

AI for Space: A Survey of Guidance, Navigation, Control, and Thermal Management

Jia Deng, Zizhe Wang, Xin Lu, and Zehao Jia

Astroup Lab¹ | astroup.com

{dengjia, wangzizhe, luxin, jiazehao}@astroup.com

Abstract—Learning methods can help spacecraft estimate their state, plan maneuvers, control motion, detect faults, and manage thermal and electrical resources. This structured narrative survey examines how these methods work within the rest of the flight system. Representative studies are compared by task, learned component, baseline, reported benefit, and evaluation environment. The examples include image-based pose estimation, trajectory initialization, feedback control, telemetry anomaly detection, and fast thermal prediction. They span synthetic data, ground experiments, processor tests, flight-data analysis, and learned attitude control in orbit. Across these applications, learning often supplies an estimate, a predictive model, or a candidate trajectory, while conventional estimation, optimization, and control software completes the processing chain. LeLaR illustrates the additional calibration, sensing, and monitoring work involved in operating a learned controller on a satellite. Thermal and energy constraints link maneuver planning to pointing, computing load, temperature, and battery reserve. We connect these findings to algorithm selection, runtime supervision, and current space-software guidance. The survey concludes with practical steps for comparing methods and testing motion, thermal, energy, and computing decisions together.

Index Terms—Spacecraft guidance, navigation and control, learning-based control, thermal control, energy management, fault detection, simulation-to-reality transfer, runtime assurance.

I. INTRODUCTION

SPACECRAFT autonomy is needed whenever the vehicle must respond faster than a ground team can intervene, continue through a communications outage, or manage more operations than can be scheduled individually from Earth. These demands arise in different forms: routine operations across a constellation, close inspection of a non-cooperative target, a terminal landing maneuver, or a long-duration deep-space mission. Model-based estimation, feedback control, optimization, and onboard planning already support autonomous flight [1]–[4]. This survey asks where learning can improve those capabilities and what it takes to make those improvements useful in flight.

Learning can enter the flight system at several points. Supervised models can extract pose-related features from images or approximate the output of an optimal-control solver. Reinforcement learning (RL) can search for feedback policies through simulated interaction. Unsupervised and self-supervised methods can identify structure in telemetry or adapt perception to a new image domain. Each role has different data needs, interfaces, and failure modes. Deep reinforcement

learning (DRL) algorithms such as proximal policy optimization (PPO) and actor–critic methods provide tools for learning these behaviors [5]–[7]; the spacecraft task determines how they are trained and evaluated.

Existing surveys provide complementary entry points. Izzo et al. [8] review AI in spacecraft guidance, dynamics, and control; Tipaldi et al. [9] examine RL applications in spacecraft control; Thangavel et al. [10] discuss trusted autonomous satellite operations; and Fejjari et al. [11] focus on telemetry anomaly detection. The present article connects these functional perspectives with deployment evidence and the thermal–energy constraints under which the algorithms must operate. Recent examples make that connection concrete: learned pose estimation is evaluated on laboratory images designed to expose domain shift [12], [13]; transformer models supply initial guesses to constrained trajectory optimizers [14], [15]; and LeLaR reports a learned attitude controller operating on a satellite [16]. These examples demonstrate different capabilities at different levels of integration.

Thermal control and energy management complete this systems view. Onboard computation draws electrical power and produces heat; instrument pointing, propulsion, and communications also affect the resources available to an autonomous maneuver. Established thermal models and controllers already manage these changing conditions [17], [18]. Learned predictors can make repeated calculations faster and help schedule work around thermal and energy limits. This coupling motivates treating motion control and resource management together.

A. Scope and Contribution

This survey examines AI for spacecraft guidance, navigation, control, and thermal management, from state estimation and maneuver planning to feedback control and temperature prediction. Fault detection, isolation and recovery (FDIR), energy management, and onboard computing connect these functions to the spacecraft’s operating conditions. Verification and deployment evidence help assess how the methods can be used in flight.

This survey shows where learning enters the flight software, what improves against each study’s baseline, and which tests connect that gain to onboard use. Representative studies are compared by their task, learned component, inputs and outputs, constraints, and evaluation environment. Following a learned output through estimation, optimization, control, and supervision reveals which parts of the result depend on conventional

¹Astroup Lab: 向上跃升（北京）科技有限公司.

software. Thermal and energy models then connect these choices to the power, temperature, and computing limits of a maneuver. Together, these comparisons help identify which module is worth changing and what evidence is needed to use it.

The companion surveys discuss the broader embodied-intelligence framing and space-robot operations [19], [20]. This article focuses on spacecraft motion and the management of onboard faults, temperature, and energy.

B. Autonomy Levels and the Place of AI

Algorithm type and mission autonomy are separate dimensions. In ECSS-E-ST-70-11C, the nominal mission-execution levels E1–E4 distinguish ground control, execution of ground-preplanned operations, event-driven adaptive operations, and goal-oriented mission replanning [21]. These are operational capabilities. Installing a learned estimator or an RL attitude controller does not by itself establish a particular ECSS level. Software criticality and the authority to command an actuator are further, separate properties.

Earlier flight systems also show why autonomy should not be equated with machine learning. The Deep Space 1 Remote Agent used model-based reasoning, while EO-1’s Autonomous Sciencecraft Experiment combined onboard planning and execution [4], [22]. Their relevance is the allocation of decisions between ground and spacecraft. Learning-based systems add components whose behavior depends on training data and optimization, requiring additional evidence about their operating domain and response to unfamiliar inputs.

Fig. 1 shows the closed-loop architecture used to organize this survey. Guidance supplies a reference trajectory, and the controller combines it with the estimated state to produce commands. A supervisor checks those commands and can filter them or change the operating mode. Thermal and energy states shape the feasible maneuver and computing budget. Learned components can enter at several points in this loop; their interfaces determine which errors and delays matter to the rest of the system.

C. Survey Method and Evidence Boundaries

This is a *structured narrative survey*, with literature coverage closed on October 15, 2026. Sources were obtained through publisher websites, author repositories, arXiv, public dataset pages, and official agency or project websites. Reference lists in selected papers and related surveys were followed to trace methods and mission claims to their original sources. Studies were selected for their spacecraft function, engineering relevance, and description of the learned component, evaluation setup, or deployment experience. The aim is to explain representative approaches and their relationships, rather than estimate how frequently each method appears in an exhaustive literature search.

The core comparisons use spacecraft studies in guidance, navigation, and control (GNC), FDIR, thermal–energy management, and onboard computing. Classical autonomous systems provide flight baselines; general learning methods and

terrestrial applications supply background where they help explain an approach. Published articles and preprints describing the same study are treated as one work. Bibliographic details follow the published record where available, while public author manuscripts provide access to the technical results. Official mission records support operational claims, datasets support benchmark descriptions, and development proposals describe research aims.

For each functional group, the synthesis asks what is learned, which measurements and actuators it uses, and how it compares with the study’s baseline. Training data, evaluation data, and constraints handled by other software are recorded separately. The tables summarize the learned module, test environment, and main result; the text gives the conditions needed to interpret representative numerical comparisons. Hardware-in-the-loop (HIL) imagery, processor execution, physical closed-loop experiments, and in-orbit operation answer different questions and are identified throughout the review.

Selection and reporting remain limited by the availability of public implementations, flight data, and comparable evaluation protocols. Studies using different dynamics, sensors, objectives, or test distributions cannot be ranked by a single performance number. The conclusions therefore identify recurring design patterns and the evidence required for their use, while keeping mission-specific results within their demonstrated scope.

II. MISSION SCENARIOS AND AUTONOMY DEMAND

Five scenario families organize the discussion. Their main demands arise from reaction time, uncertain geometry, contact, limited communications, or coordination among vehicles. Table I connects these demands to representative learning studies. Each mission sets its own planning horizon, estimator update rate, and control period.

A. Routine Attitude and Orbit Control

Pointing, slews, momentum management, and station-keeping provide familiar interfaces for evaluating learned control. A policy can be compared with a conventional controller under the same state estimate, actuator limits, and maneuver request. LeLaR supplies an in-orbit attitude-control case [16]. Simulation studies address Sun–Earth halo-orbit station-keeping [23] and low-thrust tight station-keeping in low Earth orbit [24].

The uncertainty depends on the vehicle and maneuver. Inertia changes, flexible appendages, wheel behavior, thrust errors, and environmental disturbances may matter even when the nominal task is routine. Comparisons should specify pointing or orbit error, control effort, settling behavior, and failure cases under common dispersions. The spacecraft’s actual processor and power allocation determine the computing limit; it cannot be inferred from the task name alone.

B. Rendezvous and Proximity Operations (RPO)

Rendezvous, docking, and inspection couple relative navigation to collision avoidance, approach geometry, and actuator

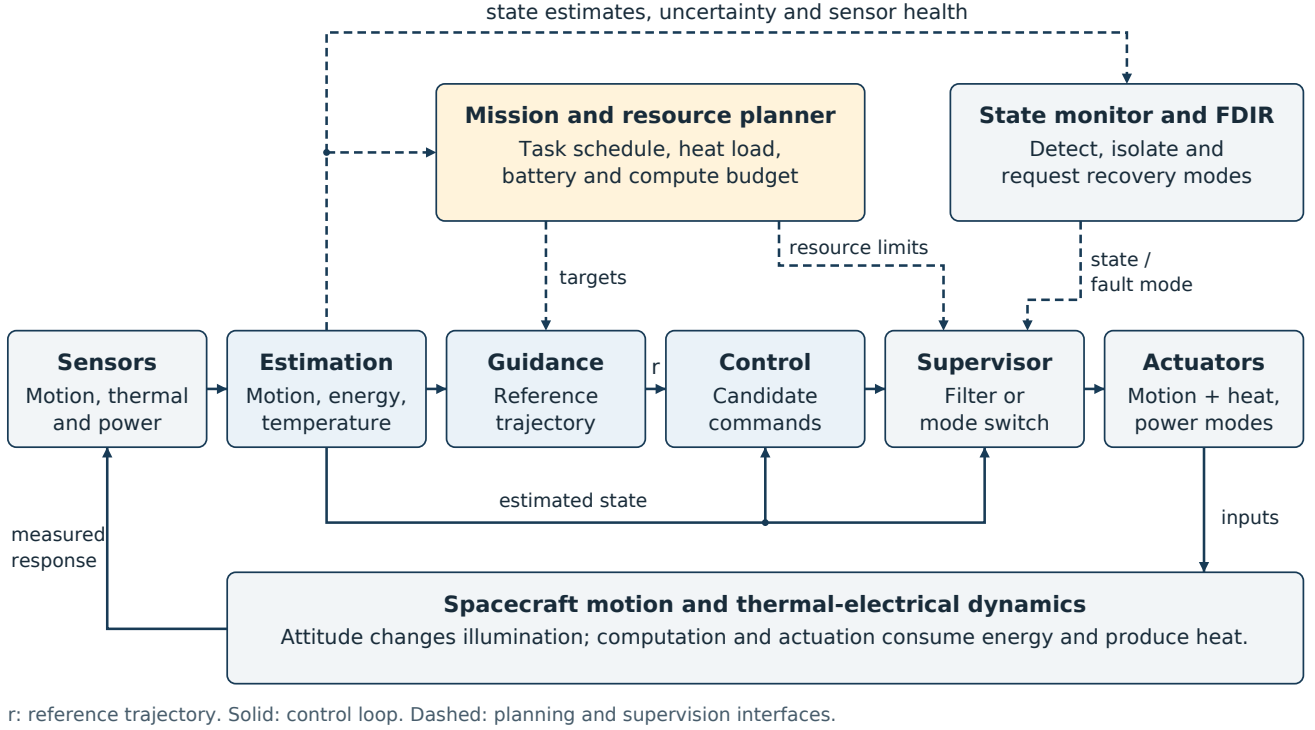


Fig. 1. Functional organization of onboard autonomy with runtime supervision. Guidance sends a reference trajectory r to the controller; state estimates also feed the controller and supervisor directly. Planning and FDIR provide task, resource, and fault information. Learned components may be used within these functions.

restrictions. A cooperative target may provide navigation aids, while a non-cooperative target may require image-based pose estimation and uncertain shape information [25]. Guidance studies range from learned proximity policies [26] to constrained docking and runtime-assured inspection [27], [28].

Evaluation must include the observation loop. A policy receiving exact simulated state has not demonstrated tolerance to image dropouts, ambiguous poses, or delayed estimates. Aerial-surrogate experiments provide useful evidence that an RL policy and its runtime-assurance software can operate in a physical loop [29]. They do not reproduce orbital free fall, vacuum actuation, or all relative-motion dynamics. The transfer value lies in the specific interfaces and failure modes exercised.

C. On-Orbit Servicing, Assembly, Manufacturing, and Debris Removal

Servicing and debris removal add capture, contact, and changes in the controlled system after attachment. Orbital Express and RemoveDEBRIS provide mission context for these operations [30], [31]. Restore-L/OSAM-1 represents a development program rather than a completed flight demonstration: NASA's project status records its cancellation and orderly shutdown [32]. These histories should be separated when discussing flight evidence.

Learning-based studies address visual tracking and capture of a spinning target in a planar laboratory experiment [33], pre-capture motion under domain randomization [34], and manipulator planning [35]. Their objectives and validation

stages differ. The main interfaces connect uncertain target state, capture geometry, contact forces, and the changing inertia and flexibility of the coupled system. Section V examines how the learned and conventional parts work together during capture; the companion survey covers the robotic interactions in more detail [20].

D. Planetary Descent, Landing, and Terminal Autonomy

Powered descent combines short decision horizons with fuel, thrust, attitude, and touchdown constraints. RL studies cover planetary landing [36] and reusable-launcher terminal recovery [37]; the latter includes atmospheric flight conditions that should not be transferred unchanged to an airless-body scenario. Terrain-relative navigation adds an image-to-map estimation problem. Mars 2020's Lander Vision System (LVS) is a flown classical baseline for that function [38], [39].

DART's SMART Nav provides another example of image-based terminal autonomy: it tracked a target and guided a kinetic impact [40]. The task ties optical tracking to a terminal maneuver, with different objectives and approach constraints from a safe landing. Terrain interpretation also supports surface mobility, an adjacent application in which route planning and driving introduce a different control loop.

E. Multi-Spacecraft and Distributed Autonomy

Formation maintenance, reconfiguration, and distributed inspection introduce shared objectives and coupled constraints. Examples include RL-based multi-impulse reconfiguration [41], learning-based model predictive control (MPC)

for a spacecraft swarm [42], and decentralized inspection policies [43], [44]. Local decision making can reduce communication demands, but it also makes state consistency, coordination failures, and collision constraints part of the evaluation.

Large constellations motivate scalable screening, planning, and operations support. Learning is one option alongside model-based screening, optimization, and rule-based automation. Its benefit should be measured against the mission's workload and baseline, including communication and computation costs. Beyond Earth orbit, limited access to navigation signals and long communication delays strengthen the case for onboard estimation and replanning. Classical optical navigation and autonomous planning provide established baselines [3], [4].

III. AI FOR NAVIGATION

Navigation converts measurements into the state information needed by guidance and control. A learned image front end, a pose solver, and a temporal state estimator perform different functions. Their errors also have different consequences: a visually plausible pose may be dynamically inconsistent, and a small average error may conceal rare failures near a safety boundary. This section therefore separates the learned module from the surrounding navigation chain.

A. Classical Absolute-Navigation Baselines

Attitude and translational navigation rely on different measurements. Star trackers and inertial sensors support attitude estimation; global navigation satellite systems (GNSS), radio-metric tracking, and optical observations can constrain position and velocity, depending on mission geometry [1], [2]. Deep Space 1's AutoNav used optical observations for onboard navigation [3], while NICER/SEXTANT demonstrated pulsar-based navigation using X-ray timing measurements [45]. These classical flight systems provide operational baselines for evaluating learned estimation components.

X-ray pulsar navigation (XNAV), for example, involves signal accumulation, pulse timing, and estimation relative to a reference model [46], [47]. A learned signal-processing component would be useful if it improved the resulting state estimate or reduced observation time under comparable photon statistics and detector conditions. The comparison therefore needs to follow the signal through to the navigation output.

