Foundation Models for Autonomous Space Robotics: Architectures, Physical Adaptation, and Validation

Zizhe Wang[1], Jia Deng[1], Xin Lu[1], Zehao Jia[1], and Yao Zhu[2]

Astroup Lab[1] | astroup.com

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

[2]Harbin Institute of Technology
Beijing Research Institute

ee_zhuy@zju.edu.cn

Abstract

Space robots must interpret tasks and act despite unfamiliar dynamics, difficult sensing conditions, and limited opportunities for human intervention. Foundation models offer reusable perceptual and behavioral knowledge, but their value depends on how that knowledge reaches physical execution. This critical review follows space-specific requirements through perception, prediction, planning, and action to deployment and validation, emphasizing orbital manipulation and proximity operations. Terrestrial comparisons establish benefits of particular pretraining stages and model interfaces; their transfer to space dynamics and constrained onboard operation remains insufficiently tested. Event cameras offer timely motion observations across strong illumination contrasts, while spiking neural networks provide possible routes to sparse local computation. We examine how sensor degradation, state estimation and computing choices affect feedback and recovery. Higher action throughput need not yield fresher feedback; lower processor power can also coexist with higher energy per inference. We organize validation as an iterative sim-to-ground-to-space pathway, connecting physical tests with telemetry-based diagnosis and closed-loop revalidation. A satellite-capture example links observations to approach, contact and recovery, while a corresponding protocol separates generalization from adaptation with limited target-domain interaction. These comparisons identify what models already contribute and the physical evidence needed to support autonomous space operation.

Keywords: space robotics; foundation models; vision–language–action models; event-based vision; neuromorphic computing; physical adaptation; validation.

1 Introduction and scope

Space robots must interpret tasks and act with limited human intervention. Tracking or control errors can cause collisions; repair may be impossible. Satellite capture couples tracking, approach, contact, and stabilization; surface sampling connects terrain assessment to tool–material interaction; cabin maintenance requires manipulation in confined, shared workspaces. These operations rely on established dynamics, estimation, planning and supervisory autonomy [1, 2]. Foundation models offer reusable knowledge and behavior when task-specific interaction data are scarce. Can reuse remain useful under changed dynamics, sensing and computation?

1.1 Definitions and scope

We define a *foundation model* by broad pretraining and downstream reuse through a shared model or backbone. Size or architecture alone does not establish this status [3–5]. We include space applications reusing identifiable foundation models and robot models with documented pretraining and reuse, assessing transfer to held-out tasks separately.

Vision–language models (VLMs) connect visual observations to linguistic or structured outputs; vision–language–action models (VLAs) extend this interface to actions. World models predict action consequences

[1]Astroup Lab: 向上跃升（北京）科技有限公司.

Table 1: Existing reviews and the questions pursued here. Their emphases overlap; the final column identifies our focus in relation to each body of literature.

Review	Established emphasis	Focus of this review
Robot foundation models [3, 5, 11]	Applications, interfaces, model use and construction	Reuse of pretrained components and evidence of transfer
Space robotics [1, 2]	Autonomy, manipulation and physical requirements	Adaptation through observations, training and execution
AI-SRO [6]	Capability lifecycle: physics, interfaces, resources and validation	Gains, failures and tested conditions; timing and energy demands
Multimodal space robots [7]	Task planning and end-to-end control (abstract inspected)	Joint tests of task transfer and physical variation
Neuromorphic methods [8–10, 12, 13]	Sensing, representations, training and hardware	Roles of event sensors and spiking computation; deployment cost

and qualify as foundation models when their pretraining and reuse support that description. Spiking neural networks (SNNs) use spikes for computation; their generality depends on training and reuse. SNNs, event cameras, task-specific reinforcement learning and conventional control inform adaptation, deployment and evaluation of pretrained components.

We emphasize orbital manipulation and proximity operations, using intravehicular and selected planetary studies to compare shared problems and physical requirements. Remote sensing and scientific question answering enter only when relevant to robot operation. We distinguish stabilized spacecraft, free-floating manipulators, pressurized cabins and lunar rovers.

1.2 Position relative to existing surveys

Existing surveys frame this discussion (Table 1). AI-SRO connects physics, model/control interfaces, simulators, deployment resources and progressive validation [6]. The multimodal space-robot review organizes work by task planning and end-to-end control; our comparison is limited to its original abstract [7]. The closed loop is a shared organizing principle.

We examine how experiments distinguish reused components' contributions from task-specific adaptation and surrounding control, including timing, energy demands, new tasks and physical conditions. Within-study comparisons trace reused components, target-domain updates and tested conditions. Event cameras and SNNs receive particular attention because observation timing and sustained local computation connect model capability to physical feedback [8–10]. We organize validation into a sim-to-ground-to-space pathway and propose a capture protocol separating held-out generalization from adaptation under a fixed interaction budget.

