Incubator. Free flyers: Autonomous coordinated operations. [Online]. Available: http://autonomyincubator.blogspot.com/2019/08/2019-08-09-free-flyers-autonomous.html