DART illustrates the connection between navigation and guidance. Its SMART Nav system extracted image blobs and centroids, tracked the selected target, and generated divert maneuvers to impact Dimorphos [40]. This classical terminal-navigation chain provides a concrete reference for assessing image-processing latency, target-selection errors, and their effect on the commanded maneuver.

B. Relative Navigation about Non-cooperative Targets

Monocular pose estimation is a well-developed learning application within the surveyed navigation literature. Sharma and D'Amico combine neural-network inference with geometric information to estimate the relative pose of a known

target [25], [48]. This task differs from estimating an arbitrary unseen spacecraft: assumptions about geometry, visible features, and range constrain what can be recovered from an image.

Open datasets make those assumptions easier to study. SPEED supports a common pose-estimation benchmark; SPEED+ adds distinct laboratory image domains to expose the gap between synthetic training images and physical imagery [12], [49]. SPEC2021 analyzes this domain gap across competing approaches [13]. SHIRT adds rendezvous sequences, while URSO provides photorealistically rendered scenes [50], [51]. SPARK addresses spacecraft recognition with rendered data [52]. Recognition, single-image pose estimation, and tracking should retain their own metrics even when they share images or network components.

SPNv2 trains a shared visual encoder on synthetic images for detection, pose, and segmentation, with optional online domain refinement (ODR) using unlabeled images [53]. On the SPEED+ lightbox domain, the offline model with scale parameter $\phi = 3$ and batch normalization reduces the reported mean translation and rotation errors from the style-augmented Keypoint Regression Network (KRN) baseline's 1.06 m and 36.14° to 0.175 m and 6.48° . Refining that SPNv2 model with 1,024 unlabeled lightbox images reduces the errors further to 0.142 m and 5.62° . Thus, the study separates gains from the offline architecture and training from the additional benefit of adaptation. It also evaluates the offline models on *prisma25*: 25 archived PRISMA flight images of Tango, processed on the ground and excluded from ODR. Adaptation improves average HIL accuracy, although some individual predictions worsen. Temporal tracking offers another way to use information across frames, as examined by Liu et al. [54].

Other architectures change the information available to the pose solver. Despond and Ulrich use stereovision with convolutional neural networks (CNNs) [55]; Bechini et al. use learned target and line-segment detection within a geometric pose pipeline [56]. Bates and D'Amico address ambiguity when shape and pose must be estimated together [57]. These choices should be compared against the sensing arrangement and geometric knowledge available on the actual mission. A second camera, a known wireframe, and a shape prior provide different sources of information.

Event-based sensing offers a different hardware and representation choice. Rathinam et al. train keypoint-regression networks on synthetic SPADES data and convert quantized models for Akida inference [58]. The coordinate-regression model runs on physical Akida V1 hardware, whereas the V2 heatmap model is evaluated through Akida Cloud simulation. V1 loses accuracy after 4-bit quantization, showing why processor efficiency and pose error need to be measured together. The reported tests use ground-truth target crops; real event streams and target detection remain outside that evaluation. Event cameras' response to brightness changes and sparse output can be useful under fast motion or high contrast [59]–[61]. Their value in navigation depends on the complete chain from event accumulation to geometric pose recovery.

Visual odometry provides a useful comparison for temporal estimation. Semi-direct visual odometry (SVO) estimates

TABLE I
MISSION DEMANDS AND REPRESENTATIVE EVALUATION EVIDENCE

Scenario	Demand to specify	Learning role considered	Evidence boundary
Attitude and orbit operations	Pointing or orbit tolerance; actuator limits; estimation and control periods	Feedback policy or maneuver scheduling	LeLaR attitude flight case; station-keeping studies use simulation [16], [23], [24]
RPO and inspection	Relative-state uncertainty; approach/keep-out constraints; abort authority	Pose estimation and constrained guidance	HIL imagery, simulation, and aerial surrogates test different parts of the loop [12], [29]
Servicing and debris removal	Target motion; contact; base-arm coupling; post-capture dynamics	Visual tracking, approach, and capture planning	Planar capture/retraction experiments and simulated pre-capture control [33], [34]
Landing and terminal operations	Terrain or target uncertainty; thrust/fuel limits; terminal deadlines	Learned descent policy or terrain interpretation	Learned landing is evaluated in simulation; LVS provides a classical flight baseline [36], [37], [39]
Distributed spacecraft	Communication limits; shared-state consistency; coupled collision constraints	Reconfiguration and inspection policies	The cited learning studies evaluate specific simulated coordination tasks [41]–[43]

motion through classical photometric and geometric optimization [62]. IONet learns inertial motion patterns from terrestrial data [63]. Applying this latter approach to spacecraft would require training and evaluation for the vehicle’s motion and sensor configuration, since its terrestrial motion priors supply information that may be absent in free flight.

C. Terrain-Relative and Small-Body Navigation

Flown terrain-relative navigation shows the importance of map quality and complete-loop validation. Mars 2020’s classical LVS matched descent images to a preloaded map and supported the rover’s safe landing in 2021 [38], [39]. Johnson et al. describe the design and expected performance; the later mission record establishes flight use.

At Bennu, OSIRIS-REx used Natural Feature Tracking with a prepared terrain model and feature catalog to guide its sampling descent [64], [65]. Hayabusa2’s target markers and optical tracking provide another small-body navigation approach [66]. These cases place different burdens on prior mapping, target preparation, and onboard recognition. Comparing only final landing or sampling accuracy would hide those differences.

Silvestrini et al. provide a learning-based example for lunar absolute navigation [67]. A CNN detects craters, geometric processing matches them to a database of known locations, and an extended Kalman filter (EKF) combines the optical measurements with altimeter readings while accounting for measurement delays. Training and trajectory testing use synthetic images. The detector reports an average crater-center error below approximately 3 pixels. On the prototype trajectory, the complete pipeline yields horizontal and vertical position errors of roughly 200 m, with crater-database navigation used down to about 20 km altitude. These results connect learned feature extraction to a state estimate; the detector’s image error is one contribution to the final navigation error, alongside matching, sensor noise, and timing. Image-to-control landing policies use a different interface and are discussed in Section IV.

D. Fusion, Observability, and Sensor Health

A navigation architecture must combine estimates while accounting for their uncertainty. Chihabi and Ulrich’s angle-only relative navigation uses adaptive extended and unscented Kalman filters with nonlinear dynamics and measurement models [68]. This classical approach adjusts the estimator to the observations and provides a useful comparison for learned uncertainty models.

Learned front ends can supply features, poses, or uncertainty estimates to such filters. Separating perception from estimation makes it easier to trace an error to its source. A learned covariance must remain symmetric and positive semidefinite, and its effect on filter gains and consistency needs to be checked. Adaptation based on misleading observations can degrade the estimate, so update rules need their own rejection or rollback criteria.

Observability depends on the dynamics, measurements, and trajectory. Unconstrained scale and symmetric target configurations require additional information or explicit priors. Viewing geometry, motion, occlusion, and sensor redundancy therefore matter as much as image-level accuracy. Disagreement between independent sensors can reveal a problem, although common-mode errors may pass that check.

The reviewed navigation studies show two practical uses of learning: improving features within a geometric estimator and adapting perception to a new image domain. SPNv2 demonstrates that these choices can reduce pose error on a fixed benchmark; the lunar-navigation example shows how learned detections enter a temporal filter. Table II records the interfaces and evaluation settings. Comparisons still need a common target, image domain, error metric, and adaptation protocol, while timing measurements should include the processing stages used in operation.

IV. AI FOR GUIDANCE

Guidance selects a trajectory or reference action for the vehicle. Learning can approximate an existing solver, generate a policy through interaction, or initialize a numerical optimizer. Each role places different demands on the surrounding control

TABLE II
NAVIGATION METHODS: LEARNED MODULES, EVALUATION, AND REPORTED OUTCOMES

Method	Learned module and input	Evaluation setting	Reported outcome and scope
Sharma–D’Amico [25]	CNN components for known-target monocular pose	Image-based pose evaluation	Learned outputs are combined with geometric constraints; target knowledge matters
SPNv2 [53]	Shared encoder; optional ODR	SPEED+ synthetic/HIL; 25 PRISMA flight images evaluated on the ground	Lower mean lightbox pose error than style-augmented KRN; flight images excluded from ODR
Akida pipeline [58]	Quantized keypoint regression with ground-truth crops	Synthetic SPADES; physical V1 hardware; V2 cloud evaluation	Quantization affects accuracy; real event data and detection are outside this evaluation
Stereo CNN [55]	CNN-assisted pose from stereo images	Ground stereovision evaluation	Stereo geometry supplies information absent from a single camera
Line-segment pipeline [56]	Target and line detection in monocular images	Synthetic and Tango mock-up imagery	Learned features feed a geometric pose solution
Lunar crater navigation [67]	CNN detections feeding geometric matching and an EKF	Synthetic images; prototype descent with crater-database navigation down to about 20 km	Mean crater-center error below about 3 pixels; prototype horizontal and vertical position errors around 200 m
IONet (terrestrial analogy) [63]	Sequence model of inertial measurements	Ground motion data	Demonstrates terrestrial odometry; spacecraft transfer is a separate task

system: an approximate policy needs reliable tracking, while a learned initial guess needs a solver that can turn it into an acceptable trajectory.

A. Learned Guidance Laws

Sánchez-Sánchez and Izzo train neural networks on optimal landing trajectories to approximate feedback control [69]. The computation shifts from repeated onboard solution of an optimal-control problem to offline generation of demonstrations and network training. The original solver then supplies a reference for measuring the learned policy’s control error and trajectory cost.

Sprague et al. combine trajectory optimization, homotopy continuation, and imitation learning to represent policies across objectives and boundary conditions [70]. Their inverted-pendulum and spacecraft-transfer examples show how related optimal demonstrations can train a policy whose objective is changed online. Expert-demonstration methods have also been applied to spacecraft rendezvous and docking [71].

Single-step action accuracy is only part of this comparison. Small errors can move the vehicle away from the demonstrated trajectories, where later predictions become less reliable. Closed-loop tests with the same initial conditions, dynamics, constraints, and termination rules reveal whether a fast approximation retains the solver’s useful behavior.

B. Reinforcement-Learned Guidance

RL is useful when the policy must balance a sequence of decisions whose consequences are difficult to express as an immediate action label. Spacecraft applications include maneuvers near small bodies [72], rendezvous in cluttered environments [73], multi-impulse reconfiguration [41], and station-keeping in halo and low Earth orbits [23], [24]. The controlled variables and objectives differ across these problems. A policy for long-horizon orbit maintenance cannot be

compared directly with a short-range docking policy by a single fuel statistic.

Rotating-target docking introduces approach geometry and coupled translational and rotational motion. Oestreich et al. study six-degree-of-freedom (6-DoF) docking with RL [74]; Federici et al. examine adaptation during finite-thrust rendezvous [75]; and Shao et al. use an asymmetric actor–critic formulation for approaching maneuvers under disturbance [76]. These studies address different combinations of dynamics, observations, and training uncertainty.

Hovell and Ulrich separate learned guidance from low-level control [26]. Their policy outputs velocity commands, which a conventional controller tracks. After training in simulation, the policy runs on a Jetson TX2 in the SPOT planar air-bearing testbed, using state measurements from an external motion-capture system. Experiments demonstrate approach and docking with stationary and rotating targets, including an obstacle-avoidance case. The same trained policies achieve behavior similar to simulation despite differences in platform mass, thrusters, and delays. This is a useful demonstration of transferring learned guidance through a conventional tracking interface.

Inspection changes the objective: obtaining informative views may matter more than reaching a fixed terminal state. Capra et al. combine learned planning with MPC-based execution [77]; Brandonisio et al. integrate image processing with a closed-loop inspection GNC pipeline [78]. Here, the chosen path also changes the quality of later observations. Comparing inspection policies therefore requires a shared camera model, visibility conditions, information objective, and safety constraints.

Across these applications, RL offers a way to learn decisions over a maneuver or observation sequence. Its value depends on the task: reduced online computation, better coverage, and lower control effort are different benefits. Training cost, constraint violations, and failed maneuvers belong in the comparison alongside the selected performance metric.

C. Transformers for Optimization and Vision–Language Assistance

Transformer-based trajectory generation has a concrete role as an initializer for constrained optimization. The model predicts a state–control sequence, and a downstream solver refines that sequence under the mission’s dynamics and constraints. Thus the learned model can affect solution quality and convergence without being the sole authority for the commanded trajectory.

Takubo et al. extend the Autonomous Rendezvous Transformer (ART) to chance-constrained trajectory optimization for low Earth orbit (LEO) rendezvous [14]. They compare the same sequential convex programming (SCP) solver initialized by ART or by a convex boundary-value solution. ART generally reduces the required iterations; its more expensive initialization is offset by comparable or shorter total solution time. In one illustrated chance-constrained case, the final maneuver cost falls from 5.2892 to 4.7917 m/s in total Δv . The gains are strongest when the convex initial guess violates more constraints, while individual trajectories can have higher costs with ART. A post-hoc check selects whether to use the learned initial guess or a fallback before SCP.

Celestini et al. condition trajectory generation on obstacle configurations, initial and goal states, and performance requirements [15]. These inputs let one model provide initial guesses for varied scenes. Evaluation includes simulation and a physical free-flyer platform, where a controller tracks the optimized trajectory. The ground demonstration shows a practical division of work: learning supplies an initial sequence, optimization applies the trajectory constraints, and feedback handles execution.

Vision–language assistance occupies another part of the decision chain. Foutter et al.’s 2024 study adapts a vision–language model using visual-question-answer data derived from AI4Mars [79]. Its evaluation concerns spatial and terrain-related questions about static images. This could support observation interpretation or task planning; executing a resulting instruction would also require state estimation, planning, and control.

Reusable guidance models would need to accommodate changes in vehicle dynamics, sensing, actuation, and task objectives. The examples above establish more limited forms of reuse, such as varying a scene or trajectory requirement within a defined problem. Extending that range requires comparisons that include the downstream solver and controller.

D. Landing Guidance

Gaudet et al. use PPO to learn integrated guidance and control for 6-DoF planetary powered descent, evaluating landing accuracy, control effort, and robustness in simulation [36]. Iafrate et al. develop PPO policies for terminal landing of a reusable launcher’s first stage using 3-DoF and 6-DoF environments [37]. Both evaluate policies within specific vehicle models, allowing the effects of initial conditions, actuator limits, and disturbances to be studied.

Furfaro et al. investigate a more direct visual interface for vertical (1-D) and planar (2-D) lunar landing: neural networks

map camera imagery to thrust actions [80]. Training uses simulated images associated with optimized descent trajectories. Evaluation on images from held-out optimal trajectories measures thrust-level classification in both cases and thrust-direction regression in the planar case. This puts perception and control within the learned mapping, whereas the crater-based navigation approach in Section III supplies measurements to a separate estimator. Visual policies can thus reduce the number of explicit processing stages, but their closed-loop tests must also capture illumination changes, image errors, and delays. Choosing a safe landing site and reaching that site remain separate evaluation tasks.

E. Safety-Constrained Guidance

Safety can enter learning through the optimization problem or through a separate runtime mechanism. Constrained policy optimization addresses expected cost constraints during learning [81]. A control barrier function (CBF) instead describes a region in which a suitable controller can keep the system under stated dynamics and input conditions [82]. A reward penalty encourages behavior; a CBF constraint supplies a different, model-based form of protection.

Wijayatunga et al. combine RL with input-constrained CBFs (ICCBFs) in two stages [83]. The first adjusts class- $\mathcal{K}$ decay-related parameters while preserving the proven inner set C^* ; the second learns a residual barrier to recover additional states. In the simulated rotating-target docking test, the combined method reduces median control effort by 11.1% relative to the ICCBF baseline and succeeds in 95 of 100 cases, compared with 88 of 100 for the baseline. These results show a task-level benefit from learning the barrier parameters. Recovering more initial states also increases the upper tail of control effort, and five cases still fail. The proof within C^* and the measured recovery performance outside it therefore describe different parts of the result.