1.3 Literature selection

This critical narrative review considers works available by 14 October 2026. Initial searches were followed by targeted retrieval and source checks on 15 October, retaining the original eligibility cutoff. Surveys seeded citation tracing, and web searches led to primary sources through arXiv, publishers, proceedings, institutions, and space agencies. Selection favored distinct operational interfaces, physical mechanisms, or validation settings; terrestrial studies explain transferable methods. Titles, DOIs, and versions supported deduplication.

The ancillary inventory records the search history, versions, and reading scope of all cited works. Mechanism comparisons use methodological passages; abstract-only sources support only their stated claims. Unresolved candidates and a local unpublished survey were excluded. Purposive selection limits completeness, and differences between tasks preclude pooled performance rankings.

The review proceeds from space requirements through the embodied closed loop to adaptation, validation and cross-study conclusions.

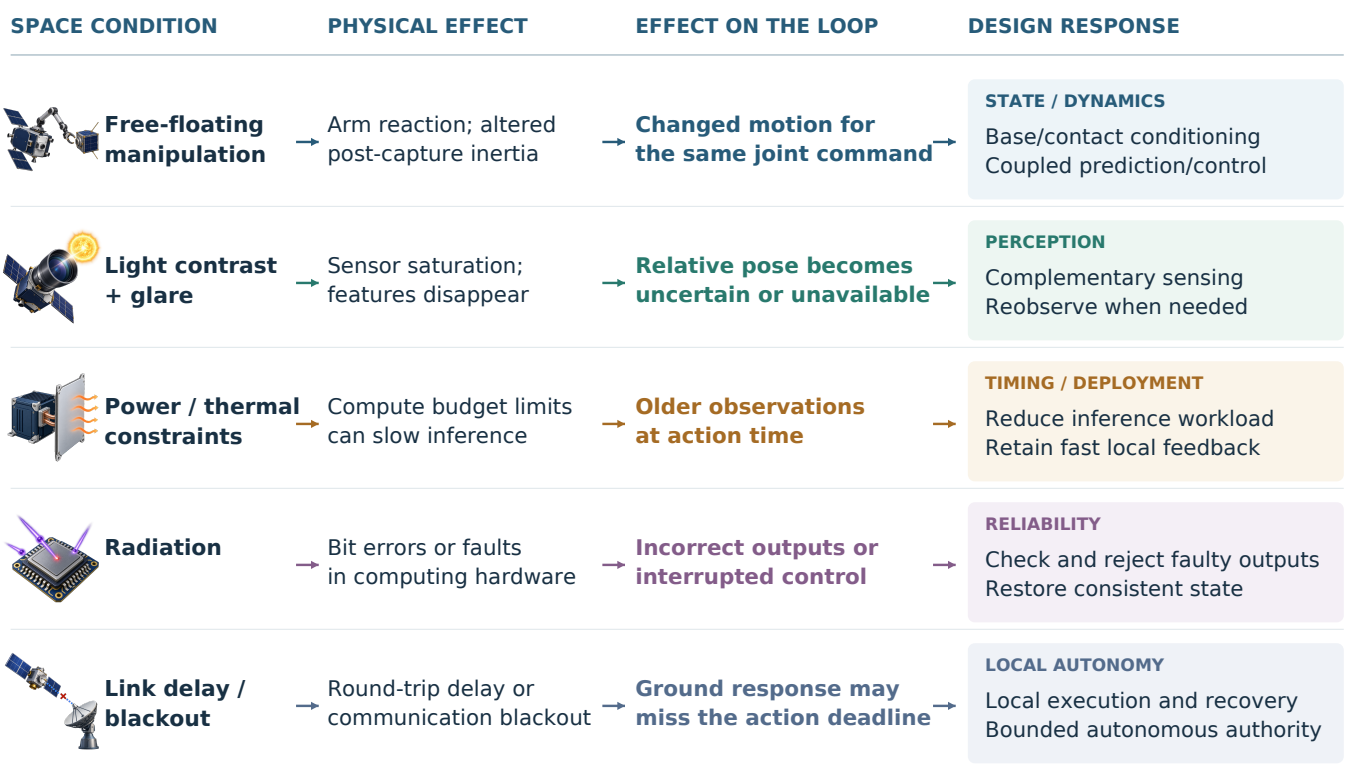

Figure 1: How space conditions change the autonomy loop. Each pathway links a physical or operational constraint to its consequences for execution and to model or control responses. The pathways illustrate mechanisms discussed in this review; their importance depends on the mission and hardware [1, 2, 8, 14, 15].

2 From space operating conditions to technical requirements

Space deployment changes sensing, prediction and control. Figure 1 traces how space conditions change action consequences, observation quality, feedback timing and recovery, motivating corresponding model and control choices.

2.1 Dynamics and contact

A fixed-base terrestrial arm can often treat its base frame as stationary. A free-floating arm exchanges momentum with the base, changing the base attitude, the robot's reachable configurations and the required control effort. Figure 2 illustrates why the same joint motion can produce a different end-effector motion. Capture changes mass distribution and motion. ETS-VII experiments and the manipulation literature establish these effects independently of learned policies [1, 16].

