SSC21-I-05

Sagittarius A* Small Satellite Mission: Capabilities and Commissioning Preview

Caleb Royer, Mike Renner, Owen Brown, Robert Polutchko, Paul Bains
Scientific Systems Company, Inc.
500 West Cummings Park, Ste 3000, Woburn, MA 01801; (781) 933-5655 x371
caleb.royer@ssci.com

Sam Gagnard, Alex Herz
Orbit Logic
7852 Walker Dr Ste 400, Greenbelt, MD 20770; (301) 982-6232
sam.gagnard@orbitlogic.com

Mark Mason
Leaf Labs
288 Norfolk St, Cambridge, MA 02139; (903) 345-5323
mason@leaflabs.com

Nick Stanley
Emergent Space Technologies
7701 N. Lamar Blvd, Ste 510, Austin, TX 78752; (301) 345-1535
nick.stanley@emergentspace.com

John Eterno
Loft Orbital
810 Brickyard Cir, Unit 3, Golden, CO 80403; (303) 746-1551
john@loftorbital.com

ABSTRACT

SSCI is leading a Defense Advanced Research Projects Agency (DARPA)-funded team launching a mission in June 2021, dubbed Sagittarius A*, to demonstrate key hardware and software technologies for on-orbit autonomy, to provide a software testbed for on-orbit developmental test & autonomous mission operations, and to reduce risk for future constellation-level mission autonomy and operations. In this paper, we present the system CONOPs and capabilities, system architectures, flight and ground software development status, and initial commissioning status. The system will fly on Loft Orbital's YAM-3 shared LEO satellite mission, and includes SSCI's onboard autonomy software suite running on an Innoflight CFC-400 processor with onboard Automatic Target Recognition (ATR). The autonomy payload has attitude control authority over the spacecraft bus and command authority of the imaging payload, and performs fully-autonomous onboard request handling, resource & task allocation, collection execution, ATR, and detection downlinking. The system is capable of machine-to-machine tip-and-cue from offboard cueing sources via cloud-based integrations. Requests for mission data are submitted to the satellite throughout its orbit from a tactical user level via a smartphone application, and ISR data products are downlinked and displayed at the tactical level on an Android Tactical Assault Kit (ATAK) smartphone. Follow-on software updates can be sent to the autonomy suite as over-the-air updates for on-orbit testing at any time during the on-orbit life of the satellite. Communications include GlobalStar inter-satellite communications for low rate task and status monitoring, and ground station links for payload data downloads. Planned demonstrations and opportunities will be discussed.

MISSION & CONOPS

Mission Overview

SSCI is leading this team to demonstrate key mission autonomy software technologies and to provide a testbed for on-orbit software developmental test & autonomous mission operations, and to reduce risk for future constellation-level mission autonomy and operations. This mission, dubbed Sagittarius A*, will perform space-based detection based on offboard cues and onboard imaging & Automatic Target Recognition (ATR), with future extension to data fusion of other data sources onboard.

Loft Orbital has provided SSCI with a "Mission-as-a-Service" by accommodating an electro-optical (EO) imager, and an Innoflight CFC-400 mission processor hosting SSCI's mission autonomy software, on the YAM-3 spacecraft launching in June 2021. This combined system has a nominal mission duration of 1 year with options for follow-on operations throughout the 5-year spacecraft lifetime.

The onboard autonomy software has attitude control authority over the spacecraft bus and command authority of the imaging payload. The SSCI Collaborative Mission Autonomy (CMA) software architecture is a decentralized architecture for all stages of onboard Tasking, Collection, Processing, Exploitation, and Dissemination (TCPED) handling. It performs fully-autonomous onboard request handling, resource & task allocation, collection execution, ATR, and detection downlinking. The system will deliver a remote sensing capability, among other additional demonstrations & tests possible. Land and maritime object signals can be collected by 3rd-party space-based platforms, and used to cue the Sagittarius A* satellite for inspection, which is performed with the EO imager and by running Machine Learning (ML) based detection & classification on the collected imagery to confirm/disconfirm presence of objects of interest.[1] This onboard ATR provides a valuable source of data to fuse with offboard data and other phenomenologies such as Synthetic Aperture Radar (SAR), including potentially from commercial constellations.

The Sagittarius A* team is comprised of experts across industry in space systems, constellation management, and mission autonomy flight software.

Mission Objectives

The mission objectives are to demonstrate*:*

- Autonomous management of bus, payload, and processing/exploitation resources to satisfy a user's TCPED Mission Service Request
- Autonomy software execution & benchmarking on COTS, limited-SWAP, commodity onboard mission processors
- The ability to upload & update mission flight software (FSW) during on-orbit operations.
- Autonomous tipping and cueing using offboard ISR resources
- Execution and test of third-party "massless payloads", i.e. software plugins that enhance asset capabilities, which are developed using our Software Development Kit (SDK)

Mission Value & Impact

This mission provides an on-orbit demonstration of critical software technologies for Proliferated LEO autonomous mission Command, Control, and Communications (C3) and associated new CONOPS. The mission autonomy software flown on this single spacecraft is designed to operate across full satellite constellations, and Sagittarius A* demonstrates a single spacecraft (also termed a "node", as in a network) performing multiple TCPED chains simultaneously on its own, running software that can seamlessly extend to multiple nodes executing multi-asset cooperative TCPED operations. The risk reduction achieved by flight-testing this software is significant, and drives down risk for constellation-scale:

1) Responsive multi-mission self-tasking and resource optimization,
2) Processing and data product dissemination,
3) Flexibility for new mission capabilities uploaded on-orbit,
4) Multi-domain capabilities,
5) and resiliency via re-optimized resources to continue missions despite node failure or loss.

The Sagittarius A* mission also provides benefits of studying software and operations for EO imaging, onboard EO ML-based ATR, and future extension to on-orbit data fusion, tip-and-cue, and other valuable constellation management capabilities in a fully space-based edge-processing environment.