Runtime-assurance studies place a supervisor around a learned docking or inspection policy [27], [28], including demonstrations on aerial surrogate vehicles [29]. The supervisor’s protection depends on the region from which recovery is possible, the available actuator authority, and the time needed to intervene. Sampling, estimation error, and switching delays must be included when moving a continuous-time design into software.

The examples in Table III suggest that the learning interface matters as much as the choice of algorithm. Hovell and Ulrich demonstrate transfer by learning velocity commands for a conventional tracker; Takubo et al. improve a retained trajectory solver; and the ICCBF study improves control effort and recovery within its docking task. These are concrete benefits with different baselines, so each should be assessed together with the computation and control mechanisms it uses.

V. AI FOR CONTROL

A. DRL Attitude Control

Attitude control tests whether a learned policy can turn state estimates into useful actuator commands. Continuous-action methods such as deep deterministic policy gradient (DDPG),

TABLE III
GUIDANCE METHODS: LEARNING ROLE, CONSTRAINT HANDLING, AND EVALUATION

Approach	Learned module	Constraint handling	Evaluation and scope
Solver imitation [69], [70]	State-to-action approximation from optimized trajectories	Constraints enter the optimized demonstrations; learned actions approximate the solver	Simulated landing and transfer; tracking errors can move the vehicle away from training trajectories
Velocity-command RL [26]	Learned guidance velocity	Conventional low-level tracking controller	Simulation and planar air-bearing experiments; external motion-capture state input
Disturbed-approach RL [76]	Asymmetric actor–critic policy	Task formulation and reward terms	Simulation under the specified disturbance model
Station-keeping RL [23], [24]	Maneuver/control decisions	Mission-specific objectives and limits	Simulated halo-orbit and LEO tasks; different horizons and dynamics
Powered-descent RL [36], [37]	Integrated 6-DoF guidance and control	Reward design and vehicle-model limits	Simulation; landing accuracy and control effort depend on the tested dispersions
Visual lunar landing [80]	Images mapped to thrust actions	Training examples from optimized trajectories	1-D/2-D synthetic-image tests; thrust-level classification and, in 2-D, thrust-direction regression
RL with ICCBF [83]	Decay parameters and residual barrier	CBF-constrained quadratic program; invariant inner set	Simulated docking: 11.1% lower median effort and 95/100 successes versus baseline 88/100
RL with runtime assurance [28], [29]	Inspection policy	Supervisory safety mechanism	Simulation and aerial surrogate; protection depends on model and timing assumptions
Robust transformer warm-start [14]	Initial state–control sequence	Acceptance check, fallback, and chance-constrained SCP	Simulated LEO rendezvous; fewer solver iterations than convex initialization
Multimodal warm-start [15]	Scene-conditioned initial sequence	Subsequent constrained optimization	Simulation and physical free-flyer; scene variation within the studied task
Vision–language assistance [79]	Image-conditioned language response	No closed-loop control assurance evaluated	Static Mars-image question answering; task-level assistance

PPO, and soft actor–critic (SAC) provide the algorithmic basis [5]–[7]. Their performance depends on the available measurements, actuator limits, initial rates, disturbances, and sampling interval. A meaningful comparison measures pointing error, settling time, control effort, and constraint violations against a baseline tuned for the same task.

Willoughby et al. study a recovery task in which a satellite must regain useful Sun exposure despite uncertain attitude-control failures [84]. Failure randomization exposes the policy to impaired control authority during training, and simulated recovery is evaluated over the sampled fault distribution. LeLaR provides flight evidence for learned reaction-wheel control on InnoCube [16]. Its December 2025 experiments compare relative-attitude maneuvers with the satellite’s existing proportional–derivative (PD) controller: the learned policy meets the specified steady-state criterion more consistently, while the PD controller produces smoother commands. The PD gains were retained from the original mission, so this is an operational reference rather than a comparison of equally retuned controllers. Section VIII gives the maneuver results and measurement conditions. Recovery of a power-positive attitude, precise pointing, and long-term inertial stabilization impose different requirements on both sensing and control.

An RL policy may also supply a correction to an existing controller. HOPAS explored this residual approach around proportional–integral–derivative (PID) control for OPS-SAT, but its reported validation remained in simulation and ground testing [16]. Residual control preserves an interpretable nominal command and makes the learned contribution easier to constrain. Whether it preserves stability or safety depends

on how that correction enters the dynamics, not merely on retaining the nominal controller in software.

B. Learning-Based MPC and Adaptive Control

Learning and MPC can divide the problem between improved prediction or planning and explicit constraint handling [85], [86]. A learned dynamics model can reduce prediction bias; a learned cost or reference generator can improve task performance; the MPC optimizer then selects admissible inputs over a finite horizon. Constraint satisfaction follows only under the stated model-error bounds, initial feasibility, terminal conditions, and successful completion of the optimization. A learned model with unbounded errors can invalidate those conditions.

Capra et al. combine a DRL mapping planner with a convex MPC tracker for proximity operations around an uncooperative target [77]. The planner selects viewing trajectories, while MPC enforces the modeled collision-avoidance and actuation constraints. Under the same mapping task, the recurrent policy gives better reported training metrics and shorter training time than the transformer; the latter has smoother training trends. This comparison concerns the learned planner architecture. In the integrated simulation, adding MPC preserves mapping performance close to that of the learned guidance while providing explicit constraint handling. Processor-in-the-loop tests of the recurrent implementation on a Raspberry Pi 4 put guidance and control computation below 0.12 s per update. These results support a computationally practical division between task planning and tracking; the evaluation uses simulated imagery and dynamics.

Ren et al. place a learned maneuver schedule within decentralized MPC for spacecraft swarm reconfiguration [42]. Their reachable-set construction and Lyapunov analysis establish recursive feasibility and asymptotic stability for the modeled cooperative-reconfiguration problem under its assumptions.

For a residual controller, $u = u_0 + \Delta u_\theta$, a small correction can still drive a state toward a constraint boundary. A defensible design therefore relates the permitted residual to a robust stability margin, filters it through a feasible safety constraint, or uses a supervisor with a proven recoverable set and a suitable backup controller [82], [87]. Sliding-mode control (SMC) with a learned adaptation law likewise depends on disturbance bounds and sufficient switching authority [88]. Runtime assurance for attitude maneuvers explicitly addresses simultaneous operational constraints [89], while learning-based time-shift governors regulate a rendezvous reference before it reaches the tracker [90]. Classical robust MPC [91] and offline Koopman-expectation guidance [92] provide useful comparisons for handling uncertainty. Using either as a backup requires explicit switching conditions and a set of states from which recovery is possible.

C. Coupled Space-Robot Control and Capture Planning

Space manipulators couple arm motion to base translation and rotation. In a free-floating system, momentum exchange can disturb the spacecraft attitude even when the arm follows its commanded trajectory; free-flying systems can use base actuators to compensate at a propellant or momentum-storage cost [93]–[95]. This coupling makes a task-space tracking score insufficient on its own. Base disturbance, joint limits, actuator saturation, and target-relative motion also determine whether a maneuver is usable.

The reviewed learning methods address several parts of this problem. Xie et al. adapt sliding-mode control for a free-floating manipulator [88]; D’Ambrosio et al. use DRL for redundant-manipulator guidance [35]; and Wei et al. plan approaches to a tumbling target while penalizing base disturbance [96]. The latter penalty encourages a desired behavior but is not a hard bound on base motion.

Crain et al. demonstrate the full sequence of tracking, capture, and arm retraction on a planar air-bearing testbed [33]. A CNN-based stereo-vision system estimates the target pose, and DRL controls the chaser’s relative pose. The arm follows an optimal deployment trajectory computed offline by Gauss pseudospectral optimization. Successful physical capture and retraction show that these learned perception and guidance modules can operate with a classical arm planner in one integrated maneuver. The experiment establishes that capability for the tested planar system; three-dimensional impact, flexible-body dynamics, and in-orbit stabilization after capture require a broader model and test environment.

D. Stability and Safety Verification

Stability and safety are related but distinct. A Lyapunov argument concerns convergence or boundedness around an equilibrium; a control barrier function concerns forward invariance of a specified admissible set [82]. For spacecraft,

either argument must account for the state estimator, actuator saturation, sampling, and the disturbances included in the model. A valid continuous-time certificate cannot be transferred unchanged to a delayed or quantized implementation.

Neural Lyapunov methods learn candidate functions and then check their defining inequalities. Physics-informed constructions and LyZNet combine learning with verification of a specified region of attraction [97], [98]. Hyperlyve instead exploits the piecewise-affine structure of a single-hidden-layer candidate network with rectified linear unit (ReLU) activations on a compact domain [99]. Its theoretical procedure is sound and complete under its stated conditions, including the required regionwise global optimization. The reported implementation uses simplicial homology global optimization (SHGO) with sequential least squares programming (SLSQP) for the nonlinear decrease test; convergence as sampling becomes dense does not make every finite numerical run a formal certificate. Verisig follows another route, encoding supported neural-network activations as a hybrid system and applying closed-loop reachability analysis [100]. Its conclusions are tied to the encoded plant, network class, initial set, and time horizon. Extending these tools to coupled 6-DoF spacecraft dynamics requires both computational scalability and a model that includes the flight-relevant constraints.

Table IV separates the learned module, the constraint mechanism, and the environment in which evidence was obtained. Training penalties, domain randomization, runtime intervention, and mathematical verification serve different purposes and should not be treated as interchangeable forms of assurance.

The representative studies show two practical contributions of learning: selecting a task-dependent reference for a constrained tracker, and generating actuator commands that meet a defined pointing objective. Capra’s architecture comparison and LeLaR’s flight reference provide evidence for these contributions within their respective tasks; Crain adds physical evidence that learned modules can be integrated through capture and retraction. Across these cases, performance depends as much on the estimator, tracker, and actuator interface as on the policy architecture.

VI. AI FOR FAULT DETECTION AND HEALTH MANAGEMENT

Fault detection, isolation, and recovery (FDIR) form a sequence of decisions: recognize a deviation, identify its likely source, select a response, reconfigure the system, and confirm that the response worked [101], [102]. Learning has been studied most extensively at the detection stage. This section therefore distinguishes telemetry monitoring from the narrower set of studies that connect learned outputs to fault-tolerant control or recovery actions.

A. Telemetry Anomaly Detection

Prediction and reconstruction models learn regularities in nominal telemetry and flag departures from them. Hundman et al. combine long short-term memory (LSTM) prediction with nonparametric dynamic thresholds for SMAP and MSL

TABLE IV
REPRESENTATIVE LEARNING-BASED CONTROL STUDIES AND THEIR VALIDATION SCOPE

Study	Learned module / task	Constraint or verification mechanism	Validation environment and interpretation
LeLaR [16]	Reaction-wheel attitude policy	Command limits; independent cage disables control on violation	In-orbit relative-attitude maneuvers; more consistent criterion attainment than the existing PD reference; Section VIII
HOPAS [16]	Residual correction to PID	Limited learned contribution	Simulation and ground tests; no reported on-orbit policy execution
Sun-search recovery [84]	RL reorientation after ADCS impairment	Failure randomization during training	Simulation; Sun-exposure recovery evaluated under sampled ADCS impairments
Reinforced MPC [77]	DRL planner with convex MPC tracker	State and input constraints in the tracker	Recurrent versus transformer mapping policies; Raspberry Pi 4 recurrent implementation below 0.12 s
Swarm MPC [42]	Learned maneuver schedule	Reachable-set collision regions; conditional feasibility and stability proof	Numerical cooperative-reconfiguration scenarios
Adaptive RL-SMC [88]	Free-floating manipulator control	Sliding-mode analysis under modeled uncertainty bounds	Numerical coupled base-arm dynamics
Capture planning [96]	RL trajectory planner	Base-disturbance term in reward	Simulation of a tumbling target; reward does not certify a disturbance bound
Capture and retraction [33]	CNN stereo vision and DRL relative-pose tracking	Offline optimal arm deployment; classical and learned modules integrated	Successful physical capture and retraction on a planar air-bearing testbed
Neural Lyapunov learning [97], [98]	Candidate certificate and attraction region	Verification of Lyapunov conditions over a declared domain	Dynamical-system benchmarks; conclusions depend on the model and verification backend
Hyperlyve [99]	Verification of shallow ReLU candidates	Hyperplane regions; global decrease test in theory	Benchmarks; finite SHGO/SLSQP implementation differs from the theoretical procedure
Verisig [100]	Neural-controller closed-loop verification	Hybrid encoding and reachability	Benchmarks with supported activations, plant models, and bounded initial sets

data [103]. The attraction is detection of changes that remain inside conventional engineering limits. In ground operations, ATHMoS explores neural feature extraction, anomaly detection, and telemetry prediction as additions to its monitoring framework [104]; Yairi et al. use probabilistic clustering and dimensionality reduction for satellite housekeeping data [105]. These approaches assist interpretation of telemetry and do not, by themselves, establish onboard recovery authority.

Hundman’s comparison also separates the predictor from the alert rule. With the same LSTM predictions, nonparametric thresholding with pruning achieves 87.5% precision and 80.0% recall over the combined SMAP/MSL anomaly sequences. The Gaussian-tail threshold at $\epsilon_{\text{norm}} = 10^{-4}$ achieves the same precision but 66.7% recall. Removing pruning from the nonparametric rule raises recall to 84.8% while reducing precision to 48.9% [103]. These event-level scores count an anomaly as detected when any predicted interval overlaps its label, without rewarding earlier detection. The five-day evaluation windows contain anomalies, so their relatively high event rate makes the reported precision optimistic for routine operations. The comparison shows how the alert rule changes the balance between detecting events and limiting false alerts with the same predictions.

More recent models represent temporal and cross-channel dependence in different ways. OmniAnomaly uses a stochastic recurrent model [106]; Anomaly Transformer scores asso-

ciation discrepancies [107]; and TranAD uses transformer-based reconstruction with self-conditioning [108]. MAG stands for *maximum information coefficient attention graph network*. It combines a graph representation of relationships between channels, correlation and attention mechanisms, and an LSTM for temporal features [109]. Fault-specific classifiers, such as the attitude-sensor CNN studied by Gallon et al., connect detection outputs to recovery logic using simulated sensor faults [110]. Table V lists specific methods and their evidence bases; the broader survey by Fejjari et al. provides additional coverage [11].

B. Datasets and Operational Evaluation

Real telemetry provides essential operating context, but labels and evaluation protocols determine what a benchmark score means. SMAP/MSL contains labeled anomaly intervals for selected channels [103]. OPS-SAT-AD, first released as a dataset in 2024 and described in a 2025 data paper, supplies expert-labeled telemetry segments and baseline implementations [111], [112]. Its nine channels cover a three-axis magnetometer and six photodiodes. This is a useful basis for reproducible detection studies, not a representative sample of every subsystem or fault mode. Table VI separates these data resources from the algorithms that use them.

Herrmann et al. show why strong claims based on anomaly-detection benchmarks require scrutiny: evaluation choices can

TABLE V
TELEMETRY MONITORING METHODS: OUTPUT, EVALUATION SETTING, AND OPERATIONAL ROLE

Method	Model and output	Evaluation setting	Role and scope
LSTM monitoring [103]	Prediction residual with dynamic threshold	Retrospective SMAP/MSL telemetry	Event-overlap precision/recall: pruning 87.5%/80.0%; Gaussian-tail 87.5%/66.7%
ATHMoS neural module [104]	Learned features, anomaly scores, and prediction	Experimental module for GSOC ground monitoring	Supports engineers in interpreting telemetry
Probabilistic clustering [105]	Reduced-dimensional nominal-data model	Satellite housekeeping data	Health monitoring; operational context needed to interpret deviations
OmniAnomaly [106]	Stochastic recurrent reconstruction model	Multivariate time-series benchmarks	General anomaly-detection method evaluated offline
Anomaly Transformer [107]	Attention-based association discrepancy	Time-series benchmarks including SMAP/MSL	Anomaly scoring; thresholds determine alert burden
TranAD [108]	Transformer with self-conditioning	Multivariate time-series benchmarks	Offline detection comparison; no flight control authority demonstrated
MAG [109]	Graph, correlation, attention, and LSTM	Four spacecraft-telemetry datasets	Unsupervised scoring of cross-channel and temporal anomalies
Attitude-sensor CNN [110]	Multichannel fault classification	AOSE simulated sensor data with injected stuck-value faults	Cross-sensor detection; persistence-filtered reaction triggering in simulated FDIR

TABLE VI
PUBLIC FLIGHT-TELEMETRY RESOURCES USED FOR ANOMALY DETECTION

Resource	Data and interpretation
SMAP/MSL [103]	Selected telemetry channels and labeled anomaly intervals; retain the supplied temporal structure when evaluating prediction-based detectors.
OPS-SAT-AD [111], [112]	Nine sensor channels, labeled segments, derived features, and 30 baseline methods in the data paper; segment classification differs from continuous alert generation.

overstate the advantage of complex models over simple alternatives [113]. In particular, point adjustment can credit an entire anomalous interval after detecting only part of it. Results should therefore state the scoring unit, threshold-selection rule, treatment of incomplete labels, and any interval adjustment. Thresholds tuned on test anomalies do not measure prospective operating performance.