Rendezvous requires relative states and orbital dynamics [17, 18]; capture adds base–target coupling and contact. State and dynamics models must follow the mission stage.

For orbital manipulation, relevant state includes joint configurations, base attitude and velocity, target-relative motion, and contact mode. Active attitude control also introduces actuator and propellant constraints. Task-specific space reinforcement learning illustrates these state requirements [19, 20]. Surface interaction poses a different problem: reduced gravity changes both robot loading and granular-medium behavior. Parabolic-flight wheel–soil experiments found reduced traction and increased sinkage relative to ground tests at matched wheel loads [21]. Reducing the robot's supported weight therefore does not reproduce all relevant soil effects.

2.2 Perception and hardware constraints

Illumination extremes and specular reflections can obscure spacecraft features, as the physical images in SPEED+ illustrate [22]. Thermal changes also perturb measurement geometry: MESSENGER images reveal temperature-dependent focal length [23]. Proton irradiation of CMOS imagers produces increased dark current and switching pixel defects [24]. Missing or displaced keypoints can then bias pose estimates

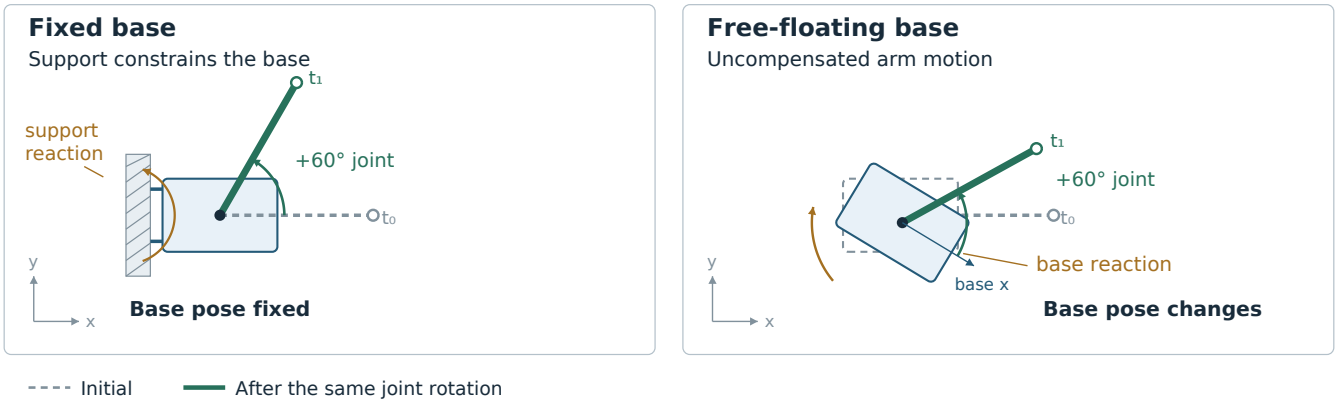

Figure 2: The same joint rotation with a fixed or freely moving base. Dashed and solid geometry denote initial and final poses. Support reactions hold the fixed base; the free base translates and counter-rotates. This planar example starts at rest in a local free-fall frame, neglecting external disturbances and active base control during a short maneuver. The response depends on mass distribution and geometry; a learned policy must account for the resulting end-effector motion [1, 16].

or interrupt tracking. The resulting requirement is to expose calibration state and measurement validity to the estimator and learned policy, alongside coordinate frames and timestamps.

Temperature-dependent calibration, dark-signal correction, and bad-pixel masks address characterized errors; learned compensation should be evaluated by pose and tracking accuracy. Neither calibration nor a plausible reconstruction restores a unique measurement where saturation or occlusion has removed it. The robot may need a new exposure, viewpoint, or modality. Event sensing improves temporal sampling [8], while force, touch, and proprioception supply contact or body-state information [25–27]; their value depends on the state uncertainty and feedback deadline.

Sustained computation depends on power and heat rejection. External vacuum removes ambient convective cooling, making structural conduction and radiation central; pressurized cabins and planetary atmospheres differ. Mounting, conductive paths, duty cycle, and thermal-control capacity determine feasible computation [14]. Model deployment must fit the spacecraft's peak and sustained power, allowable temperature, and task-energy budgets.

Radiation can disrupt computation and storage through single-event effects and accumulated dose [15], requiring error detection, redundancy, recovery, and appropriate devices. Proton and neutron tests measure device responses beyond what image-domain adaptation can establish [28, 29]. Redundancy also consumes power and may remain vulnerable to common failures, so protection and computational efficiency must be assessed together.

2.3 Communication and limited interaction data

Communication delay, jitter, and outages limit reliance on human intervention. SUPVIS Justin demonstrated task-level supervision by ISS astronauts while a ground robot planned and executed locally [30]. Collision avoidance and contact response need local feedback; Section 5 examines the division of authority.