Practically speaking, this mission also establishes an on-orbit testbed for continuous, iterative, and evolving test of mission applications to enable experimentation of autonomous satellite and/or multi-domain systems relevant to many government & commercial entities.

Concept of Operations (CONOPs)

The mission CONOPs are shown in Figure 1. On-orbit, low-level spacecraft bus operations are controlled by Loft Orbital, and the SSCI team interacts with the system for command & telemetry via SSCI ground software connected to Loft Orbital's web-based Cockpit mission ground software.

During on-orbit demonstrations, semantic-level Mission Service Requests (MSRs) are submitted to the satellite. MSRs are submitted at any time from a smartphone Human Machine Interface (HMI) application, as well as by machine-to-machine automated tip-and-cue driven by radio frequency (RF) satellite data sources. These MSRs are transmitted directly to the satellite via Globalstar Inter-Satellite Link (ISL).

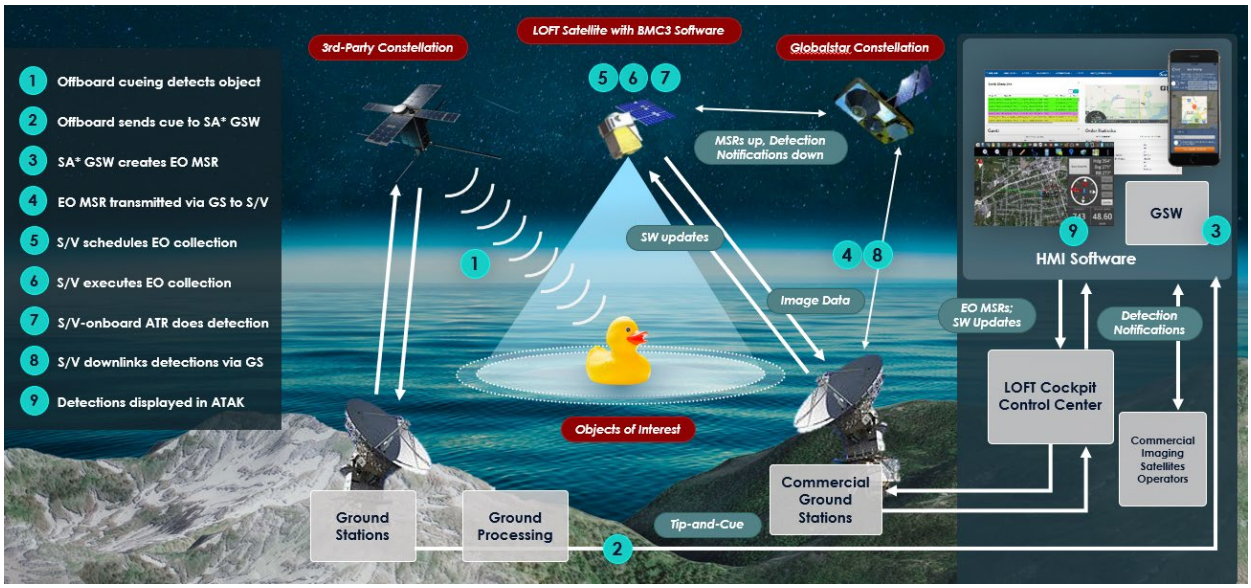

Figure 1: Mission Concept of Operations (CONOPs)

Based on MSRs from the ground system, the onboard autonomy plans & executes image collections, then the onboard machine learning object detection software detects & classifies objects in the imagery. Immediately, the products are downlinked to the user on the ground via Globalstar for display on an Android Tactical Assault Kit (ATAK) smartphone. Raw imagery, engineering data, & telemetry are downlinked during ground station contacts. Ground station contacts are used to uplink payload management commands and configuration data, as well as flight software updates.

In addition to traditional command & telemetry during ground passes, the system has the capability for "Direct Mode", which is effectively a live shell session with the mission processor. Ground scripts send commands to the CFC-400 onboard, where the response is packaged and returned through the communications system in live fashion during ground passes. This creates a highly innovative Read-Eval-Print Loop (REPL) environment for operators, in addition to traditional spacecraft command scheduling.

Loft Orbital Rideshare

In Loft Orbital's offering for this mission, the imager and CFC-400 payloads are flown on a flight-proven small satellite bus, and Loft Orbital manages the delivery of the payloads to orbit and supports operations. SSCI operators retain full Command and Control (C2) of the payloads on orbit via ground software integrations to the Loft Orbital ground system software, called Cockpit. This approach enabled rapid payload accommodation, interfacing, and operations, and provided a fast, simple, and low-cost path to orbit for the Sagittarius A* mission.

The planned spacecraft orbit is a 500-600km Sun Sync Orbit. Loft Orbital's spacecraft is configured as a rideshare, meaning that multiple customer payloads share the onboard resources of the 100-kg class microsatellite platform. The SSCI team & payloads have reserved 10% orbit average time for the imaging duty cycle and 25% orbit average time for the mission data processor. These reservations are not power, storage, or imager limited, and can be negotiated to perform demonstrations of longer periods, though longer operations will eventually reach duty cycle limits driven by power draws and sun tracking.

SYSTEM OVERVIEW & ARCHITECTURE

System Overview

The system is comprised of a LeoStella spacecraft bus, Simera EO imager, CFC-400, Mission Flight Software, Loft Orbital-provided interface hardware & software, and ground software.

Bus operations are controlled by Loft Orbital, and the SSCI team commands payload operations from SSCI ground software during scheduled operations windows. The SSCI team accesses telemetry and data generated by the payload elements through Loft Orbital's web-based Cockpit software.

This system provides a valuable testbed for fully autonomous TCPED operations, and for demonstrating

critical software technologies for autonomous PLEO Constellation Mission Management as well as multi-domain composable systems.