For mission use, chronological splits and tests on held-out operating modes help expose drift and leakage. Overlapping windows from the same event should stay in one partition. A detector should be compared with engineering limits and simple statistical baselines using event recall, false alerts per unit operating time, and detection delay. Fault isolation additionally needs a confusion matrix over the stated fault set; recovery needs success rate, response time, and the resource cost of the intervention. These quantities answer different questions, so a single F1 score cannot summarize the full FDIR chain.

C. From Detection to Isolation and Recovery

An anomaly score says that a pattern differs from the learned reference. It does not establish whether the cause is a sensor fault, a maneuver, a mode transition, or a changed environment. Isolation requires additional evidence, such as sensor redundancy, model residuals, command history, or a classifier trained on specified fault classes [101], [102]. A practical interface passes the affected channels, timing, persistence, and uncertainty to the decision logic. Ground systems can route this information to an operator; onboard systems need predefined authority and response conditions.

Gallon et al. connect a multichannel CNN to the simulated FDIR system of the Astrone KI small-body exploration concept [110]. The AOCS Offline Simulation Environment (AOSE) generates trajectories, accelerometer and inertial-measurement signals, injected stuck-value faults, and the Packet Utilization Standard (PUS) reaction logic. Cross-sensor information helps distinguish a genuinely stuck signal from a legitimate constant reading, including faults that remain inside a univariate limit check. The CNN's fault flags pass through persistence checks before triggering a response. The study evaluates missed faults, delays, false-positive durations, and reaction precision and recall; longer persistence suppresses short false detections but delays responses. This provides evidence about the combined detector and reaction logic in software simulation. After reconfiguration, the remaining measurements and actuators must still be checked against the selected operating mode.

Fault-tolerant control addresses a complementary problem. The Sun-search policy of Willoughby et al. attempts recovery under sampled ADCS impairments without explicitly diagnosing each failure [84]. Maintaining useful behavior can buy time for diagnosis, but compensation may also hide degradation. Coupling the controller to explicit health reporting allows operators or supervisory logic to distinguish successful

adaptation from continued operation with reduced margins. Recovery authority should be evaluated using the combined detector, decision logic, and controller, including delayed or incorrect alarms.

D. Prognostics and Degradation-Aware Planning

Prognostics estimates future degradation rather than an immediate deviation. Richardson et al. demonstrate Gaussian-process forecasting of lithium-ion battery state of health from terrestrial cell-aging data, estimating capacity loss together with its uncertainty [114]. Transfer to a mission requires validation against its cell chemistry, cycling pattern, temperature range, and aging history. Prediction intervals also need calibration on held-out degradation trajectories; a narrow interval is useful only if its coverage is reliable.

State of health, state of charge, and remaining useful life serve different planning decisions. Capacity degradation can change how much energy a battery may safely supply, whereas state of charge describes its current charge relative to a capacity estimate. A remaining-life prediction is conditional on future use and on the chosen end-of-life criterion. These estimates can inform reduced-duty operation, load scheduling, and maintenance planning, provided that the planner retains margins for uncertainty and the estimate is refreshed as new measurements arrive.

The telemetry studies support learning primarily as a way to recognize temporal and cross-channel patterns that fixed limits miss. Hundman's results show that the alert rule can change precision and recall substantially even with an unchanged predictor, while Gallon's simulation shows why the persistence and reaction logic belong in the evaluation. For mission use, the detector should be assessed together with the rules that turn its outputs into alerts and recovery actions, using both alert burden and response delay.

VII. AI FOR THERMAL CONTROL AND ENERGY MANAGEMENT

Thermal and power states constrain which actions a spacecraft can sustain. A pointing maneuver changes solar-array illumination and surface exposure; a perception workload consumes electrical power and releases heat; a heater preserves a component's operating range while drawing down the battery. Conventional thermal design and control already address changing environments through passive design, heaters, thermostats, and other actuators [17], [18]. Learning is most useful where it improves an uncertain model or reduces the cost of repeated prediction. The evidence reviewed here is strongest for thermal surrogates and state estimation, with a smaller basis for autonomous closed-loop thermal control.

A. Thermal Models, Estimation, and Flight Data

A reduced thermal network makes the main dependencies explicit. For node i , one convenient balance is

$$C_i \dot{T}_i = Q_i^{\text{diss}} + Q_i^{\text{abs}} + Q_i^{\text{exch}} - Q_i^{\text{emit}}, \quad (1)$$

where C_i is heat capacity, Q_i^{diss} is internal dissipation, Q_i^{abs} is absorbed external radiation, Q_i^{exch} is net conductive and

radiative exchange with other spacecraft nodes, and Q_i^{emit} is radiation emitted to the external environment. Solar irradiance is about 1361 W/m^2 at 1 AU [115], but absorbed power also depends on projected area, absorptance, shadowing, and attitude. Planetary infrared radiation and reflected sunlight depend on geometry and view factors. Heat rejection therefore cannot be represented by a universal cold-sink temperature alone.

High-fidelity thermal networks support design, test correlation, and operations. Their repeated solution across many parameter choices or planning horizons can be expensive. Tanaka and Nagai combine physics-informed learning with proper orthogonal decomposition (POD) to construct steady-state spacecraft thermal surrogates [116]. Yamashita et al. extend the approach to transient system-wide temperature fields using POD and recurrent networks [117]. Against the conventional thermal mathematical model, their numerical experiments report an average prediction error of 1.53 K at 0.23% of its computational cost. The benefit is rapid prediction of the full temperature field for repeated analysis. Operational planning additionally depends on peak error near a component limit and error accumulated over the prediction horizon, which an all-node average does not describe.

State estimation solves a different problem: infer temperatures that are not directly measured. Tanaka and Nagai combine a thermal mathematical model with sparse temperature observations in a physics-informed estimator [118]. Their numerical study uses a 100-node small-satellite model, three model-error cases, and variations in sensor placement and number. The result shows how observations can correct an uncertain temperature field. It also makes sensor coverage part of the learning problem, rather than assuming that every model state is available for training or feedback.

Ground tests and flight data provide complementary checks. Jang et al. construct lunar-orbiter temperature surrogates by combining principal component analysis (PCA) with radial-basis-function or deep neural networks [119]. A parameterized full-order thermal model supplies the training responses. Evaluation covers both cold and hot thermal-balance conditions and compares predicted electronic-equipment temperatures with measurements from the physical thermal-vacuum test. The full-order model provides the numerical reference, while the measured temperatures test whether the reduced representation remains useful for the actual spacecraft. Their later study selects informative simulations through active learning and adapts the model with Danuri's measured orbital temperatures through transfer learning [120]. This extends prediction from ground-test conditions to the flight environment within a ground-based virtual-metrology workflow. Table VII separates these modeling and estimation roles.

B. Predictive Thermal Control

Fast prediction could help a controller decide when to heat a mechanism, defer a computational burst, or change a payload duty cycle. A surrogate supplies future temperatures to the decision process; the controller still needs actuator limits, thermal margins, and an estimate of model error. Learning-based MPC offers methods for incorporating such predictions into

constrained control [85]. Its usefulness depends on whether the surrogate is accurate over the states and commands considered by the optimizer, including conditions that differ from the training data.

The main comparison is with an existing thermal network, reduced-order observer, or conventional predictive controller under the same mission plan. Relevant outcomes include the largest temperature-limit excursion, heater energy, computation time, and the number of tasks completed. A lower prediction error need not reduce heater use, and a faster model need not improve closed-loop reliability. Before transferring a ground surrogate into the control loop, processor timing, numerical precision, sensor outages, and responses near thermal limits have to be evaluated together. A bounded prediction-error allowance can be used to tighten the constraints; outside that validated range, the supervisor can reduce the workload or retain the established thermal-control mode.

C. Energy Forecasting and Battery Health

Energy planning combines predicted generation, scheduled loads, and usable storage. Illumination geometry and eclipse timing are largely physical prediction problems. Learning can contribute estimates of uncertain load profiles, battery capacity, or model residuals, but it should be compared with those physical baselines. Gaussian-process battery forecasting provides a useful example of degradation prediction with uncertainty, evaluated on terrestrial cell-aging data [114]. Spacecraft application requires the temperature and cycling conditions discussed in Section VI. Current state of charge and long-term state of health should remain separate inputs to the planner.

Coupled power and orbital control is already a classical optimization problem. Ding et al. study ultra-low Earth orbit maintenance using MPC with power availability, battery behavior, eclipse, and thrust scheduling [121]. This supplies a physical optimization baseline for joint resource and orbit management. A learned load or degradation model can be assessed by its effect on that planner's feasible maneuver schedule under the same energy margins.

D. Coupling GNC, Computing, and Thermal Limits

Temperature-aware computing provides a general basis for adapting workload or clock rate to thermal conditions [122]. On a spacecraft, the scheduler must also account for radiator capacity, battery limits, and the deadlines of navigation and control. Reducing processor power can extend a safe observation window, but delaying perception may degrade a pose estimate or miss a maneuver deadline. Energy-efficient hardware [123] and workload scheduling address different parts of this tradeoff.

Consider a conceptual inspection maneuver in low Earth orbit. Tracking a target may reduce solar-array illumination while image processing raises processor dissipation. A joint prediction uses the planned attitude and orbit to calculate available electrical generation and external heat loads, and the workload schedule to calculate power consumption and

internal heating. Over a short planning window, a simplified storage model is

$$\begin{aligned} \dot{E}_b &= \eta_c P_c - P_d / \eta_d, \\ P_{\text{solar}} + P_d &= P_{\text{load}} + P_c + P_{\text{curt}}, \\ E_b(0) &= E_0, \quad E_{\min} \leq E_b(t) \leq E_{\max}, \\ 0 \leq P_c &\leq \bar{P}_c, \quad 0 \leq P_d \leq \bar{P}_d, \\ P_c P_d &= 0, \quad 0 \leq P_{\text{curt}} \leq P_{\text{solar}}. \end{aligned} \quad (2)$$

Here E_b is stored energy and the estimated initial value satisfies $0 \leq E_{\min} \leq E_0 \leq E_{\max}$. Charging power P_c and discharging power P_d are measured at the bus; constant efficiencies $0 < \eta_c, \eta_d \leq 1$ convert them to changes in stored energy. The fixed bounds represent the usable storage and power limits over this window, and $P_c P_d = 0$ prevents simultaneous charging and discharging. P_{solar} denotes the available solar-electric power, while P_{curt} is the part not admitted to the bus. Self-discharge and capacity aging are neglected over the maneuver.

The same task schedule supplies inputs to both balances. Computing, actuator, and heater commands determine P_{load} together with the other bus demands. Equipment dissipation and losses in the battery and power converters enter Q_i^{diss} at the corresponding thermal nodes. The planned attitude and orbit determine both P_{solar} and the absorbed external heat Q_i^{abs} .

This formulation permits excess generation to be curtailed when the battery is full and prevents continued discharge below the energy reserve during eclipse. Coupled with (1), it shows why maneuver feasibility depends jointly on temperature, energy, pointing, and processing deadlines. A learned thermal surrogate or load predictor could accelerate evaluation of candidate schedules; a constrained planner would then decide whether to shorten the imaging sequence, defer a noncritical inference task, or select another viewing interval. Holding the inspection objective and resource margins fixed would reveal whether the faster prediction enables a useful scheduling improvement. The example motivates the architecture and deployment choices discussed in Section X.

E. AI-Assisted Thermal Design

Thermal design offers an earlier entry point because candidate outputs can be checked with established analysis and test procedures before flight. NASA's 2024 SBIR solicitation identifies automated model geometry and insulation-performance prediction among the research needs for AI-assisted thermal-control design [124].

A practical research workflow could learn geometry simplification, parameter estimation, or the selection of expensive simulation cases. Each step needs a check appropriate to its output: conservation and connectivity for a generated network, held-out test cases for a surrogate, and thermal-balance data for model correlation. NASA RP-1121 supplies measured solar absorptance and thermal emittance for coatings [125]. These material properties can support surface modeling, whereas effective multilayer-insulation performance also depends on seams, penetrations, layer compression, and

installation. Coating data alone cannot train or validate that insulation model. The state-estimation and lunar-orbiter studies above provide more concrete evidence for combining physical models with measurements than a fully automated end-to-end design pipeline [118]–[120].

F. Operating Across Thermal Regimes

A model trained on a narrow orbital temperature range may extrapolate poorly during prolonged eclipse, changed attitude, a high-duty payload campaign, or component degradation. On-line correction is therefore useful only when the measurements can identify the changing parameters and the update preserves the required physical and operational constraints. Unobserved nodes and new heat paths can remain uncertain even after measured temperatures fit well.

The clearest demonstrated gain is reduced prediction cost: Yamashita’s surrogate approximates a full thermal analysis cheaply, while Jang’s test and flight-data studies connect such reduced models to measured spacecraft temperatures. These results support repeated analysis, model correlation, and ground planning. Translating that gain into lower heater energy or longer observation windows depends on the controller and resource constraints, which are separate from surrogate accuracy.

VIII. FROM SIMULATION TO ORBIT: SIM2REAL AND THE LeLaR CASE

A. Building Evidence Across Test Environments

Simulation, processor tests, physical facilities, and flight answer different validation questions. Software-in-the-loop tests permit large sweeps of initial conditions, disturbances, and failures. Processor-in-the-loop tests measure timing and numerical behavior on a specific computing platform. Hardware-in-the-loop and facilities such as optical testbeds [12] and the EPOS robotic simulator [126] expose sensor, actuator, and integration effects that a numerical model can miss. Flight tests then exercise the complete environment, usually over a much smaller set of conditions. These activities support one another; passing them in sequence is not by itself a proof that all operational hazards are covered.

Domain randomization varies uncertain parameters during training [127], [128]; simulation adaptation uses measured behavior to refine the training environment [129]. Both depend on choosing discrepancies that matter to the intended task [130]. A useful validation plan records the assumed range of each parameter, the evidence used to set it, and the tests that challenge the range. Emerging frameworks such as MMEDR aim to connect multi-view simulation, digital twins, and physical rendezvous experiments [131]. Its reported status includes RL guidance under development and progress toward integrated experimental validation, so it illustrates a test-framework direction rather than an already demonstrated pipeline for correcting flight policies.

B. Engineering Anatomy of LeLaR

LeLaR provides a concrete example of staged training and flight integration [16]. Its base policy contains reaction-wheel

and magnetorquer branches; post-training of the reaction-wheel subnetwork produces the flight policy. The observation vector remains 39-dimensional, while the action vector changes from six commands in base-policy training to three in the flight policy. Ground sensor measurements and post-launch calibration improve the simulator; policy training remains on the ground.

The first maneuver on 30 October 2025 used only the reaction-wheel subnetwork of the base-agent. Later December experiments evaluated the flight-agent and the existing PD controller. Control runs at 1 Hz, and the reported attitude is relative attitude obtained by integrating gyro measurements, without an accurate absolute attitude reference. Experiments are limited to 15 minutes. The first maneuver reached the paper’s criterion of less than 1° per axis for at least 15 s after 114 s.