Hardware access and the consequences of failure restrict interaction data. Terrestrial pretraining supplies object semantics and action structure, but representative floating-base contacts and recovery trajectories remain difficult to acquire. Studies should identify physical variations absent from pretraining and the new interaction needed to handle them [3, 31].

2.4 Connecting requirements to a task

Consider satellite capture. Visual pretraining may identify a grasp feature, but its location at contact depends on target-relative motion and observation age. Approach needs pose tracking and uncertainty propagated to the intended action time [18]; closing the gripper adds contact evidence, and stabilization requires the coupled base–target state [1]. Degraded or stale observations can therefore require a slower approach or reacquisition before engagement. Adaptation data and performance across targets assess reuse; contact loads, base disturbance, and stabilization assess execution. This task connects the following architecture to the evaluation in Section 8.

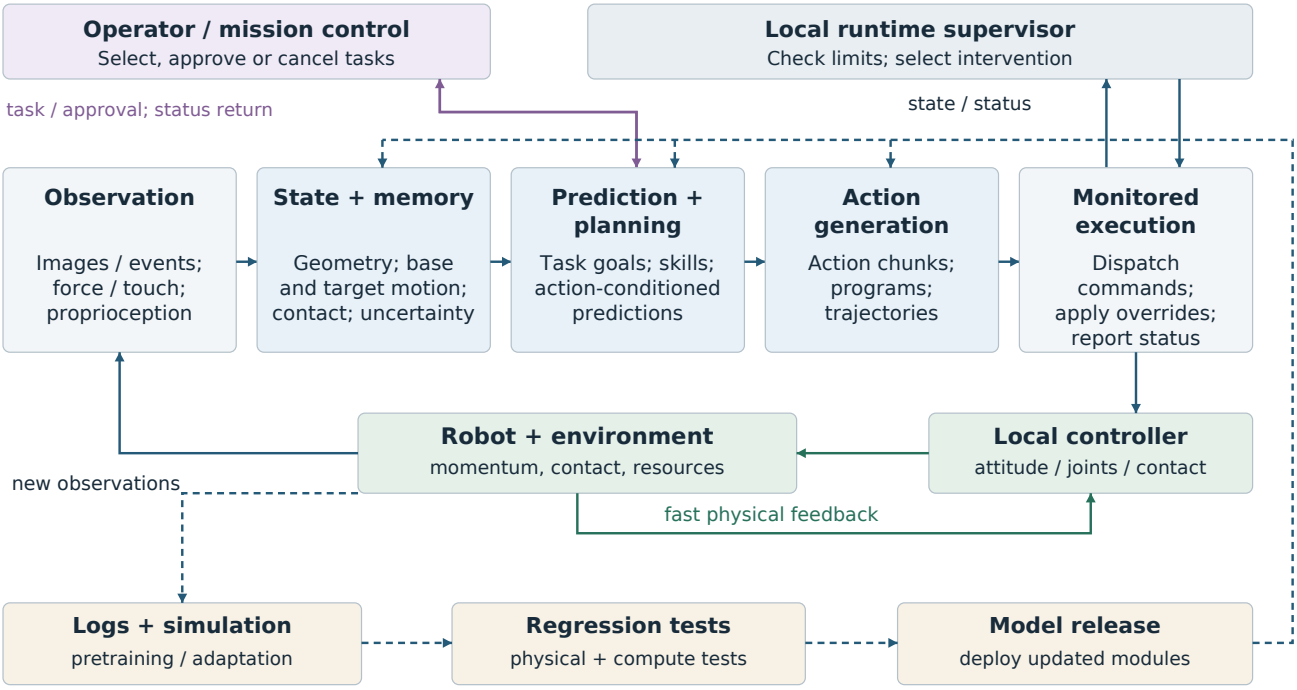

Figure 3: A functional architecture for embodied space autonomy. Operators propose and approve tasks; local runtime supervision monitors execution, and status returns to mission control. Fast physical feedback remains local. Dashed arrows show data collection, offline adaptation, and component release. Blue boxes indicate possible pretrained modules; foundation models may span several functions [30, 33–37].

3 An embodied closed loop for model integration

3.1 Functional interfaces and feedback

Embodied systems act on the environment and revise decisions from subsequent observations. Figure 3 organizes this loop into observation, state and memory, prediction and task planning, action generation, and monitored execution. A unified VLA can span several of these functions, while a modular system can combine pretrained models with conventional estimators and controllers. Inputs, outputs, update rates and feedback paths reveal each model's contribution [11].

Grasp the handle requires locating it, estimating relative motion and detecting contact. The planner needs feedback distinguishing attempted and completed actions. Inner Monologue shows how such feedback can improve recovery, while errors in the success detector can mislead planning [32]. Interfaces should therefore carry observable task status together with timestamps, coordinate frames, and indications of invalid estimates or failed actions.