Flight Software Architecture

The Sagittarius A* mission Flight Software (FSW) is comprised of multiple microservices that interoperate to achieve mission goals, focused initially on object detection & software testbed capabilities. The flight software applications fall into one of two categories, which are detailed in subsequent sections. The first category is Constellation Mission Management Software for spacecraft & payload autonomous TCPED execution. The second category is Infrastructural Software for operating system (OS), file, and application management. All flight software runs on Linux onboard the CFC-400 mission processor.

The Infrastructural Software is capable of performing on-orbit flight software updates, including minor bug fixes & configuration updates, as well as uplinking entirely new software plug-ins, and even replacing the operating system. Any application conforming to the Software Development Kit (SDK) can be uploaded, connected to onboard message buses, & operated from ground. All software can be checked out for execution & compute resource usage in the Hardware-in-the-Loop (HITL) environment on the ground, then uplinked to the spacecraft for operations.

This capability & process allows building toward & following a highly innovative spacecraft-level Continuous Integration / Continuous Deployment (CI/CD)-type process, with multiple software loads & OS loads onboard the spacecraft. At all times there are at minimum three OS loads (Gold, Side-A, Side-B), and four mission application sets (Gold, Staging, Secondary, Primary), so that the system can always roll back software safely, but also experiment & push updates to applications running onboard. In addition, the system has been set up to execute Python-based applications, opening the door for executing future Python-based Machine Learning software onboard.

CONSTELLATION MISSION MANAGEMENT SOFTWARE

SSCI's legacy Collaborative Mission Autonomy (CMA) is a C3 software for handling large numbers of targets, large numbers of concurrent TCPED activities, & heterogenous or homogenous assets, all with resiliency and adaptability. This software performs constellation mission management via a hybrid deliberative/reactive architecture with intelligent agents implemented as microservices communicating over an onboard middleware message bus, and the architecture is shown in Figure 2. The software autonomously executes TCPED pipelines from MSR to mission data delivery. Key innovations of SSCI's constellation mission management software include:

1) A decentralized software that can be extended to utilize multiple satellites for autonomous, adaptive, resource planning and coordination based on high-level mission requests.

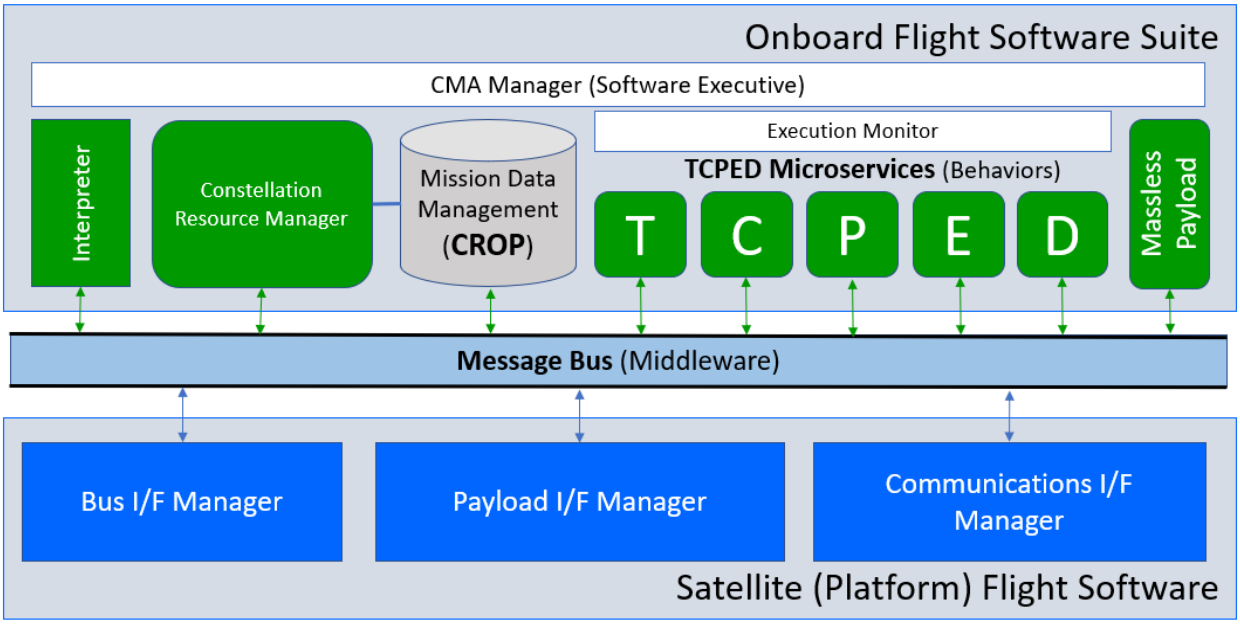

Figure 2: Constellation Mission Management Software Architecture

2) Platform-flexible algorithms, which enable execution on ground- and space-based processors. The Constellation management autonomy software is designed to be Bus-, Payload-, and Communications-agnostic, with specific interface managers to adapt the C3 applications to specific vendor hardware & software.

3) Payload-flexible algorithms, which can task, process, fuse/exploit, and route the data of multiple sensor phenomenologies & collection mechanisms

Interpreter

Semantic-level Mission Service Requests (MSRs) are sent to the satellite and decomposed onboard into Pipelines of Work Items (e.g. Task, Collect, Exploit, etc.) with the request Interpreter microservice. The Pipeline & Work Item concept invented & implemented by SSCI is very different from traditional collection planning systems where a massive, detailed task deck is submitted to a ground-based scheduler, with collections scheduled rigidly and days in advance. Instead, the Pipeline concept is dynamic, inherently handles task dependency trees, and Pipelines can spawn additional Pipelines autonomously such as when an ATR algorithm identifies an object of interest and cross-cues another sensing phenomenology or asset. Pipelines can also achieve periodic collections, instead of more cumbersome repetitive tasking decks days in advance.