The December tests provide two useful references [16]. In the repeated-maneuver sequence, all seven flight-agent maneuvers reached that criterion, compared with three of seven maneuvers using the existing PD gains. The corresponding flight-agent simulations settled in 14–38 s, whereas the flight maneuvers took 44–96 s. The main outcome is therefore successful steady-state transfer despite slower transients in orbit. PD retained its original mission tuning and produced smoother commands; different initial rates and actuator behavior also limit direct performance rankings. The reported attitude errors remain relative to the integrated gyro estimate.

The independent safety cage monitors actuator and body-rate limits. A violation disables the active controller, enters idle, and subsequently turns the wheels off. This response relies on InnoCube remaining power-positive while tumbling. Flight telemetry revealed unmodeled wheel startup delays of about 20 s and speed-measurement jumps around 185 rpm [16]. Continued operation through those events demonstrates tolerance to the observed hardware behavior; protection coverage depends on the cage’s monitored limits and the mission’s safe response.

LeLaR shows how policy training, sensor calibration, flight software, and independent monitoring work together. A new mission would need a response suited to its own operating conditions: idle operation that is acceptable for this CubeSat could be unsuitable near another spacecraft or when survival requires active pointing. Likewise, a deployed policy update changes the controlled system and must be evaluated with its estimator, actuator behavior, and supervisory limits. LeLaR’s released maneuver telemetry allows other teams to examine the achieved performance and the conditions in which it was measured [16].

C. Tracking Sim2Real Discrepancies

Discrepancies should be recorded by source and by the evidence for their mitigation. Actuator delays, dead zones, saturation, and quantization affect which commands are realized. Calibration and dynamics randomization are useful tools, but a delay outside the trained range still calls for a dedicated closed-loop test. Sensor noise, bias, dropouts, and estimator resets affect what the policy observes. Training with the actual

TABLE VII
SPACECRAFT THERMAL LEARNING STUDIES: FUNCTION AND EVIDENCE

Study / function	Learning method	Validation data or environment	Supported use and boundary
Tanaka and Nagai [116]: steady state	Physics-informed learning with POD	Spacecraft thermal-model numerical experiments	Accelerated ground thermal analysis across sampled model conditions
Tanaka and Nagai [118]: state estimation	Physics-informed model and sparse observations	Numerical 100-node satellite; three model-error cases	Estimation of unmeasured temperatures; dependence on sensor coverage
Yamashita et al. [117]: transient prediction	POD and recurrent neural network	System-wide thermal simulations; conventional thermal model as reference	Average prediction error 1.53 K at 0.23% of the reference computational cost
Jang et al. [119]: test-condition prediction	PCA with radial-basis or deep neural models	Simulation-trained; checked against measured hot/cold thermal-balance data	Reduced-order temperature prediction with numerical and physical test references
Jang et al. [120]: flight-data adaptation	Active learning and transfer learning	Thermal simulations and measured Danuri orbital temperatures	Ground virtual metrology adapted to the observed flight environment

estimator interface avoids granting the simulated policy state information that flight hardware cannot provide.

Dynamic discrepancies include inertia errors, flexibility, slosh, and changed contact conditions. Their effect depends on the controller bandwidth and on whether the robust or adaptive design covers the resulting uncertainty. Appearance discrepancies include illumination, surface reflectance, and camera characteristics; rendering, randomization, and facility imagery address complementary parts of this gap [12], [127]. For each observed discrepancy, the record should distinguish what was modeled during training, what appeared only in testing, and which intervention was actually exercised. Robustness observed in one maneuver is a test result; a claim that a specific training choice caused it requires an appropriate comparison or ablation.

D. Selected In-Space Autonomy and Learning Milestones

Fig. 2 selects six demonstrations between 1999 and 2025 and distinguishes symbolic autonomy, classical GNC, and learned inference or control. Remote Agent commanded Deep Space 1 in 1999 using onboard planning, execution, and model-based diagnosis [132]. EO-1's Autonomous Sciencecraft Experiment demonstrated autonomous science planning and execution in 2004 [4]. These predecessors established task-level autonomy with mission-specific command authority.

Learned payload analysis also predates 2020. EO-1 ran learned cloud classifiers and salience-based novelty detection during November 2016–March 2017 [133]. The later Φ sat-1 cloud-screening demonstration used onboard AI for Earth-observation data selection [134]. DART's 2022 autonomous optical terminal guidance is a classical GNC milestone [135]; LeLaR adds learned attitude control in 2025. These selected events do not define a single maturity ladder: the authority to prioritize imagery differs substantially from the authority to command actuators.

Table VIII broadens the record and uses the year of the stated capability demonstration. ETS-VII's automatic rendezvous and docking occurred in 1998; its space-robot experiments provide a separate reference for classical robotic

control [136], [137]. RemoveDEBRIS performed net and vision experiments in 2018 and the harpoon experiment in 2019; its vision experiment acquired data in space for processing on the ground [138], [139]. OPS-SAT demonstrated neural inference on a reconfigurable FPGA and later reprogrammed the implementation in flight [140], [141]. Φ sat-2 entered its science phase after commissioning its onboard applications in 2025, following launch in 2024 [142]. The table records each demonstration's function and control authority.

The flight record shows progress along several distinct paths: learned payload processing, reconfigurable inference, and learned actuator control. LeLaR adds evidence that a simulation-trained policy can meet a relative-attitude objective on real hardware, while its slower flight transients show why simulation alone cannot set maneuver timing. Flight validation is most informative when it records both the achieved task and the discrepancies that the deployed system had to accommodate.

IX. ONBOARD COMPUTING, VERIFICATION, AND ASSURANCE

A. Compute and Energy Constraints

An onboard model must meet a mission's execution-time, memory, power, and thermal limits on its intended processor. Processor selection also depends on radiation tolerance and fault handling [143], [144]. These requirements apply to the complete processing chain, including sensor preprocessing, data transfers, inference, and supervisory checks. A useful hardware benchmark therefore reports both model performance and the resources needed to produce an output within the mission deadline.

Neuromorphic processors illustrate the importance of workload-specific measurements. For the radio-resource-management workload studied by Lagunas et al., the reported energy-per-task ratio between a CNN on a Xilinx VCK5000 and a spiking neural network (SNN) on Loihi 2 exceeded 10^2 [145]. The ratio depends on the task, devices, latency, and batch size; the Loihi 2 measurements used Intel's remote research infrastructure. A GNC implementation would need

TABLE VIII
SELECTED IN-SPACE DEMONSTRATIONS, CLASSIFIED BY FUNCTION AND CONTROL AUTHORITY

Mission / experiment	Year	Method category	Demonstrated function	Authority or evidence boundary
ETS-VII [136], [137]	1998	Classical robotics and GNC	Automatic rendezvous and docking; space-robot experiments	Actuator control; not a learned-control demonstration
DS1 Remote Agent [132]	1999	Symbolic, model-based autonomy	Planning, execution, diagnosis, and recovery	Spacecraft commands during a defined experiment
EO-1 ASE [4]	2004	Symbolic planning and execution	Autonomous science planning and response	Observation scheduling and task execution
Orbital Express [30]	2007	Classical autonomous GNC	Rendezvous, capture, and servicing operations	Closed-loop mission autonomy; no learned module asserted
EO-1 data analysis [133]	2016–2017	Learned cloud classifiers; statistical novelty detection	Onboard analysis of Hyperion images	Payload-data products; not attitude control
RemoveDEBRIS [138], [139]	2018–2019	Capture hardware and classical vision	Net (2018), vision data (2018), harpoon (2019)	Released or deployed test targets; vision processing on ground
Φsat-1 [134]	2020	Learned image analysis	Onboard cloud screening	Payload-data selection
OPS-SAT CIAR [140]	2021	Neural-network inference	In-orbit inference on a reconfigurable FPGA	Experimental payload processing
DART SMART Nav [135]	2022	Classical optical GNC	Autonomous terminal targeting and impact	Direct terminal maneuver commands
OPS-SAT inference update [141]	2023	Neural-network implementation	In-flight FPGA reprogramming for inference	Payload software/hardware update; not onboard policy learning
Φsat-2 [142]	2025	Multiple learned payload applications	Commissioned onboard Earth-observation processing	Payload applications; 2024 was the launch year
LeLaR [16]	2025	Learned reaction-wheel control	Short-duration relative-attitude maneuvers	Independent cage: disable, idle, then wheels off

1999**Deep Space 1**

Remote Agent
Onboard planning and execution

SYMBOLIC AI**2004****EO-1 ASE**

Science planning
Autonomous science operations

SYMBOLIC AI**2016-2017****EO-1**

Cloud classification
Cloud and novelty analysis in orbit

LEARNED ANALYSIS**2020****PhiSat-1**

Cloud screening
Onboard image filtering

LEARNED PAYLOAD**2022****DART**

Terminal guidance
Optical tracking and impact

CLASSICAL AUTONOMY**2025****LeLaR / InnoCube**

Attitude control
Learned commands in orbit

LEARNED CONTROL

Fig. 2. Selected in-space autonomy and learning demonstrations, 1999–2025. Dates identify the stated capability demonstrations, and categories distinguish symbolic autonomy, classical GNC, and learned processing or control. The cards are ordered chronologically rather than spaced to a proportional time axis [4], [16], [132]–[135].

the same comparison with its own inputs and update rate. The Akida pose-estimation pipeline offers a closer application example: physical AKD1000 measurements show lower power and lower throughput than the Jetson Orin Nano implementation, while quantization affects pose accuracy [58]. Selecting between them requires accuracy, latency, energy per completed

inference, and the measurement boundary to be considered together.

Compression can reduce memory use and computation, but it also changes the deployed model. Quantization, pruning, and distillation therefore require new tests of output accuracy and, for a control component, closed-loop behavior. Peak memory,

worst observed execution time, timing margins, and energy over the actual duty cycle are more informative for mission integration than a model's parameter count alone.

B. Deployment and Model Updates

Deployment starts with a defined input interface, operating range, and response to invalid data or missed deadlines. The trained model is then converted to the target representation and tested with the processor, runtime, and numerical precision that will be used in operation. Closed-loop tests must include estimation errors, actuator limits, scheduling delays, and the supervisor's response. The ECSS machine learning handbook includes guidance on data qualification, model evaluation, post-training optimization, and integration within a safety cage [146].

Three deployment modes impose different maintenance needs. A *fixed model* performs inference with unchanged weights. A *ground-updated model* is retrained and evaluated on the ground before a controlled upload. An *onboard adaptive model* changes its parameters or policy using data collected during operation. Its update objective, training data, regression tests, and acceptance rules become part of the deployed system.

OPS-SAT experiments demonstrated onboard neural inference and the reprogramming of a neural-network implementation after launch [140], [141]. This supports the feasibility of updating inference software and firmware in flight. Safe continual learning additionally requires a way to evaluate a proposed update before it gains authority, retain a known acceptable configuration, and recover after an unsuccessful change. Existing software configuration and maintenance processes remain relevant to these tasks [147].

C. Verification and Runtime Assurance

An assurance argument combines evidence about prediction, closed-loop behavior, and implementation. Held-out data measure prediction performance under the tested distribution. Simulation and hardware tests expose interactions between perception, planning, actuation, and timing. Formal methods establish specified properties over a defined model and domain, while runtime monitors detect selected conditions and initiate a prescribed response.

Neural-network reachability and Lyapunov methods provide evidence tied to a network class, dynamics model, and operating region [97], [98], [100]. Section V discusses Hyperlyve's verification procedure and the distinction between its theoretical conditions and numerical implementation [99]. Deploying any verified controller also requires the implemented arithmetic, sampling, and execution timing to preserve the assumptions used in the analysis.

CBF filters, MPC, and runtime assurance supervisors can restrict a learned controller's commands [82], [85], [87]. Their guarantees require a suitable initial set, bounded model and estimation errors, sufficient actuator authority, feasible optimization or recovery actions, and acceptable intervention delays. For residual control, the analysis must cover the

baseline and learned correction together, including saturation and switching.

LeLaR makes the monitoring response concrete: its independent safety cage checks limits, disables the active controller after a rule violation, and places the system in idle before the wheels are switched off [16]. The PD controller is an experimental comparator. Assessing this arrangement requires connecting the monitored quantities and shutdown response to mission hazards, test results, operating restrictions, and failure modes that the monitor cannot detect.

D. Standards and Qualification

Space software engineering and product assurance are addressed by ECSS-E-ST-40C Rev. 1 and ECSS-Q-ST-80C Rev. 2, both issued in April 2025 [147], [148]. Their lifecycle processes include verification, configuration control, operation, and maintenance, providing a basis for managing software changes after launch. Machine learning (ML) adds the need to connect requirements to training data and learned behavior, and to reassess that connection when the model or operating conditions change.

ECSS-E-HB-40-02A, published in November 2024, provides guidance on developing and evaluating ML functions and on safety-cage integration [146]. This supporting handbook supplies a starting point for project-specific assurance; its stated scope excludes software of category A criticality. Highly critical functions and adaptation during operation therefore need assurance work beyond the handbook's guidance.

Experience from other domains offers complementary methods. ISO/PAS 8800:2024 addresses AI safety in road vehicles; UL 4600 organizes the evaluation of autonomous products around a safety argument [149], [150]. EASA's published Concept Paper Issue 2 gives guidance for Level 1 and Level 2 ML applications, including learning assurance and human-AI interaction [151]. Proposed Issue 3 was released for consultation in June 2026 [152]. These documents differ in scope and normative status. Their data, lifecycle, and safety-argument practices can inform space projects, but they do not directly qualify a spacecraft GNC function.

The required assurance effort should follow the consequences of failure and the system architecture. Control authority matters, but so do exposure to hazards, available recovery time, independent monitoring, shared failure modes, and reliance on a model's outputs by other functions. An advisory output can become safety-relevant when an operator or downstream controller depends on it. ECSS mission-execution autonomy levels describe how operations are scheduled and carried out; they are not substitutes for software criticality or assurance requirements [21], [146].

E. Interpreting Validation Evidence

Fig. 3 separates five evaluation modes: numerical or synthetic tests, laboratory imagery or ground robots, target-device inference, flight-data model tests, and learned flight control. A study can contribute to several columns. Here, *target-device inference* means measured execution on a processor or accelerator explicitly selected as a deployment candidate. This

includes LeLaR on InnoCube’s EFR32FG12 microcontroller and ground testing of the pose network on the physical Akida AKD1000 [16], [58]. Ordinary workstation benchmarks are recorded separately: SPNv2 reports CPU inference timing, and Celestini et al. use a workstation for their numerical comparisons [15], [53]. The target-device column concerns implementation, with space-environment suitability assessed through the additional requirements described above.

The flight-data column identifies the origin of evaluation data, while the learned-flight-control column identifies where the controller acted. SPNv2 was evaluated on synthetic and laboratory imagery and on 25 archived PRISMA flight images in *prisma25* [53]. The PRISMA images were used on the ground to evaluate the offline-trained network; online domain refinement was tested on the laboratory image domains. Jang et al. likewise evaluate a thermal model against lunar-orbiter telemetry on the ground [120]. LeLaR supplies both maneuver telemetry and evidence of a learned controller operating in orbit [16]. The remaining studies add other combinations: Celestini et al. include a physical free-flyer demonstration, the Akida study combines synthetic event data with processor tests, and Tanaka and Nagai assess a surrogate against numerical thermal analysis [15], [58], [116].

A technology-readiness assessment relates such results to a specified component or system, intended mission, relevant environment, and integration boundary [153]. Reporting the evaluation modes preserves these distinctions and helps identify what remains to be tested for a new application.

X. METHOD SELECTION AND SYSTEM DESIGN

Method selection starts with the limitation in the existing system: slow optimization, unreliable perception, costly task planning, or an inadequate prediction model. Learning is useful when it improves that function enough to justify its data, computation, and integration costs. Table IX connects the candidate interfaces reviewed in this paper to their intended benefits and the comparisons needed to select between them.