Foundation models can produce annotations, synthetic training data, or distillation targets offline. Models trained on these outputs need separate evaluation of task performance and resource use.

3.2 How models are coupled to execution

Model integration depends on its outputs, training and execution feedback. A language model may select a complete skill, generate a program, or issue an incremental motion command. A robot policy instead learns an observation-to-action mapping from interaction data. Predictive models can support either approach by supplying training experience or evaluating candidate actions online. These are combinable design choices. Table 3 compares their physical assumptions and feedback responsibilities.

Explicit interfaces expose skill identifiers, programs, motion targets, or trajectory constraints. SayCan scores skills using semantic relevance and feasibility; Code as Policies generates programs that call perception and control routines [34, 37]. Selecting a whole skill delegates its feedback loop, whereas issuing successive motion commands makes model latency part of that loop. Typed choices restrict the available commands, while the executor determines their physical meaning [38]. A learned warm start supplies another interface: it initializes an optimizer that retains explicit dynamics and constraints [36].

Learned robot policies include OpenVLA and π_0, which inherit visual–language components, and Octo, whose backbone is pretrained across robot datasets [33, 39, 40]. Their decoders learn action conventions and observation fusion; fast feedback may still come from a lower-level controller. Calling a robot API alone does not establish this training history or make a language model a VLA.

Prediction can support either interface: UniSim supplies interactive training experience, while Cosmos Policy can use predicted outcomes and values to select candidates online [41, 42]. Table 3 locates physical state and feedback within these routes; Figure 5 follows their implications through a capture attempt.

3.3 Timing, supervision, and model updates

Task reasoning, action generation, and servo control operate at different rates. A policy may execute only part of a predicted sequence before processing a new observation, while a faster controller supplies feedback during execution. Prediction horizon, execution interval, inference delay, and observation age therefore describe different aspects of responsiveness; Section 6 defines them.

The local supervisor can reject or interrupt actions. Abate et al. demonstrate ground intervention using a backup controller to maintain a line-of-sight constraint [43]. Humans select and parameterize tasks [30]; conversational advice remains separate from control [44]. The architecture must specify approval and cancellation authority, status feedback, and permitted local behavior when assistance misses a task deadline.

Updates to model weights or skills alter future behavior and require separate evaluation from recovery with existing capabilities. Operational logs can support adaptation, followed by regression tests before the updated component is released.

4 Multimodal perception and state representation

Planning and control require estimates of target-relative motion, spacecraft-base motion, joint states, and contact [1]. We collect these quantities in a task state, $x_t = (x_t^{\mathrm{rel}}, x_t^{\mathrm{base}}, q_t, \dot{q}_t, c_t)$. Pretrained semantic and geometric features must connect to these physical quantities.

4.1 Connecting semantic features to physical state

Space-LLaVA adapts visual–language knowledge to extraterrestrial imagery and evaluates it on offline image–language tasks [45]; the same paper uses GPT-4o in a separate lunar-rover simulation. To guide motion, semantic features must also support metric estimates. A planetary visual-odometry pipeline uses SAM rock landmarks and DINOv2 descriptors before geometric motion estimation, obtaining metric scale from simulator depth on LuSNAR or stereo disparity on Katwijk analogue sequences [46].

FoundationPose estimates object pose through render-and-compare refinement and ranking, using synthetic pretraining and RGB-D observations [47]. It requires a CAD model or reference views with provided or estimated poses; the latter are used to fit an object-specific neural field. Novel-object transfer thus depends on geometric information; noncooperative targets may still have known geometry.

A specialist pipeline feeds SPNv2 image measurements into an unscented Kalman filter to track relative orbital state, attitude, and angular velocity [18]. It propagates state between measurements and adapts process-noise covariance; SHIRT evaluates it on synthetic and ground-testbed sequences with a known target. For a capture policy, this suggests an interface containing the predicted state, uncertainty, and observation age, so that missed visual updates remain visible to action selection.

PaLM-E embeds images and continuous states into language-token space to generate textual instructions that invoke downstream policies [35]. HPT uses visual and proprioceptive stems, a shared Transformer trunk, and action heads; transfer initializes new stems and heads while freezing the trunk [27]. These methods connect state to language decisions or adapt interfaces around a shared representation.

4.2 Event, spike, and single-photon sensing

During spacecraft approach, motion and illumination contrasts can degrade features just when fresh pose updates are needed. Differential event cameras provide high dynamic range and asynchronous intensity-change measurements [48, 49]. Combining them with image–language priors requires preserving timestamps

through encoding and estimation. Asynchronous readout is distinct from radiation qualification: neutron irradiation produced noise bursts in a tested event sensor, although motion information remained usable in the reported experiment [50]. Its radiation response and tracking benefits require separate evidence.