CROP

Pipelines & Work Items are stored onboard in the Common Relevant Operating Picture (CROP) database along with their status and other supporting data. The pipelines are distributed among offboard sources (and future additional satellite nodes) via lightweight CROP synchronization & reconciliation algorithms to enable tip-and-cue, distributed processing, and decentralized planning and execution. CROP synchronization is the mechanism for future constellation coordination. The CROP is persisted on disk using high speed key-value storage.

The Pipelines & Work Items have states which are updated over time as they are completed, all of which can be synchronized to the ground. The Work Item states include: Unplanned, Planned, InProgress, Complete, Failed, and Canceled. Work Items completions constitute the stages of satisfying an MSR and generating the required mission data for the user, and all of these Work Items are scheduled for execution via the onboard planners based on available resources.

Planning

The Autonomous Planning System (APS) is the onboard planning software, and rapidly and autonomously allocates & schedules resources such as the sensor (and other satellites & offboard resources), given availability and resource constraints.[2] The APS software can handle thousands of requests executing diverse TCPED tasks, and can accommodate data quality of various sensors in the optimization reward functions. As soon as APS generates a local plan, the plan is placed in the CROP for distribution to other nodes. Those other nodes (if available) then execute planning to account for booked tasks so as not to plan lower scoring or redundant task execution (unless desired). If multiple assets or resources are available (unlike the single-string Sagittarius A* mission), plans are built to generate a deconflicted solution within constraints and resource limits.

The APS software implements a hierarchical planning methodology, with Specialized Autonomous Planning Agents (SAPA) aggregated by a Master Autonomous Planning Agent (MAPA) in a highly extensible & powerful planning framework. The APS software monitors the CROP for new Work Items, and uses the latest CROP data at each planning cycle. Any valid target imaging access opportunities are scored using an approach which combines user priority, mission priority, region priorities, and imaging opportunity quality scoring.[2] Once scored, the MAPA then generates the spacecraft activity plan based on inputs from all SAPAs and available resources within a configurable time window. The planning cycles are configurable, but currently re-plans every 1 second (if new Work Items are added) with a lookahead window of 5 minutes in a receding horizon fashion.

This decentralized CROP-based planning system outperforms typical non-coordinated (greedy) approaches and consensus-based methods, and achieves similar optimality to centralized planning systems, but with the benefit of resilience to node failure/loss, rapid reaction to changing state, and scalability to indefinitely large numbers of participating assets.

The APS resource planner microservice of the BMC3 system optimizes the sensor usage and converts mission activities to high-level bus & payload commands, which are sent to the DASHER timeline executive for timed executive of specific low-level commands.

Execution

The DASHER timeline manager is an executive for automating on-board satellite operations.[3] DASHER receives & stores tasks to be forwarded to the Bus or a Payload at a specific time, such as collections and

related pointing commands. A task to be executed by DASHER consists of timing, identification, resource members and associated data. DASHER can accommodate command inputs from a variety of sources including the ground operations team or advanced on-board planning applications such as APS, and can be extended with addition of temporal constraints and vehicle health constraints to improve system safety & robustness to malfunctions.

The Bus & Payload Manager components are bridges between the mission autonomy applications and the specific Bus (LeoStella) and Payload (Simera Imager). These managers subscribe to DASHER published tasks & commands, forward them to other subsystems as necessary, and report status information to the DASHER executive. These components also publish telemetry and state of the connected subsystems. An additional component, Telemetry Manager, is responsible for aggregating logs & telemetry, and recording traffic off the onboard message bus.

Processing & Exploitation

For this mission demonstration, the Payload Manager automatically receives imagery from the EO Imagery, formats it as GeoTIFF, and passes it to the Exploitation microservice which executes the Machine Learning (ML)-based ATR for object detection, with reports downlinked immediately over the GlobalStar link.

The ML-based ATR (also called "massless payload") entails a detector based on a low-SWAP object detection Convolutional Neural Network (CNN). The detector & classifier are state-of-the-art CNN models specialized for overhead object detection, and detect ships, aircraft, and dozens of other classes of vehicles. The model was trained using commercial imagery, with a variety of GSDs. The CNN architecture chosen is smaller and shallower than other Deep Learning models, making this ATR more performant on low-SWaP, low-memory devices like the CFC-400. The algorithm can be scaled to operate on devices with limited memory by decreasing the sliding window size, at the expense of longer processing time. The ATR returns a success code and array of detections, sorted by maximum class probability, in addition to a position-velocity tracklet for detections.

INFRASTRUCTURAL FLIGHT SOFTWARE

Software Development Kit & Middleware

As mentioned previously, the Constellation Mission Management Software is implemented as a set of microservices communicating via an onboard middleware message bus. This middleware is a key piece of the infrastructural software, and also includes a Software Development Kit (SDK), which is an innovation that allowed the 13 onboard mission applications to be rapidly developed & interfaced. Additionally, any future 3rd-parties can develop additional applications against this SDK, and the new applications uploaded to the mission processor for full interoperability with the satellite & mission autonomy software.

In selecting the SDK & middleware, several industry middlewares were assessed in a trade study against 11 criteria including: development effort required, features & limitations, developer community, maintainability, available schemas, code generation capability, and compute performance. The selected middleware was chosen for the specific needs of the mission and driven by the compressed timeline for developing the Sagittarius A* flight software. The selection was the ASPIRE middleware, an AFRL-funded software, that has proven to be lean & performant in both size and compute needs.[4] Across the team, this middleware & associated SDK took only a few hours to install, build, create projects, and modify with additional messages.

The middleware provides messaging between applications, message auto-generation from XML schemas, and software patterns to achieve plug-and-play interoperability on the message bus. It supports C++ and Python applications and multiple messaging paradigms, though most message interactions follow publish/subscribe or request-reply patterns.