When repeated optimization is the main cost, solver imitation and learned warm-starts offer two different choices. Imitation moves most of the solution effort to offline trajectory generation and network training, producing a direct policy for a defined family of landing or transfer problems [69], [70]. A warm-start retains the online optimizer, allowing current constraints to enter the final solve [14], [15]. In Celestini et al.’s numerical obstacle-variation tests, ART-Rs reduced infeasible cases by about 82% relative to relaxation-based initialization of the same sequential convex programming solver. The practical benefit was better initialization across changing scenes. This favors a warm-start when solver convergence is the bottleneck and time remains for constrained refinement; direct imitation is attractive when a fast state-to-action mapping is needed and the task family can be bounded closely. In both cases, selection depends on total runtime and completed feasible trajectories, including failed solves and accumulated tracking error.

RL is useful when an action’s value depends on later observations or maneuvers, as in inspection planning. It can

also tune an existing constrained controller. In the rotating-target docking example of Wijayatunga et al., learned barrier parameters reduced median control effort by 11.1% relative to the ICCBF baseline; adding the second stage increased successful cases from 88 to 95 out of 100, with a larger upper tail of control effort [83]. A mission with a tight reserve must weigh the additional completed approaches against that recovery cost. The operating set and model assumptions determine which safety claims apply, as discussed in Sections IV and IX. Comparisons should therefore include task completion, constraint violations, interventions, and recovery effort under the same initial conditions.

For perception, the choice is often between greater robustness to image changes and a smaller, cheaper implementation. SPNv2’s domain refinement improves average pose performance on laboratory image domains, but some individual predictions deteriorate; its use depends on monitoring the effect on the downstream state estimator [53]. The Akida results show a different tradeoff: lower processor power accompanies reduced throughput and quantization-related accuracy loss [58]. A low-power device is useful only if it still provides sufficiently accurate updates at the required rate. Acquisition, preprocessing, inference, and state estimation must be timed together, because accelerating one stage may leave the dominant delay elsewhere.

Thermal surrogates and battery-health models support a further choice: spending computation to compare more operational options before committing to a maneuver or workload [114], [116], [117], [120]. Their value is measured by the resulting schedule—completed observations, preserved reserves, and fewer temperature-limit violations—as well as prediction error. A fast model is most useful when its valid range covers the candidate actions and its uncertainty can be carried into the planning constraints.

A. An Example of Coupled GNC, Computing, and Resource Decisions

The coupled model in Section VII can be used to compare two schedules for a spacecraft inspecting a non-cooperative target before eclipse. One schedule acquires and processes a dense image sequence immediately; another maintains the navigation update rate but defers optional inspection processing to a later interval. This illustrative design analysis links method selection to a specific operational decision. The first schedule may increase coverage before eclipse, while its pointing and computation change solar power, heat generation, and battery demand. The second leaves more immediate power and thermal margin but delays some mission products.

A learned pose estimator supplies measurements for the navigation filter, while a thermal surrogate predicts the temperature response to each workload and attitude sequence. The planner uses the shared power and heat inputs defined in Section VII to predict temperature and battery reserve for each schedule. It checks temperature limits and the storage constraints in (2), preserving energy for attitude control, communication, and eclipse heating. Collision and pointing limits, navigation deadlines, and a feasible termination maneuver

	Numerical / synthetic	Lab images / ground robot	Target-device inference	Flight-data model tests	Learned flight control
LeLaR attitude control	●	—	●	●	●
SPNv2 pose estimation	●	●	—	●	—
Akida V1 pose estimation	●	—	●	—	—
Takubo trajectory warm-start	●	—	—	—	—
Celestini trajectory warm-start	●	●	—	—	—
Tanaka thermal surrogate	●	—	—	—	—
Jang lunar thermal model	●	—	—	●	—

● Reported for the cited method

A study may report several kinds of tests; see Section IX for details.

Fig. 3. Evaluation modes reported by selected studies [14]–[16], [53], [58], [116], [120]. Dots mark reported tests; dashes mark tests not reported under the column definitions. Target-device inference uses a designated deployment processor or accelerator. Flight data may be processed on the ground, as in SPNv2 and Jang’s thermal model; LeLaR additionally demonstrates learned control in orbit. Section IX gives the devices and test conditions.

TABLE IX
LEARNING INTERFACES, INTENDED BENEFITS, AND SELECTION CRITERIA

Function	Candidate learning interface	Reason to use it	Conditions and decisive comparisons
Landing or transfer	Solver imitation or a learned trajectory warm-start [14], [15], [69]	Shift repeated computation offline, or improve the initial guess for an online solver	Reliable reference solutions and a defined task range; compare fuel, feasible completions, tracking error, and total runtime
RPO and docking	Learned guidance or barrier parameters with a constrained control layer [27], [83]	Improve approach behavior or reduce conservative control effort	Observable relative state and feasible intervention within disturbance bounds; compare completion, violations, and recovery effort
Inspection	Learned planner followed by a constrained tracker [77]	Account for future observation value when choosing maneuvers	Compatible planner/tracker models and timing; compare coverage, navigation error, fuel, and tracking feasibility
Attitude control	A fixed policy trained across modeled dispersions [16]	Encode a maneuver strategy in a compact onboard controller	Validated state interface and independent shutdown response; compare pointing, settling, wheel use, and disturbances with the flight baseline
Relative navigation	Learned pose estimation, optional domain refinement, or quantized inference [53], [58]	Improve visual feature extraction and adaptation, or reduce processing power	Representative target and sensor data; compare pose error, estimator consistency, update rate, and energy per inference
Telemetry monitoring	Anomaly scores feeding diagnosis or operator workflows [103], [113]	Prioritize unusual behavior in multichannel telemetry	Temporal test separation and an alert/response policy; compare event recall, false alarms, detection delay, and operator workload
Thermal prediction	A reduced-order or learned temperature model [116], [117], [120]	Evaluate candidate operating conditions with less model computation	Validated boundary conditions and independent protection; compare transient error, runtime, and effects on thermal decisions
Energy and workload scheduling	Battery-health prediction as an input to scheduling [114]	Update available-capacity estimates as the battery ages	Mission-relevant chemistry and cycling, power limits, and a feasible baseline schedule; compare reserve prediction and completed tasks

enter the same decision. Conservative uncertainty margins or a robust formulation account for estimation and prediction errors.

The choice of learning interface follows from this comparison. If evaluating temperature limits dominates planning time, a surrogate may allow more schedules to be considered before the deadline. If pose uncertainty is the limiting factor, improved visual estimation or a better viewing geometry may be more valuable than faster planning. If both candidate schedules violate the reserve constraint, reducing optional processing or ending inspection takes priority over improving either learned model. A useful evaluation compares these choices with a feasible fixed schedule under the same sensing, power, and thermal constraints, recording completed observations, fuel and energy use, temperature extrema, missed navigation updates, and termination performance.

XI. OPEN CHALLENGES AND A ROADMAP BASED ON EVIDENCE

A. Assurance for Integrated and Changing Systems

Assurance methods need to cover the interactions among learned perception, state estimation, guidance, control, and the physical vehicle. For a 6-DoF system, relevant effects include actuator saturation, uncertain inertia, flexible components, observation errors, and asynchronous execution. A certificate for one component or an idealized model is useful only when its assumptions can be connected to these system conditions. Extending verification to larger models therefore requires both computational progress and explicit accounting for model and implementation errors [97], [98], [100].

Adaptation during operation changes the system that was originally evaluated. A practical research step is to run candidate updates alongside a retained baseline before allowing them to issue commands. Tests can then measure performance changes, check operating limits, and exercise rollback. This makes the transition from learning a policy to using it in flight a concrete engineering process.

B. Comparable Evaluation and Reusable Data

The guidance and control literature would benefit from benchmarks that specify dynamics, observation availability, disturbances, actuator limits, initial-state distributions, and computational budgets. Performance should include success and constraint-violation rates, fuel or energy use, intervention frequency, recovery outcomes, and uncertainty across independent runs. Large errors and rare failures matter alongside averages. The anomaly-detection literature shows why evaluation choices such as threshold selection and event scoring must be stated before methods are ranked [113].

SPEED/SPEED+ and OPS-SAT-AD provide useful examples of shared evaluation resources [12], [49], [111]. LeLaR also makes control-maneuver telemetry available [16]. Reusable guidance/control benchmarks could build on these releases by linking the data to experiment commands, sensor characteristics, calibration records, software versions, and a common scoring protocol. Measured quantities, reconstructed states, and simulation-derived labels should be identified so that other teams can interpret the same data consistently.

C. Coupled Resources and Transfer Across Tasks

Thermal and energy models need to be evaluated as parts of operational decisions. The reviewed thermal-surrogate studies support temperature prediction under particular numerical, test, or mission-data conditions [116], [117], [119], [120]. Closing the loop introduces additional questions: whether predictions remain accurate after a new workload or attitude, how uncertainty changes near limits, and whether the available heaters, radiators, and operating modes can implement the proposed response. A useful next step is a joint experiment that changes computing load and pointing while measuring temperature, energy reserve, and GNC performance.

Transfer across missions can be studied by varying the vehicle, sensors, dynamics, and objectives separately. Trajectory transformers already reuse information across conditions represented in their training and evaluation sets [14], [15]. Testing on held-out configurations, together with the downstream optimizer or supervisor, would show which capabilities transfer and where new training is needed.

D. Milestones and Acceptance Criteria

The following milestones organize development around observable results. Each applies to a specific function and its intended operating conditions.

- 1) *Reproduce a bounded task.* Publish the operating assumptions, reference methods, training/test separation, and complete evaluation protocol. Demonstrate a useful improvement under the same constraints, including unsuccessful trials and the cost of any safety interventions.
- 2) *Validate the deployed implementation.* Run the converted model on its intended processor with representative sensor inputs, execution delays, actuator limits, and fault injections. Measure accuracy, memory, latency, energy, and the supervisor's response during both nominal and disturbed operation.
- 3) *Test the coupled system.* Evaluate the learned function with the navigation, control, thermal, and power interfaces that constrain its use. Demonstrate that a defined termination or recovery response remains feasible under the modeled uncertainties, and document the conditions outside that result.
- 4) *Demonstrate a restricted operational use.* Define the allowed maneuvers or decisions, independent monitoring, telemetry needed to interpret outcomes, and responsibilities for intervention. Report both performance and discrepancies between predicted and observed behavior. Expand the operating range only when the new conditions are supported by evidence.
- 5) *Evaluate controlled updates.* Preserve the accepted configuration and its evidence, assess each proposed update against the same safety and performance requirements, and exercise rollback. Onboard learning can be introduced within this process when its data, computation, and acceptance checks fit the mission constraints.

These milestones apply to thermal and energy functions as well as GNC. For example, a thermal predictor can first support ground analysis, then inform an onboard advisory

process, and finally influence heater or workload decisions. The tests would expand with its role, from temperature-prediction error to heater use, missed tasks, and responses near operating limits.

XII. CONCLUSION

Learning now contributes to several parts of spacecraft autonomy: extracting visual features, initializing trajectory solvers, generating feedback commands, detecting telemetry anomalies, and predicting temperature. The reviewed studies show progress at different stages of deployment. Pose and thermal models have been evaluated using laboratory and flight measurements; planar experiments combine learned tracking with capture and retraction; and LeLaR demonstrates learned attitude control in orbit. Data source, execution hardware, and control authority explain what each result contributes.

The clearest design pattern is the use of learning alongside established models and algorithms. A learned visual front end can feed a geometric estimator, a trajectory predictor can initialize an optimizer, and a thermal surrogate can speed repeated analysis. Comparisons are most useful when they identify the baseline, the performance gain, and its computing cost under the same task conditions. For learned controllers, the analysis also has to include actuator limits, estimation errors, and supervisory actions. Current ECSS guidance helps organize this development and verification work.

Thermal and energy constraints connect these functions. An observation or maneuver affects power generation, processor load, and temperature as well as pointing accuracy and fuel use. Evaluating these effects together can reveal whether a faster predictor or a better controller actually allows the spacecraft to complete more useful work. Coupled experiments, shared data, and reproducible comparisons are therefore the next practical steps toward broader autonomous operation.

ABOUT ASTROUP

Founded in 2026 and headquartered in Beijing, AstroUp focuses on the development and application of embodied robots for space.

Drawing on embodied intelligence, brain-inspired models, and multimodal perception, the company develops robotic systems with perception, decision-making, and manipulation capabilities for satellite servicing, in-orbit research, development, and manufacturing, and lunar and deep-space exploration.

The team consists of young scientists in aerospace and artificial intelligence. Its core members have experience in the overall design of space robotic systems and in multiple in-orbit validation campaigns. By combining engineering experience gained from real space missions with artificial intelligence, AstroUp advances the development, in-orbit validation, and practical deployment of space robotic systems.

REFERENCES

- [1] F. L. Markley and J. L. Crassidis, *Fundamentals of Spacecraft Attitude Determination and Control*. Springer, 2014.
- [2] J. L. Crassidis and J. L. Junkins, *Optimal Estimation of Dynamic Systems*, 2nd ed. CRC Press, 2011.
- [3] J. Riedel et al., “Using autonomous navigation for interplanetary missions: The validation of Deep Space 1 AutoNav,” conference paper, May 2000, NASA NTRS record 20060032521. [Online]. Available: <https://ntrs.nasa.gov/citations/20060032521>
- [4] S. Chien et al., “Using autonomy flight software to improve science return on Earth Observing One,” *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 4, pp. 196–216, 2005.
- [5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” arXiv:1707.06347, 2017.
- [6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *Proc. ICML*, 2018, pp. 1861–1870.
- [7] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning,” arXiv:1509.02971, 2015.
- [8] D. Izzo, M. Märtens, and B. Pan, “A survey on artificial intelligence trends in spacecraft guidance dynamics and control,” *Astrodynamics*, vol. 3, no. 4, pp. 287–299, 2019.
- [9] M. Tipaldi, R. Iervolino, and P. R. Massenio, “Reinforcement learning in spacecraft control applications: Advances, prospects, and challenges,” *Annual Reviews in Control*, vol. 54, pp. 1–23, 2022.
- [10] K. Thangavel, R. Sabatini, A. Gardi et al., “Artificial intelligence for trusted autonomous satellite operations,” *Progress in Aerospace Sciences*, vol. 144, art. 100960, 2024, doi:10.1016/j.paerosci.2023.100960.
- [11] A. Fejjari, A. Delavault, R. Camilleri, and G. Valentino, “A review of anomaly detection in spacecraft telemetry data,” *Applied Sciences*, vol. 15, no. 10, p. 5653, 2025.
- [12] T. H. Park, M. Märtens, G. Lecuyer, D. Izzo, and S. D’Amico, “SPEED+: Next-generation dataset for spacecraft pose estimation across domain gap,” in *Proc. IEEE Aerospace Conf.*, 2022, pp. 1–15.
- [13] T. H. Park et al., “Satellite Pose Estimation Competition 2021: Results and analyses,” *Acta Astronautica*, vol. 204, pp. 640–665, 2023.
- [14] Y. Takubo, T. Guffanti, D. Gammelli, M. Pavone, and S. D’Amico, “Towards robust spacecraft trajectory optimization via transformers,” in *Proc. IEEE Aerospace Conf.*, 2025.
- [15] D. Celestini, A. Afsharrad, D. Gammelli, T. Guffanti, G. Zardini, S. Lall, E. Capello, S. D’Amico, and M. Pavone, “Generalizable spacecraft trajectory generation via multimodal learning with transformers,” in *Proc. American Control Conf. (ACC)*, Denver, CO, USA, 2025. Author manuscript: arXiv:2410.11723. [Online]. Available: <https://arxiv.org/abs/2410.11723>
- [16] K. Djebko, T. Baumann, E. Dilger, F. Puppe, and S. Montenegro, “LeLaR: The first in-orbit demonstration of an AI-based satellite attitude controller,” *IEEE Access*, 2026, doi:10.1109/ACCESS.2026.3678816. Author manuscript: arXiv:2512.19576v5, <https://arxiv.org/abs/2512.19576v5>. Telemetry: <https://github.com/kdjebko/lelar-in-orbit-data>.
- [17] D. G. Gilmore, *Spacecraft Thermal Control Handbook*, 2nd ed. Aerospace Press, 2002.
- [18] J. Meseguer, I. Pérez-Grande, and A. Sanz-Andrés, *Spacecraft Thermal Control*. Woodhead Publishing, 2012.
- [19] J. Deng, “Embodied intelligence for space: Space-grade hardware, the operational brain, and the scientific brain—a survey,” unpublished manuscript, 2026.
- [20] J. Deng, “The operational brain of space robots: Sim-to-ground-to-space transfer, neuromorphic computing, and space-specific perception—a survey,” unpublished manuscript, 2026.
- [21] European Cooperation for Space Standardization, “Space engineering: Space segment operability,” ECSS-E-ST-70-11C, 2008.
- [22] N. Muscettola, P. P. Nayak, B. Pell, and B. C. Williams, “Remote Agent: To boldly go where no AI system has gone before,” *Artificial Intelligence*, vol. 103, no. 1–2, pp. 5–47, 1998.
- [23] S. Bonasera, N. Bosanac, C. J. Sullivan et al., “Designing Sun–Earth L2 halo orbit station keeping maneuvers via reinforcement learning,” *Journal of Guidance, Control, and Dynamics*, vol. 46, no. 2, pp. 301–311, 2023.
- [24] N. A. Tafañidis, A. Banerjee, S. G. Satpute et al., “Reinforcement learning based intelligent control for low-thrust tight station keeping in low Earth orbit,” *International Journal of Control, Automation and Systems*, vol. 23, pp. 2105–2116, 2025.
- [25] S. Sharma and S. D’Amico, “Neural network-based pose estimation for noncooperative spacecraft rendezvous,” *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 6, pp. 4638–4658, 2020.
- [26] K. Hovell and S. Ulrich, “Deep reinforcement learning for spacecraft proximity operations guidance,” *Journal of Spacecraft and Rockets*, vol. 58, no. 2, pp. 254–264, 2021.