Figure 4 compares frame, differential-event, integrating-spike and single-photon avalanche diode (SPAD) outputs under changing illumination. Table 2 relates these measurements to model inputs and information limits. Event streams can feed conventional networks, while SNNs can process frame-camera inputs [10]. Section 7 examines spiking computation.

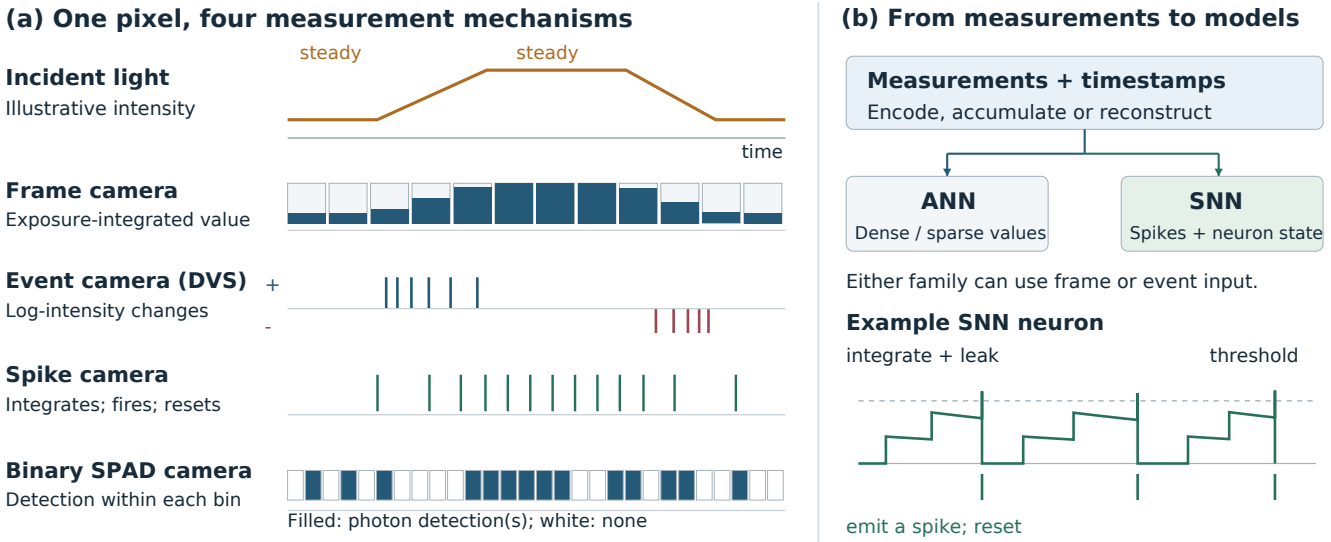

Figure 4: From light to measurements and neural computation. Left: idealized single-pixel outputs share a non-quantitative time axis. Frame-bar height represents integrated light; differential events mark log-intensity changes; integrating pixels emit spikes at an accumulated-light threshold. The passive SPAD512 example records detection within each exposure; other SPAD readouts differ. Right: suitable encodings connect measurements to either network family; a leaky integrate-and-fire neuron illustrates state accumulation and spike emission. Traces are explanatory illustrations [8, 10, 13, 51].

Table 2: Four sensing mechanisms, input representations and information limits. Sensor outputs do not determine the neural-network implementation.

Sensor	Measurement	Model input and information limit
Conventional frame camera [8, 51]	Light integrated over an exposure, then read as intensity values	Image tensors; motion during exposure can blur features; saturation loses intensity detail
Differential event camera (DVS) [8]	Thresholded log-intensity changes with location, timestamp and polarity	Event tensors/tokens or reconstructed images; unchanged intensity is not directly measured
Intensity-integrating spike camera [13, 52]	Intensity accumulated to a threshold; each spike resets the integrator	Temporal spike encoding or reconstruction; intensity requires spike history
Passive SPAD camera (SPAD512) [51, 53]	Binary value per pixel/exposure: no detection or at least one	Binary-frame accumulation estimates intensity; one exposure does not resolve photon multiplicity

Accumulating differential events into frames allows reuse of image encoders, at the cost of temporal compression over an observation window [9]. EventBind instead adapts CLIP through temporal event encoding and prompts, aligning event, image, and text embeddings contrastively [54]. EventGPT combines spatiotemporal aggregation with event–language adaptation to generate language responses [55].

E-VLA converts aligned event windows into frame-like inputs, using either an RGB overlay or adapters inside a frozen visual encoder; the resulting tokens condition a SmolVLA action expert [56]. Event-VLA instead uses action queries and gated cross-attention to select event tokens while retaining RGB–language priors [57]. Both are evaluated in terrestrial manipulation under degraded visibility. Spacecraft evidence includes the SPADES event-camera dataset [48] and S2E2, a compact SNN that estimates pose using a custom split of real ground-testbed recordings from the SEENIC dataset [58].