Mission Executive

Given the high number of onboard mission applications running on the CFC-400 (at least 13 microservices at launch), there is a need for a top-level executive application. The Mission Executive manages the mission-critical, application-level software on the mission processor. It is responsible for starting, stopping, and monitoring the mission applications, and enables the software update configuration control via configuration settings that specify the multiple OS & mission software application installations. The Mission Executive interfaces with the messaging middleware, but is independent, as it is responsible for starting and managing the middleware.

The Mission Executive process always runs at boot of the system, and upon startup or user directive to restart, it reads the configuration information for critical parameters, including the list of applications to run & monitor. The Mission Executive performs important Fault Detection & Correction (FDC) behavior, including application monitoring via heartbeat messages, and response to app failures with various

methods including “no-action”, “stop”, “restart”, “stop-all”, “restart-all”, and “failover”.

OS Bridge & FTP

Another important infrastructural flight software component is the Operating System (OS) Bridge. The OS Bridge is the application responsible for fielding commands from ground operators to interact with the CFC-400 OS and file system, and to execute predefined scripts. The OS Bridge also helps monitor compute resource usage, and enables an innovative shell-like command capability (termed Direct Mode) by SSCI operators on the ground during ground contact. Lastly, the OS Bridge is the primary manipulator of the FTP server used for transferring files between the PICU & CFC-400 for all mission data & logs, including automatic end-of-window downloads. Unlike the mission applications, the OS Bridge implements the necessary messaging and maintains direct low-level communication with the PICU & communications system. The OS Bridge component & functionality is a critical enabler for updating the FSW & OS on-orbit.

HARDWARE SUBSYSTEMS

CFC-400 Mission Processor

Innoflight’s CFC-400 is a flight-proven Commercial-off-the-shelf (COTS) onboard mission processor with advanced capabilities, and is shown in Figure 3. The CFC-400 incorporates a rad-tolerant Multi-Processor System on Chip (MPSoC) which provides extended reliability in space environments. The CFC-400 architecture provides high-performance operation in a 0.5U CubeSat form factor. The CFC-400 receives GPS and time from the bus, and also has Ethernet & RS422 connections to the other subsystems via the PICU & PDPU.

Table 1: Innoflight CFC-400 Specifications

CFC-400 Parameter	Value
Max Power Draw	15W
CPU	Quad-core ARM x64 @ up to 1.2 GHz
RAM	2Gb
Persistent Disk Storage	16 Gb
FPGA	~550K Flip-Flops, ~4 MB Block RAM, 512 Mb Volatile RAM
Operating System	Commodity Linux

Ground-based Hardware-in-the-Loop (HITL) tests on the CFC-400 show the constellation autonomy software using a very lean 10% CPU and 12% memory under nominal planning & execution. The largest onboard compute resource usage comes from the ML-based ATR, which when executing can be tuned to use more or less memory, traded against execution time.

Max continuous processing time for the Sagittarius A* payload system is ~24 minutes, only limited by Mission Reservation Agreement with Loft Orbital.

Figure 3: Innoflight CFC-400 Mission Processor

Spacecraft Bus

The spacecraft bus for this mission, shown in Figure 4, is a LeoStella LEO-100 bus, manufactured near Seattle, Washington. The bus product line is manufactured by LeoStella for the BlackSky imaging constellation, and six of this bus model are on orbit. The bus includes an Attitude Determination & Control System (ADCS) to perform 3-axis control with 4x reaction wheels, 7x sun sensors, 2x star trackers, 2x IMUs, 1x GNSS receiver (with an additional GNSS receiver in the Loft Orbital Payload Hub), 1x magnetometer, and 3x magnetorquers.

Table 2: LeoStella LEO-100 Bus Specifications

Bus Parameter	Value
Pointing Accuracy	55 arcsec (1 sigma)
Pointing Knowledge	25 arcsec (1 sigma)
Slew Rate	Minimum 1 deg/s all axes
Electrical Power System	2x fixed Solar Array 2 x 4.5 AHr lithium ion batteries
Peak Power	150W for Payloads
Orbit Average Power	55 W for Payloads
Planned Orbit	500-600km, Sun-Sync Orbit

Electro-Optical Imaging Payload

The imaging payload, shown in Figure 5, is an electro-optical, nadir-aligned, Simera MonoScape100 snapshot imager employing a monochrome AMS CMV12000 sensor. The MonoScape100 is produced by Simera Sense Ltd. in South Africa, and is intended for earth