- [27] K. Dunlap, M. Mote, K. Delsing, and K. L. Hobbs, "Run time assured reinforcement learning for safe satellite docking," *Journal of Aerospace Information Systems*, vol. 20, no. 1, pp. 25–36, 2023.
- [28] K. Dunlap, K. Bennett, D. van Wijk, N. Hamilton, and K. L. Hobbs, "Run time assured reinforcement learning for six degree-of-freedom spacecraft inspection," arXiv:2406.11795, 2024.
- [29] K. Dunlap, N. P. Hamilton, Z. Lippay, M. Shubert, S. Phillips, and K. L. Hobbs, "Demonstrating reinforcement learning and run time assurance for spacecraft inspection using unmanned aerial vehicles," in *AIAA SciTech Forum*, 2025.
- [30] R. B. Friend, "Orbital Express program summary and mission overview," in *Proc. SPIE*, vol. 6958, 2008.
- [31] J. L. Forshaw et al., "RemoveDEBRIS: An in-orbit active debris removal demonstration mission," *Acta Astronautica*, vol. 127, pp. 448–463, 2016.
- [32] NASA, "On-orbit servicing, assembly, and manufacturing 1 (OSAM-1)," project status, accessed Oct. 15, 2026. [Online]. Available: <https://www.nasa.gov/mission/on-orbit-servicing-assembly-and-manufacturing-1/>
- [33] A. Crain, K. Hovell, C. Savitska, and S. Ulrich, "Optimal capture of a spinning spacecraft via deep learning vision and guidance," *Journal of Spacecraft and Rockets*, vol. 62, no. 5, pp. 1727–1739, 2025.
- [34] B. Beigomi and Z. H. Zhu, "Towards real-world efficiency: Domain randomization in reinforcement learning for pre-capture of free-floating moving targets by autonomous robots," in *Proc. IEEE ICRA*, 2024, pp. 11753–11759.
- [35] M. D'Ambrosio, L. Capra, A. Brandonisio, S. Silvestrini, and M. Lavagna, "Redundant space manipulator autonomous guidance for in-orbit servicing via deep reinforcement learning," *Aerospace*, vol. 11, no. 5, p. 341, 2024.
- [36] B. Gaudet, R. Linares, and R. Furfaro, "Deep reinforcement learning for six degree-of-freedom planetary landing," *Advances in Space Research*, vol. 65, no. 7, pp. 1723–1741, 2020.
- [37] D. Iafrate, A. Brandonisio, R. Hinz, and M. R. Lavagna, "Propulsive landing of launchers' first stages with deep reinforcement learning," *Acta Astronautica*, vol. 227, pp. 40–56, 2025.
- [38] A. E. Johnson et al., "The Lander Vision System for Mars 2020 entry descent and landing," in *Proc. AAS Guidance and Control Conf.*, Breckenridge, CO, USA, 2017, paper AAS 17-036. [Online]. Available: <https://ntrs.nasa.gov/citations/20190025658>
- [39] NASA, "Maturing the Lander Vision System for NASA's Mars Perseverance Rover," in *Transitions of Flight Tested Technologies*, accessed Oct. 15, 2026. [Online]. Available: <https://www.nasa.gov/stmd-flight-opportunities/transitions-of-flight-tested-technologies/>
- [40] E. Adams, N. Chabot, A. Cheng, A. Rivkin, E. Reynolds, and C. Shearer, "Final technical report to the National Aeronautics and Space Administration for the Double Asteroid Redirection Test (DART) mission," Johns Hopkins University Applied Physics Laboratory, Laurel, MD, USA, Oct. 2023. NASA NTRS 20230015804. [Online]. Available: <https://ntrs.nasa.gov/api/citations/20230015804/downloads/DART%20Final%20Technical%20Report.pdf>
- [41] L. Xu, G. Zhang, S. Qiu, "Reinforcement learning-based multi-impulse rendezvous approach for satellite constellation reconfiguration," *Acta Astronautica*, vol. 224, pp. 325–337, 2024.
- [42] H. Ren, R. Zhong, and H. Gui, "Learning-based model predictive control for cooperative spacecraft swarm reconfiguration," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 62, pp. 818–834, 2026, doi:10.1109/TAES.2025.3624705.
- [43] J. Aurand, S. Cutlip, H. Lei, K. Lang, and S. Phillips, "Deep Q-learning for decentralized multi-agent inspection of a tumbling target," *Journal of Spacecraft and Rockets*, vol. 61, no. 2, pp. 341–354, 2024.
- [44] H. H. Lei, M. Shubert, N. Damron, K. Lang, and S. Phillips, "Deep reinforcement learning for multiagent autonomous satellite inspection," in *Proc. 44th AAS Guidance, Navigation and Control Conf.*, 2022, pp. 1391–1412.
- [45] NASA Goddard Space Flight Center, "Pulsar-based navigation & solar-system timekeeping," NICER technology development, updated May 6, 2026. Accessed: Oct. 15, 2026. [Online]. Available: <https://heasarc.gsfc.nasa.gov/docs/nicer/technology/spinoffs/pulsar-pnt.html>
- [46] S. I. Sheikh, D. J. Pines, P. S. Ray, K. S. Wood, M. N. Lovellette, and M. T. Wolff, "Spacecraft navigation using X-ray pulsars," *Journal of Guidance, Control, and Dynamics*, vol. 29, no. 1, pp. 49–63, 2006.
- [47] W. Zheng, Y. Wang, K. Jiang, and Y. Wang, "Overview of X-ray pulsar-based navigation methods," *Acta Aeronautica et Astronautica Sinica*, vol. 44, no. 3, Art. no. 527451, 2023, doi:10.7527/S1000-6893.2022.27451.
- [48] S. Sharma and S. D'Amico, "Pose estimation for non-cooperative spacecraft rendezvous using neural networks," in *AAS/AIAA Space Flight Mechanics Meeting*, 2019, paper AAS 19-404.
- [49] S. Sharma, T. H. Park, and S. D'Amico, "Spacecraft Pose Estimation Dataset (SPEED)," Stanford Digital Repository, 2019, doi:10.25740/rv692ff0717.
- [50] T. H. Park and S. D'Amico, "SHIRT: Satellite hardware-in-the-loop rendezvous trajectories dataset," Stanford Digital Repository, 2022, doi:10.25740/zq716br5462. [Online]. Available: <https://purl.stanford.edu/zq716br5462>
- [51] P. F. Proença and Y. Gao, "Deep learning for spacecraft pose estimation from photorealistic rendering," in *Proc. IEEE ICRA*, 2020, pp. 6007–6013.
- [52] M. A. Musallam et al., "SPARK: Spacecraft recognition leveraging knowledge of space environment," arXiv:2104.05978, 2021.
- [53] T. H. Park and S. D'Amico, "Robust multi-task learning and online refinement for spacecraft pose estimation across domain gap," *Advances in Space Research*, vol. 73, no. 11, pp. 5726–5740, 2024, doi:10.1016/j.asr.2023.03.036.
- [54] K. Liu et al., "Revisiting the domain gap issue in non-cooperative spacecraft pose tracking," in *Proc. IEEE/CVF CVPR Workshops (AI4Space)*, 2024.
- [55] F. T. Despond and S. Ulrich, "Real-time stereovision-based spacecraft pose determination using convolutional neural networks," *Journal of Spacecraft and Rockets*, vol. 62, no. 1, pp. 269–279, 2025.
- [56] M. Bechini, G. Gu, P. Lunghi, and M. Lavagna, "Robust spacecraft relative pose estimation via CNN-aided line segments detection in monocular images," *Acta Astronautica*, vol. 215, pp. 20–43, 2024.
- [57] E. Bates and S. D'Amico, "Removing ambiguities in concurrent monocular single-shot spacecraft shape and pose estimation using a deep neural network," in *AAS Guidance, Navigation and Control Conf.*, Breckenridge, CO, 2025.
- [58] A. Rathinam, J. Lecomte, J. Reelsen, G. Lenz, A. von Arnim, and D. Aouada, "Efficient onboard spacecraft pose estimation with event cameras and neuromorphic hardware," arXiv:2604.04117v1, 2026. [Online]. Available: <https://arxiv.org/abs/2604.04117v1>
- [59] G. Gallego et al., "Event-based vision: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 1, pp. 154–180, 2022, doi:10.1109/TPAMI.2020.3008413.
- [60] G. Cohen, S. Afshar, B. Morreale, T. Bessell, A. Wabnitz, M. Rutten, and A. van Schaik, "Event-based sensing for space situational awareness," *The Journal of the Astronautical Sciences*, vol. 66, no. 2, pp. 125–141, 2019.
- [61] S. Afshar, A. P. Nicholson, A. van Schaik, and G. Cohen, "Event-based object detection and tracking for space situational awareness," *IEEE Sensors Journal*, vol. 20, no. 24, pp. 15117–15132, 2020.
- [62] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *Proc. IEEE ICRA*, 2014, pp. 15–22.
- [63] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: Learning to cure the curse of drift in inertial odometry," in *Proc. AAAI*, 2018.
- [64] W. Steigerwald, "Bennu's boulders shine as beacons for NASA's OSIRIS-REx," NASA, Mar. 9, 2020. [Online]. Available: <https://www.nasa.gov/missions/bennu-boulders-shine-as-beacons-for-nasas-osiris-rex/>
- [65] D. S. Lauretta et al., "Spacecraft sample collection and subsurface excavation of asteroid (101955) Bennu," *Science*, vol. 377, no. 6603, pp. 285–291, 2022.
- [66] Y. Tsuda, T. Saiki, F. Terui, S. Nakazawa, M. Yoshikawa, and S.-i. Watanabe, "Hayabusa2 mission status: Landing, roving and cratering on asteroid Ryugu," *Acta Astronautica*, vol. 171, pp. 42–54, 2020.
- [67] S. Silvestrini, M. Piccinin, G. Zanotti, A. Brandonisio, I. Bloise, L. Feruglio, P. Lunghi, M. Lavagna, and M. Varile, "Optical navigation for Lunar landing based on Convolutional Neural Network crater detector," *Aerospace Science and Technology*, vol. 123, art. 107503, 2022, doi:10.1016/j.ast.2022.107503.
- [68] Y. Chihabi and S. Ulrich, "Nonlinear adaptive angle-only relative navigation on perturbed eccentric orbits," *Acta Astronautica*, vol. 228, pp. 494–514, 2025.
- [69] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via deep neural networks: Study on landing problems," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 5, pp. 1122–1135, 2018.
- [70] C. I. Sprague, D. Izzo, and P. Ögren, "Learning dynamic-objective policies from a class of optimal trajectories," in *Proc. 59th IEEE Conf. Decision and Control (CDC)*, 2020, pp. 597–602, doi:10.1109/CDC42340.2020.9303931.