For intensity-integrating cameras, SSDEFormer encodes temporal spike accumulations and uses a Transformer for stereo depth estimation, evaluated on synthetic data and real PKU-Spike-Stereo recordings [52]. This specialist terrestrial method offers a candidate geometric input for robot policies; space-robot control remains untested.

In lunar-analogue experiments, Fast Vision in the Dark compares SPAD512 and conventional-camera imagery from a Leo Rover at LunaLab [51]. Pretrained SAM ViT-H is tested on one selected scene under several lighting conditions; ORB-SLAM3 is evaluated on recorded rover trajectories. The teleoperated recordings come from SPICE-HL3, which also provides conventional visual and motion measurements [53]. Integrated photon frames permit reuse of image models, but lower bit depths can fragment segmentation, and processing raw binary streams introduces bandwidth and representation challenges [51, 53].

4.3 Force, touch, and proprioceptive encoding

Force–torque sensors measure interaction wrench, while visuotactile sensors encode local deformation through images. AnyTouch learns representations across tactile sensors through shared tokenization, masked modeling, multimodal alignment, and cross-sensor matching [26]. FuSe addresses a complementary problem: a pretrained policy may ignore a newly added sensor. It supplements imitation learning with contrastive and language-generation objectives to connect heterogeneous observations to task semantics [59]. Together, these methods distinguish making sensor representations compatible from ensuring that a policy uses the information they contain.

For a floating robot, interpreting wrench also requires the sensor frame, calibration, joint motion, and base response. These quantities must be available to the policy or to the controller that processes force measurements. Section 6 follows this information into contact-sensitive execution.

Perception supplies semantic descriptions, geometric estimates, and contact or body state. Their calibration, validity, and timing determine how policies can use them. Semantic confidence cannot substitute for pose uncertainty.

5 Prediction and task planning

Prediction estimates action consequences, task planning selects capabilities and their order, and motion planning computes feasible trajectories. Physical constraints enter through learned predictions, structured task descriptions, or the planner that turns them into motion.

5.1 World models: prediction targets and control roles

Predictive models generate training experience, support action learning, or evaluate candidate actions during execution. DreamerV3 learns latent dynamics and trains a policy on imagined trajectories; its generality reflects a shared algorithmic configuration across tasks rather than a reusable pretrained model [60].

UniSim learns action-conditioned future observations through video diffusion and uses simulated interaction for policy learning [41]. DreamGen fine-tunes an image-to-video model to generate robot videos from an initial frame and instruction, then recovers pseudo-actions for a separate policy [61]. Learning can inherit transition or action-inference errors.

V-JEPA 2-AC predicts future representations rather than images [62]. It freezes a pretrained video encoder and learns an action-conditioned predictor from robot trajectories. A receding-horizon planner selects actions toward a goal representation, executes the first, and replans. Franka pick-and-place experiments supply two intermediate goal images before the final goal and switch between them on a fixed schedule. The reported single-RTX-4090 configuration takes 16 seconds to compute an action. Camera sensitivity, externally supplied subgoals and planning time shape demonstrated autonomy. Space applications additionally need representations preserving relative motion and contact information.

GR-2 learns autoregressive video prediction before jointly predicting images and action trajectories with a conditional variational action decoder [63]. Cosmos Policy adapts a video model to predict actions, future states, and values [42]. Its direct-policy evaluation discards state and value predictions, whereas its planning mode assesses candidates with them. With equal numbers of gradient updates, pretrained initialization outperforms random initialization (Table 6).

A specialist continual-adaptation method updates learned dynamics after robot failures [64]. It freezes DreamerV3's observation encoding, decoding, and reward prediction while updating recurrent and latent-transition components; the actor–critic learns from the revised imagined trajectories. In simulations of wheel failures, thruster failures during microgravity navigation, and screwdriver misalignment, recovery is partial and not always sustained. Capture additionally requires representations of base motion and changing contact modes accurate enough to predict impulses and constraint violations [1].

5.2 Grounding language in available capabilities

SayCan scores skills, PaLM-E conditions language reasoning on observations, and Code as Policies calls perception and action APIs [34, 35, 37]. Inner Monologue uses execution feedback to revise plans [32]. VoxPoser composes spatial affordance and avoidance maps for a motion planner [65]. It uses known robot dynamics and mostly assumes static scenes, replanning end-effector paths from observations. Its reusable spatial objective interface would need a planner predicting target motion and base–arm coupling while enforcing whole-body constraints for orbital capture.

LLM-KGFP organizes lunar operational knowledge in a graph and revises plans through feedback in lunar south-pole simulation [66]. SpaceMind combines a VLM, skill instructions, tool calls, and execution memory [67]. Its skill evolution revises task-specific failure-recovery instructions rather than motor weights; it shows no cross-task skill transfer or benefit where baselines already succeed. Evaluation comprises 174 UE5 runs and 18 planar ground-robot runs, without orbital dynamics. Rendezvous succeeded in 6/6 ground trials and search and approach in 4/6; one of six inspection runs collided. The collision motivates continued monitoring after command acceptance: intended motion or mechanical engagement must occur and constraints remain satisfied.