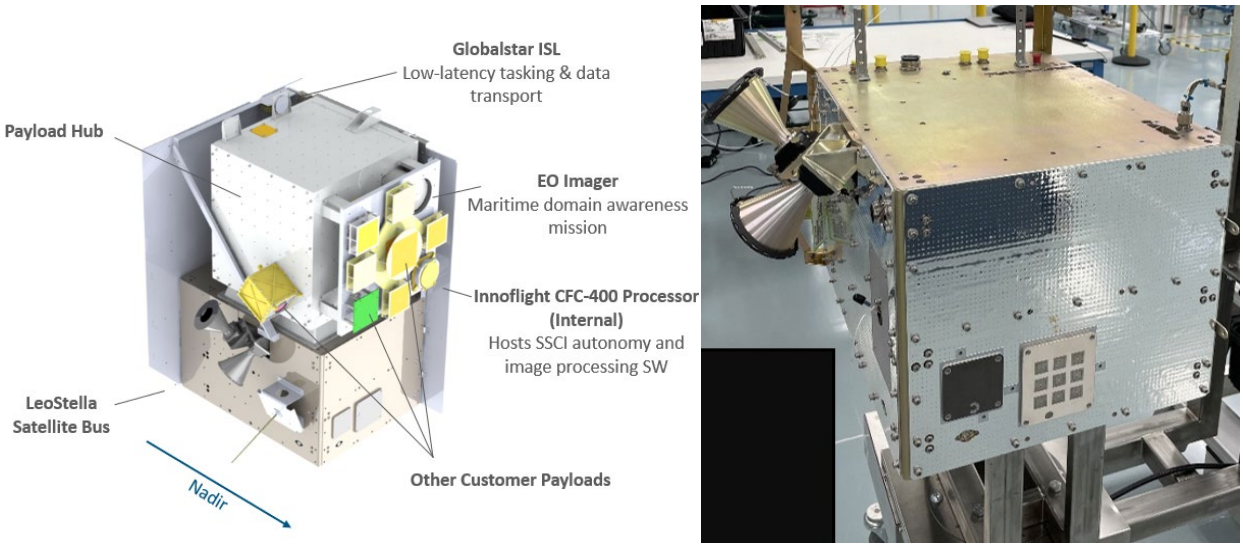

Figure 4: (a) Spacecraft Bus & Payload Hub Integrated Diagram, (b) Spacecraft Bus without Payload Hub

observation applications. Once delivered to the CFC-400, imagery is formatted as GeoTIFF for mission applications. The imager itself can store 128Gb of images, or approximately 8000 frames, and has several useful imaging modes including burst capture (60 fps), video at reduced resolution, thumbnailing, and image compression. Images average 10-15Mb uncompressed.

Table 3: Simera MonoScape100 Specifications

Imager Parameter	Value
Footprint (cross-track)	19.4 km @ 500 km orbit
Footprint (along-track)	14.6 km @ 500 km orbit
Ground Sample Distance	4.75 m @ 500 km orbit
Detector Effective Pixels	4096 x 3072
Detector Pitch	5.5 µm
Aperture	95 mm
Focal Length	580 mm
Maximum Frame Rate	150 fps
Spectral Range	Monochromatic spectral response, from 450 nm to 900 nm

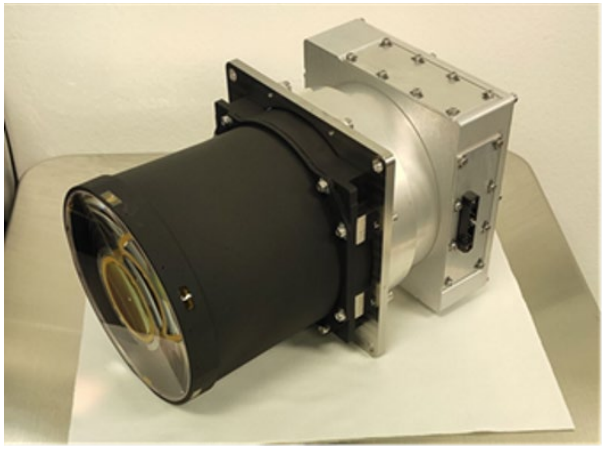

Figure 5: Simera MonoScape100 EO Imager

Loft Orbital PDPU & PICU

To interface multiple payloads with the bus & auxiliary communications, Loft Orbital provides a Payload Hub as part of the Mission-as-a-Service offering. The Payload Hub includes a Payload Data Processing Unit (PDPU) and Payload Interface Control Unit (PICU) to which the CFC-400 mission data processor and Simera imager interface. Between the CFC-400 and PICU/PDPU, the traffic is CCSDS packets containing FlatBuffer messages. The PICU & PDPU provide connection to ground link & Globalstar communications, power management, mission data storage & retrieval, bus & payload status information, and ADCS/GNSS/time data. The PICU acts as a translator between bus and payload protocols and electrical interfaces. The PICU & PDPU allow isolation between the payloads and bus, and with the SSCI platform-agnostic C3 software, collectively enables mission-specific payloads & software applications (including autonomy software) to fly on a variety of buses with no modifications to the payloads or bus software or hardware. The system supports common space industry data interface standards including SpaceWire, CAN, LVDS, CAMLINK, Ethernet, SPI, UART, SERDES, RS422, and others.

Communications

The spacecraft communication subsystems are used to receive Telecommand (TC) and files uplinked from the ground, and transmit Telemetry (TM), payload data, and mission data products to the ground. Communications onboard the spacecraft include a GlobalStar low-rate duplex inter-satellite link (ISL) with 5-8 bytes/sec data rate for low rate tasking and status monitoring throughout nearly all of the YAM-3 orbit. The satellite also has X-band & S-band radios for TC/TM and payload data downloads to the ground

stations. The ground station links are S-band for TC with an uplink rate 154 kbps, X-band downlink for TM and mission data at 94 Mbps, and a UHF backup. Large files of 10s to 100s of Mb can be uploaded in standard operations via file sharding across multiple passes.

Payload Data Latency from on-orbit collection to SSCI servers is expected to be less than 120 minutes. This latency applies to data downlinked via the bus X-band and S-band radios, whereas the Globalstar ISL low-rate data is near-real-time. For all communications, end-to-end AES-256 encryption is used from Loft Orbital's servers to the spacecraft. All data transfers are CRC error checked to ensure complete transfers.

GROUND SOFTWARE

Loft Orbital provides a web-hosted ground system called Cockpit to handle ground contacts for TC/TM and data uplink/downlink communications, to which is interfaced the Sagittarius Team's cloud-based, virtual Mission Operations Center (vMOC) ground software. Cockpit is designed for scheduled payload commanding and operators can command payloads on a task-by-task basis or define rules for automated operations.

The Cockpit ground system allows the SSCI ground software to communicate directly with the CFC-400 processor, via the set of abstractions provided by the Loft Orbital ground software & PICU/PDPU hardware & software, including communicating with the satellite via the GlobalStar duplex link for real-time requesting.