- [71] S. Shao, D. Zhou, G. Sun, L. Zhang, and M. Jiang, "Imitation learning-based spacecraft rendezvous and docking method with expert demonstration," arXiv:2601.12952v1, 2026. [Online]. Available: <https://arxiv.org/abs/2601.12952v1>
- [72] S. Willis, D. Izzo, and D. Hennes, "Reinforcement learning for spacecraft maneuvering near small bodies," in *AAS/AIAA Space Flight Mechanics Meeting*, vol. 158 of *Advances in the Astronautical Sciences*, 2016, pp. 1351–1368.
- [73] J. Broida and R. Linares, "Spacecraft rendezvous guidance in cluttered environments via reinforcement learning," in *29th AAS/AIAA Space Flight Mechanics Meeting*, 2019, pp. 1–15.
- [74] C. E. Oestreich, R. Linares, and R. Gondhalekar, "Autonomous six-degree-of-freedom spacecraft docking with rotating targets via reinforcement learning," *Journal of Aerospace Information Systems*, vol. 18, no. 7, pp. 417–428, 2021.
- [75] L. Federici, A. Scorsoglio, A. Zavoli, and R. Furfaro, "Meta-reinforcement learning for adaptive spacecraft guidance during finite-thrust rendezvous missions," *Acta Astronautica*, vol. 201, pp. 129–141, 2022.
- [76] S. Shao, D. Zhou, G. Sun, W. Ma, and R. Deng, "Asymmetric deep reinforcement learning-based spacecraft approaching maneuver under unknown disturbance," *Aerospace*, vol. 12, no. 3, p. 170, 2025.
- [77] L. Capra, A. Brandonisio, and M. R. Lavagna, "Reinforced model predictive guidance and control for spacecraft proximity operations," *Aerospace*, vol. 12, no. 9, p. 837, 2025.
- [78] A. Brandonisio, M. Bechini, G. L. Civardi, L. Capra, and M. Lavagna, "Closed-loop AI-aided image-based GNC for autonomous inspection of uncooperative space objects," *Aerospace Science and Technology*, vol. 155, p. 109700, 2024.
- [79] M. Foutter, P. Bhoj, R. Sinha, A. Elhafsi, S. Banerjee, C. Agia, J. Kruger, T. Guffanti, D. Gammelli, S. D'Amico, and M. Pavone, "Adapting a foundation model for space-based tasks," in *Robotics: Science and Systems Workshop*, Delft, The Netherlands, 2024. [Online]. Available: https://slab.stanford.edu/sites/g/files/tbjybj25201/files/media/file/foutter_ea_rss2024.pdf
- [80] R. Furfaro, I. Bloise, M. Orlandelli, P. Di Lizia, F. Topputo, and R. Linares, "Deep learning for autonomous lunar landing," in *Advances in the Astronautical Sciences*, vol. 167, 2018, pp. 3285–3306.
- [81] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. ICML*, 2017, pp. 22–31.
- [82] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *Proc. European Control Conf.*, 2019, pp. 3420–3431.
- [83] M. Wijayatunga et al., "Learning safety-guaranteed, non-greedy control barrier functions using reinforcement learning," arXiv:2602.00366, 2026.
- [84] M. Willoughby, K. Richelmy, and H. Peng, "Satellite reorientation using reinforcement learning under unknown attitude failure: Sun-searching implementation," in *AIAA SciTech Forum*, 2025, paper 2025-1145.
- [85] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Towards safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.
- [86] L. Brunke et al., "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.
- [87] J. D. Schierman, M. D. DeVore, N. D. Richards, and M. A. Clark, "Runtime assurance for autonomous aerospace systems," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 12, pp. 2205–2217, 2020, doi:10.2514/1.G004862.
- [88] Z. Xie, T. Sun, T. H. Kwan, Z. Mu, and X. Wu, "A new reinforcement learning based adaptive sliding mode control scheme for free-floating space robotic manipulator," *IEEE Access*, vol. 8, pp. 127048–127064, 2020.
- [89] C.-K. McQuinn, K. Dunlap, N. Hamilton, J. Wilson, and K. L. Hobbs, "Run time assurance for simultaneous constraint satisfaction during spacecraft attitude maneuvering," in *Proc. IEEE Aerospace Conf.*, 2024.
- [90] T. Kim, R. I. Kee, I. Kolmanovsky, and A. Girard, "Constrained control for autonomous spacecraft rendezvous: Learning-based time shift governor," in *AIAA SciTech Forum*, 2025, paper 2025-1700.
- [91] A. M. Bokor, F. Biertümpfel, P. Seiler, and R. Tóth, "Robust model predictive control for spacecraft rendezvous under sector-bounded nonlinearities," *IEEE Control Systems Letters*, 2025.
- [92] L. Arora and S. Ulrich, "Koopman expectation-based optimization for offline spacecraft rendezvous guidance under parametric uncertainties," *Journal of Guidance, Control, and Dynamics*, vol. 49, no. 7, pp. 2144–2153, 2026.
- [93] E. Papadopoulos and S. Dubowsky, "On the nature of control algorithms for free-floating space manipulators," *IEEE Trans. Robot. Autom.*, vol. 7, no. 6, pp. 750–758, 1991.
- [94] S. Dubowsky and E. Papadopoulos, "The kinematics, dynamics, and control of free-flying and free-floating space robotic systems," *IEEE Trans. Robot. Autom.*, vol. 9, no. 5, pp. 531–543, 1993.
- [95] Y. Umetani and K. Yoshida, "Resolved motion rate control of space manipulators with generalized Jacobian matrix," *IEEE Trans. Robot. Autom.*, vol. 5, no. 3, pp. 303–314, 1989.
- [96] Y. Wei, X. Bai, and H. Lu, "Trajectory planning of free-floating space robot for non-cooperative tumbling target capture based on deep reinforcement learning," *Robotica*, vol. 43, no. 7, pp. 2674–2692, 2025.
- [97] J. Liu, Y. Meng, M. Fitzsimmons, and R. Zhou, "Physics-informed neural network Lyapunov functions: PDE characterization, learning, and verification," *Automatica*, vol. 175, p. 112193, 2025.
- [98] J. Liu, Y. Meng, M. Fitzsimmons, and R. Zhou, "LyZNet: A lightweight Python tool for learning and verifying neural Lyapunov functions and regions of attraction," in *Proc. HSCC*, 2024.
- [99] J. Wayment, B. Yarbrough, J. Wang, S. Sundaram, and P. E. Paré, "Hyperlyve: Hyperplane partitioning for neural Lyapunov verification," arXiv:2605.03992v2, 2026. [Online]. Available: <https://arxiv.org/abs/2605.03992v2>
- [100] R. Ivanov, J. Weimer, R. Alur, G. J. Pappas, and I. Lee, "Verisig: Verifying safety properties of hybrid systems with neural network controllers," in *Proc. HSCC*, 2019, pp. 169–178.
- [101] I. Hwang, S. Kim, Y. Kim, and C. E. Seah, "A survey of fault detection, isolation, and reconfiguration methods," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 3, pp. 636–653, 2010.
- [102] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part I," *IEEE Trans. Ind. Electron.*, vol. 62, no. 6, pp. 3757–3767, 2015.
- [103] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, "Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding," in *Proc. ACM SIGKDD*, 2018, pp. 387–395.
- [104] C. O'Meara, L. Schlag, and M. Wickler, "Applications of deep learning neural networks to satellite telemetry monitoring," in *Proc. SpaceOps*, 2018, paper 2018-2558, doi:10.2514/6.2018-2558.
- [105] T. Yairi, N. Takeishi, T. Oda, Y. Nakajima, N. Nishimura, and N. Takata, "A data-driven health monitoring method for satellite housekeeping data based on probabilistic clustering and dimensionality reduction," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 53, no. 3, pp. 1384–1401, 2017.
- [106] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proc. ACM SIGKDD*, 2019, pp. 2828–2837.
- [107] J. Xu, H. Wu, J. Wang, and M. Long, "Anomaly Transformer: Time series anomaly detection with association discrepancy," in *Proc. ICLR*, 2022.
- [108] S. Tuli, G. Casale, and N. R. Jennings, "TranAD: Deep transformer networks for anomaly detection in multivariate time series data," *Proc. VLDB Endowment*, vol. 15, no. 6, pp. 1201–1214, 2022.
- [109] B. Yu, Y. Yu, J. Xu, G. Xiang, and Z. Yang, "MAG: A novel approach for effective anomaly detection in spacecraft telemetry data," *IEEE Trans. Ind. Informat.*, vol. 20, no. 3, pp. 3891–3899, 2024, doi:10.1109/TII.2023.3314852.
- [110] R. Gallon, F. Schiemenz, A. Menicucci, and E. Gill, "Convolutional neural network design and evaluation for real-time multivariate time series fault detection in spacecraft attitude sensors," *Advances in Space Research*, vol. 76, no. 5, pp. 2960–2976, 2025, doi:10.1016/j.asr.2025.06.068.
- [111] B. Ruszczak, "OPS-SAT-AD: Anomaly detection dataset for satellite telemetry," version 1, Zenodo, Jun. 29, 2024, doi:10.5281/zenodo.12588359.
- [112] B. Ruszczak, K. Kotowski, D. Evans, and J. Nalepa, "The OPS-SAT benchmark for detecting anomalies in satellite telemetry," *Scientific Data*, vol. 12, art. 710, 2025, doi:10.1038/s41597-025-05035-3.
- [113] L. Herrmann, M. Bieber, W. J. C. Verhagen, F. Cosson, and B. F. Santos, "Unmasking overestimation: A re-evaluation of deep anomaly detection in spacecraft telemetry," *CEAS Space Journal*, vol. 16, no. 2, pp. 225–237, 2024, doi:10.1007/s12567-023-00529-5.
- [114] R. R. Richardson, M. A. Osborne, and D. A. Howey, "Gaussian process regression for forecasting battery state of health," *Journal of*

- Power Sources*, vol. 357, pp. 209–219, 2017.
- [115] G. Kopp and J. L. Lean, “A new, lower value of total solar irradiance: Evidence and climate significance,” *Geophysical Research Letters*, vol. 38, p. L01706, 2011.
- [116] H. Tanaka and H. Nagai, “Thermal surrogate model for spacecraft systems using physics-informed machine learning with POD data reduction,” *International Journal of Heat and Mass Transfer*, vol. 213, p. 124336, 2023.
- [117] D. Yamashita, H. Tanaka, T. Ikami, and H. Nagai, “Transient system-wide surrogate model for spacecraft thermal analysis using recurrent neural networks with POD data reduction,” *International Journal of Heat and Mass Transfer*, vol. 255, art. 127854, 2026, doi:10.1016/j.ijheatmasstransfer.2025.127854.
- [118] H. Tanaka and H. Nagai, “Data-driven thermal state estimation for in-orbit systems via physics-informed machine learning,” *Acta Astronautica*, vol. 212, pp. 316–328, 2023, doi:10.1016/j.actaastro.2023.07.039.
- [119] B. Jang, W. Lee, J.-J. Lee, and H. Jin, “Artificial neural network-based temperature prediction of a lunar orbiter in thermal vacuum test: Data-driven reduced-order models,” *Aerospace Science and Technology*, vol. 145, art. 108867, 2024, doi:10.1016/j.ast.2023.108867.
- [120] B. Jang, M.-J. Jeon, S. Lee, and H. Jin, “Neural network-based thermal model for virtual metrology of lunar orbiter temperatures via active and transfer learning,” *International Communications in Heat and Mass Transfer*, vol. 165, part A, art. 109055, 2025, doi:10.1016/j.icheatmasstransfer.2025.109055.
- [121] J. Ding, X. Bai, W. Zhao, and M. Xu, “Orbit maintenance strategy integrated with power management for ultra-low Earth orbit satellites,” in *Proc. 76th International Astronautical Congress*, Sydney, Australia, 2025, pp. 1304–1314, doi:10.52202/083087-0115.
- [122] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan, “Temperature-aware microarchitecture: Modeling and implementation,” *ACM Trans. Architecture and Code Optimization*, vol. 1, no. 1, pp. 94–125, 2004.
- [123] P. Lunghi et al., “Energy efficiency analysis of spiking neural networks for space applications,” *Astrodynamics*, vol. 9, pp. 909–932, 2025.
- [124] NASA, “Artificial intelligence for spacecraft thermal control systems,” in *Fiscal Year 2024 SBIR Phase I Solicitation*, 2024. [Online]. Available: <https://www.nasa.gov/wp-content/uploads/2024/01/2024sbirphaseisolicitationfinal.pdf>
- [125] J. H. Henninger, “Solar absorptance and thermal emittance of some common spacecraft thermal-control coatings,” NASA RP-1121, Apr. 1984. [Online]. Available: <https://ntrs.nasa.gov/citations/19840015630>
- [126] H. Benninghoff, F. Rems, E.-A. Risse, and C. Mietner, “European proximity operations simulator 2.0 (EPOS)—a robotic-based rendezvous and docking simulator,” *J. Large-Scale Research Facilities*, vol. 3, p. 107, 2017.
- [127] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *Proc. IEEE/RSJ IROS*, 2017, pp. 23–30.
- [128] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization,” in *Proc. IEEE ICRA*, 2018, pp. 3803–3810.
- [129] Y. Chebotar et al., “Closing the sim-to-real loop: Adapting simulation randomization with real world experience,” in *Proc. IEEE ICRA*, 2019, pp. 8973–8979.
- [130] S. Höfer et al., “Sim2real in robotics and automation: Applications and challenges,” *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 398–400, 2021.
- [131] L. Banker, M. Wozniak, M. Alameer, S. N. Paul, D. Meisinger, G. Baer, T. Hunting, R. Dunham, and J. Kamdar, “Current state of the multi-agent multi-view experimental and digital twin rendezvous (MMEDR-Autonomous) framework,” arXiv:2603.20575v1, 2026. [Online]. Available: <https://arxiv.org/abs/2603.20575v1>
- [132] P. P. Nayak et al., “Validating the DS1 Remote Agent Experiment,” in *Proc. International Symposium on Artificial Intelligence, Robotics and Automation in Space*, 1999. [Online]. Available: <https://ai.jpl.nasa.gov/public/documents/papers/rax-results-isairas99.pdf>
- [133] K. L. Wagstaff, A. Altinok, S. A. Chien, U. Rebbapragada, S. R. Schaffer, D. R. Thompson, and D. Q. Tran, “Cloud filtering and novelty detection using onboard machine learning for the EO-1 spacecraft,” in *Proc. AI in the Oceans and Space Workshop, IJCAI*, Melbourne, Australia, Aug. 2017. [Online]. Available: <https://ai.jpl.nasa.gov/public/documents/papers/wagstaff-ijcai2017-novelty.pdf>
- [134] European Space Agency, “Digital Twin Earth, quantum computing and AI take centre stage at ESA’s Φ -week,” Sep. 28, 2020. [Online]. Available: https://www.esa.int/Applications/Observing_the_Earth/Digital_Twin_Earth/quantum_computing_and_AI_take_centre_stage_at_ESA_s_Phi-week
- [135] R. T. Daly et al., “Successful kinetic impact into an asteroid for planetary defence,” *Nature*, vol. 616, pp. 443–447, 2023, doi:10.1038/s41586-023-05810-5.
- [136] T. Suzuki, I. Kawano, M. Yamada, and H. Maeda, “Rendezvous-docking experiments of Engineering Test Satellite 7,” *Toshiba Review*, vol. 54, no. 6, pp. 9–12, 1999, in Japanese. [Online]. Available: <https://repository.exst.jaxa.jp/dspace/handle/a-is/41428>
- [137] K. Yoshida, “Engineering Test Satellite VII flight experiments for space robot dynamics and control,” *Int. J. Robotics Research*, vol. 22, no. 5, pp. 321–335, 2003.
- [138] G. S. Aglietti et al., “RemoveDEBRIS: An in-orbit demonstration of technologies for the removal of space debris,” *The Aeronautical Journal*, vol. 124, no. 1271, pp. 1–23, 2020, doi:10.1017/aer.2019.136.
- [139] University of Surrey, “RemoveDEBRIS mission,” mission documentation. [Online]. Available: <https://www.surrey.ac.uk/surrey-space-centre/missions/removedebris>. Accessed: Oct. 15, 2026.
- [140] Thales Alenia Space, “AI test success on OPS-SAT,” news article, Dec. 17, 2021. Accessed: Oct. 15, 2026. [Online]. Available: <https://www.thalesaleniaspace.com/en/news/ai-test-success-ops-sat>
- [141] L. Chavier, B. Bonham-Carter, H. Burd, T. Heydrich, G. O’Shea, J. Prud’homme et al., “Deploying artificial intelligence capabilities by hybridizing a neural network on a satellite,” in *Proc. ASCEND 2023*, Las Vegas, NV, USA, Oct. 2023, paper 2023-4718, doi:10.2514/6.2023-4718.
- [142] European Space Agency, “ Φ sat-2 begins science phase for AI Earth images,” Jul. 15, 2025. [Online]. Available: https://www.esa.int/Applications/Observing_the_Earth/FutureEO/Phsat-2/Phsat-2_begins_science_phase_for_AI_Earth_images
- [143] G. Furano et al., “Towards the use of artificial intelligence on the edge in space systems: Challenges and opportunities,” *IEEE Aerosp. Electron. Syst. Mag.*, vol. 35, no. 12, pp. 44–56, 2020.
- [144] B. Denby and B. Lucia, “Orbital edge computing: Nanosatellite constellations as a new class of computer system,” in *Proc. ACM ASPLOS*, 2020, pp. 939–954.
- [145] E. Lagunas et al., “Performance evaluation of neuromorphic hardware for onboard satellite communication applications,” *IEEE Wireless Communications*, vol. 31, no. 6, pp. 78–84, 2024.
- [146] European Cooperation for Space Standardization, “Space engineering—Machine learning handbook,” ECSS-E-HB-40-02A, Nov. 15, 2024. [Online]. Available: <https://ecss.nl/home/ecss-e-hb-40-02a-15-november-2024/>
- [147] European Cooperation for Space Standardization, “Space engineering—Software,” ECSS-E-ST-40C Rev. 1, Apr. 30, 2025. [Online]. Available: <https://ecss.nl/standard/ecss-e-st-40c-rev-1-software-30-april-2025/>
- [148] European Cooperation for Space Standardization, “Space product assurance—Software product assurance,” ECSS-Q-ST-80C Rev. 2, Apr. 30, 2025. [Online]. Available: <https://ecss.nl/standard/ecss-q-st-80c-rev-2-software-product-assurance-30-april-2025/>
- [149] International Organization for Standardization, “Road vehicles—Safety and artificial intelligence,” ISO/PAS 8800:2024, Dec. 2024. [Online]. Available: <https://www.iso.org/standard/83303.html>
- [150] UL Standards & Engagement, “Standard for evaluation of autonomous products,” UL 4600, 3rd ed., Mar. 17, 2023. [Online]. Available: https://www.shopulstandards.com/ProductDetail.aspx?productId=UL4600_3_S_20230317
- [151] European Union Aviation Safety Agency, “Artificial intelligence concept paper: Guidance for Level 1 & 2 machine-learning applications,” Issue 2, Mar. 2024. [Online]. Available: <https://www.easa.europa.eu/en/document-library/general-publications/easa-artificial-intelligence-concept-paper-issue-2>
- [152] European Union Aviation Safety Agency, “Artificial intelligence concept paper—Proposed Issue 3,” consultation document, Jun. 3, 2026. [Online]. Available: <https://www.easa.europa.eu/en/document-library/general-publications/easa-artificial-intelligence-concept-paper-proposed-issue-3>
- [153] NASA Earth Science Technology Office, “Technology readiness levels.” Accessed: Oct. 15, 2026. [Online]. Available:

<https://esto.nasa.gov/trl/>