The Sagittarius A* mission demonstration includes sending user MSRs as well as tip-and-cue automated tasking requests driven by RF data from the vMOC. Low rate data returned over Globalstar includes Work Item status & object detection notifications. The larger image & telemetry data are automatically provided to SSCI operator web-based UIs, an API for interactive querying & TC/TM, and in the case of files, automatically transferred to SSCI's cloud servers after ground passes.

Mission Control Interface

The Cockpit ground system exposes multiple programmatic APIs in addition browser-based UIs, and the Sagittarius ground software applications interoperate via several additional internal APIs. To collect and integrate these various APIs, SSCI developed a ground component called the Mission Control Interface (MCI) to perform ground-based data control & routing between APIs. This component forms the software glue between the Cockpit software and the rest of the vMOC applications, including Order Logic and SpyMeSat Servers, TAK Server for ATAK, AWS S3 file I/O, telemetry I/O, and ground CMA node for realtime C3 software synchronization to the ground.

SpyMeSat and OrderLogic

The SpyMeSat and OrderLogic applications are tailored commercial applications designed for MSR submission, tip-and-cue request translation, request tracking, user management, and region management. The SpyMeSat application is a mobile app for smart devices, and is a prototype UI to allow tactical users to submit MSRs to the spacecraft C3 autonomy. These ground software applications then allow users to monitor the fulfillment status of their MSRs. A Mission Service Request (MSR) is a semantic-level request to the spacecraft (or future constellation), and is a minimal amount of information, such as "find objects of interest in an AOR centered at XYZ with radius 500nmi over the next 7 days". This eliminates user training for overhead sensing expertise, such as specific asset capability, configuration, location, or other detailed tasking parameters.

ATAK

Disseminated data products associated with a request are downlinked to users once the Pipelines are complete, and displayed on an Android Tactical Assault Kit (ATAK) smart device. Target track information produced by the onboard ATR algorithms are translated into Cursor-on-Target messages and overlaid on a map display with raw image data. The display provides actionable tactical information, and allows flexible display of tracks, detections, and entities, all configurable to user needs. Additionally, multiple users can access the same mission data on individual devices.

Ground-Based Collaborative Mission Autonomy Node

As mentioned in the Constellation Mission Management Software section, the onboard CMA software manages potentially thousands of Pipelines of Work Items stored in and constituting the Common Relevant Operating Picture. These Pipelines & Work Items are updated over time as they are completed, and are synchronized to the ground. The architecture includes a partial "ground node" of the onboard software in the vMOC, which receives synchronization of CROP from the satellite to provide operators & future tactical users with realtime Situational Awareness. This CROP synchronization is a critical mechanism for future constellation multi-asset coordination.

INTEGRATION & TEST

Methodology & Approach

The bus, imager, and PICU/PDPU testing and qualification were performed by Loft Orbital as part of their Mission-as-a-Service offering. SSCI, through teamwork with Loft Orbital, performed the I&T, qualification, and checkout of the CFC-400 processor and all mission software.

To organize development, test, and integration of the 13+ mission software applications, a Mission Software Integration & Test Plan was developed, with an integration roadmap for the large number of mission software applications & companies contributing. The test plan outlined the test phases, types, and methodologies to allow each contractor to develop their software, perform component level verification, and deliver software for integrated testing. The software development process began with development of requirements and design documents. The flight software implementations and port-overs began in March 2020 with initial end-to-end capability demonstrations first shown in a software test environment in November 2020.

Software testing is performed at multiple stages during integration. At a high-level, the test approach was to perform:

- **Component-Level Unit Testing**, where each development team was responsible for unit testing using industry best practices, and releasing software to the integrated team testing environment.
- **Integrated Software Testing**, where container-based tools were (and continue to be) used for rapid test iteration. At this level, full internal software functionality can be exercised, but without flight-like hardware environment and interfaces in the loop. Initial integrations, tests, and demos were executed with Docker container, development board, and Engineering Development Units (EDU) integrated software environments.
- **Benchtop Testing**, which covers a range of test activities with a mixture of test environments and setups. Higher-fidelity hardware was gradually introduced to form an integrated Payload Hub with both Payload and PICU flight units. Functionality was assessed at each level of integration using functional test procedures. This included Software Acceptance Testing at the end of the software integration phase using EDU & Flight units.
- **Integrated Spacecraft Testing**, which was performed once the payloads were integrated into the Payload Hub, and the Payload Hub integrated into the spacecraft. Functional tests were executed to regression test the software on the flight units as integrated, as well as perform Day-in-the-Life (DITL) testing.
- The last step remaining is **On-Orbit Testing** via the Ground Segment, once the satellite is integrated with the ground segment, launched, and commissioned on-orbit.

Integration & Test Execution

Primary Test goals throughout the I&T campaign were:

- Mission software communication & interface verification,
- Spacecraft to Mission Processor (CFC-400) interface verification & logic,
- TCPED processing chain (Imaging & Object Detection) execution and end-to-end data flow,
- and Software Update, OS Update, and onboard Configuration Management (CM) verification

Table 4: Functional Test Procedure List

ID	Functional Test Procedures
PBFT-001	OS Install & Mission Processor Setup
PBFT-002	Power On & Boot Sequence
PBFT-003	Nominal Imaging Sequence
PBFT-004	Basic Mission Software Update
PBFT-005	OS Image Load
PBFT-006	Operation Mission Software Update
PBFT-007	Operational Image Sequence
PBFT-008	Mission Executive Test
PBFT-009	OS Bridge Test

In addition to Functional Test Procedure execution, early demos were developed to establish progressive, Agile-style demonstrations and drive down integration risk for the two key capabilities: Imaging and Object Detection, and Flight Software Application Update.

The flight software was built and tested for multiple platforms in multiple test environments. The test environments used included either full or partial use of simulators. The flight software was run on representative hardware which used the same processing architecture (ARM64) and processing core (Corex-A53) and memory. For software integration the test environments included:

- Software-in-the-loop (SITL) integration & test using a Docker container environment
- ARM x64 Xilinx Zynq Ultrascale ZCU-102 Evaluation Board (as representative processing hardware) hardware-in-the-loop (HITL)
- ARM x64 Engineering & Flight Module CFC-400 processor hardware-in-the-loop (HITL)

During the I&T and qualification, these processor environments were combined & recombined with a mixture of: GSE simulations of the PICU, EGSE hardware of the PICU, and flight PICU & PDP hardware. Figure 6 shows the test setup with PICU EGSE and a CFC-400 EDU running HITL.

With major development epic & release completion, each major & minor software version was released with a Version Description Document (VDD) to record the software configuration, release notes, software inventory, incorporated change list, test set executed, any known issues, and proper usage. During the T&E, risks were identified & tracked, deficiencies were identified & dispositioned, and new features identified.

Status & Qualification

For the processing hardware, the CFC-400 Flight Model was delivered in June 2020 with an Acceptance Test Procedure executed in August 2020. Throughout September 2020 – February 2021, the PICU EGSE was used for testing and software I&T, with HITL testing & flight qualification complete in April 2021 by SSCI Team personnel on site at Loft Orbital's facility in Golden, CO. This HITL testing demonstrated all necessary mission software requirements and benchmarked compute resource usage.

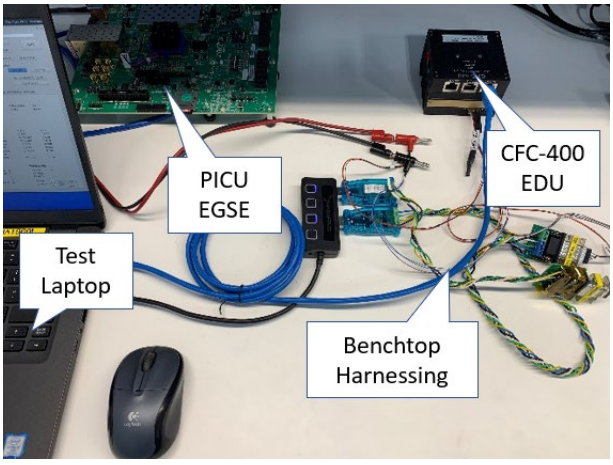

Figure 6: Benchtop HITL Test Setup

The Loft Orbital Payload Integration was completed in early May 2021, and Space Vehicle integration completed in May 2021, with a Pre-Ship Review (PSR) held in May 2021 & satellite integration to the launch vehicle completed for launch in June 2021.

PATH FORWARD

Launch & Commissioning

The spacecraft is scheduled for launch on SpaceX's dedicated rideshare Transporter-2 Falcon 9 rocket in June 2021, from Cape Canaveral, FL. The Sagittarius A* payload commissioning will begin once Loft Orbital completes bus, PICU, and PDPU component checkouts & commissioning following separation from the launch vehicle. Commissioning of the payload is expected to last one month, and verify all software application execution. Commissioning & early operations will include analysis of imagery and detection processing, assessing latency, and assessing mission autonomy software performance. Following the commissioning of the payload, operations will begin.

Flight Operations

Operations will include performing dynamically requested and scheduled imaging, while working up to progressively more complicated end-to-end autonomous demonstrations of object detection and onboard ATR, including via cueing from offboard data. Objectives will be to initially establish an experimentation baseline for the EO Imager and CFC-400 mission processor, and work up to more challenging & interesting autonomous operations scenarios.

During this time, compute resource usage, image quality, ATR performance, and mission processor performance will be analyzed. These operations & demonstrations will be executed by the distributed SSCI team via the cloud-based, vMOC SSCI has developed. Any change requests & anomalies will be resolved, with software checked out on the ground-based HITL testbed prior to upload & checkout on the mission processor on-orbit. During operations, daily health and safety checking will leverage the Cockpit software to assess status of all components via telemetry trends. Special activities may also be performed for specific collections, testing newly uploaded algorithms, and performing follow-on mission demonstrations.

Demonstrations

Initial demonstrations for this mission include object detection based on offboard cues with onboard imaging & ATR, and upload of new algorithms to the testbed for

on-orbit checkout of autonomous mission operations. In the future, strong interest from the Space community exists for continuing support and experimentation on the testbed.

Future opportunities include:

1) Characterization of dynamically-determined edge vs. ground processing.
2) Theater C2 & cross-mission machine-to-machine, multi-domain demonstrations for tactical ISR.
3) Integration & interoperability demonstrations with commercial ISR systems.
4) Demonstration of extending conventionally ground-centric DevOps to space for mission refresh & onboard update, including via uploaded containers.

Acknowledgments

The team who worked this program is significantly larger than the authors listed in this paper. The authors would like to thank our additional SSCI team members & colleagues from Loft Orbital, Orbit Logic, Emergent Space Technologies, Oxford Systems, Leaf Labs, Kitware, and Raytheon BBN for their contributions toward this mission. This mission was funded in part by the Defense Advanced Research Projects Agency (DARPA). This paper is Approved for Public Release, Distribution Unlimited, by DARPA.

References

1. Couchman-Crook, R., et al. "NovaSAR and SSTL S1-4: SAR and EO Data Fusion," Proceedings of SmallSat Conference, Logan, Utah, August 2020.
2. Herz, E. et al., "Onboard Autonomous Planning System," Space Ops Conference, Pasadena, CA, May 2014.
3. Woodbury, T., "DASHER: Execution Software for Distributed Space Mission Autonomy", 2021 Flight Software Workshop, John Hopkins University/Applied Physics Laboratory, February 2021.
4. Center, K., et al. "Improving Decision Support Systems Through Development of a Modular Autonomy Architecture," I-SAIRAS 2012, Turin, Italy, September 2012.