5.3 From mission specifications to feasible motion

OrbitTAMP maps language to orbital motion [17]. A language model parses instructions into a partial *MissionConfig*. Behavior-duration planning completes the behavior sequence, waypoint planning supplies missing relative-state coordinates, and trajectory optimization computes state and control trajectories. Constraints can include flight time, Δv budget, and keep-out-zone dimensions. Physics enters through the planning representation and optimizer.

With downstream planning fixed, held-out tests compare language-model parsers against an ontology-aware rule parser (Table 6). Their relative performance changes with the instruction style. Separately, two verifier-guided revisions raise Qwen3.5-9B's exact recovery from 75% to 88% on 100 development commands; five revisions add no benefit. Structurally valid semantic errors can escape verification, and the version-1 trajectories are nominal and open-loop. A converged optimizer solves the encoded problem, which may still differ from the intended mission.

SELENE specifies a complementary interface through Atomic Action Contracts, connecting semantic planning to VLA, RL, or optimization-based executors [68]. Preconditions, goals, safety envelopes, and validation metadata define when a skill may run and what it must achieve. Tests cover ground LunarBot manipulation, simulated lunar-gravity traversal, and offline skill instantiation and registration. They do not yet demonstrate detecting a missing capability at runtime, creating a new skill, and executing it. Whereas OrbitTAMP completes a numerical mission specification, SELENE describes how existing capabilities are selected and called.

5.4 Autonomous decisions and requests for assistance

Planning also allocates human supervision. In SUPVIS Justin, ISS astronauts selected task commands and parameters for Earth-based Rollin' Justin, which planned and executed them locally [30]. NealAI assists astronauts supervising a ground robotic team [44]. NASA's Perseverance account describes another allocation: Claude-assisted planning on Earth, digital-twin checking before uplink, and execution on Mars [69]. By comparison, AEGIS selects scientific targets onboard through a specialized system [70].

KnowNo uses held-out conformal calibration to form sets of candidate next steps; non-singleton sets trigger assistance [71]. Its guarantee applies to semantic choices under the calibrated scenario distribution and help mechanism. Physical uncertainty and execution deviations require responses such as renewed observation or

controller-level recovery, including permitted local action during communication delays. These extensions require testing and intervention accounting beyond KnowNo's semantic guarantee.

6 Action generation and physical execution

Joint displacements, end-effector targets, throttle vectors, and skill identifiers have different physical meanings despite sharing numerical encodings. Transfer requires matching the action representation to the robot's controller.

6.1 Direct commands and learned action policies

Language models can issue motion commands through structured interfaces. GPT-3.5 and LLaMA are fine-tuned on telemetry to return throttle-vector function calls in KSPDG spacecraft pursuit–evasion simulation [72]. With fixed weights, GPT-6-Astra instead uses images and joint feedback to command XLeRobot and generate local feedback programs [73]. Machine-readable geometry and synchronized experience reduce completion time; the twelve physical trials use operator-confirmed button contact as their endpoint.

A broader Astra study compares direct end-effector commands with a hybrid that accepts or corrects $\pi_{0.5}$ proposals [74]. On 50 paired RoboDojo simulation instances, sampled mainly from tasks with low published $\pi_{0.5}$ success, hybrid control completes 24 versus 13 for direct control; 85.6% of its execution steps come from the learned policy. Here $\pi_{0.5}$ is task-finetuned; on RoboLab, where Direct performs better, the hybrid uses a DROID-trained checkpoint zero-shot. The ranking therefore reflects both task requirements and the fit of the reused policy.

Typed decision models offer a different interface. In a planar-arm simulation, Jev selects discrete torque levels or evaluates code-generated skill outcomes [38]. Raw torque selection completes 0/20 episodes on the native task, compared with 9/20 using skills and external simulation. The comparison changes both action abstraction and predictive support: it does not show that Jev itself has learned a dynamics model. Command granularity and the source of physical prediction therefore matter alongside model identity.

Robot policies learn action mappings directly from trajectories. RT-2 predicts action tokens [75]; OpenVLA combines DINOv2 and SigLIP features with Llama 2 and discretizes each action dimension into 256 bins spanning training-data quantiles [33]. Scaling and command semantics determine how those tokens move the robot.

Diffusion Policy learns conditional denoising of continuous action sequences [76]. Octo places a small diffusion head after a pretrained Transformer, so repeated denoising need not rerun the backbone [39]. In π_0, PaliGemma initializes the visual–language component; a separate action expert processes robot states and noisy actions, generating a chunk by integrating a learned conditional vector field [40]. Different training targets and repeated computations affect both latency and energy use.

Table 3 compares where these routes obtain physical behavior and how execution returns feedback. A shared model can support several interfaces; their suitability for orbital capture depends on the state and constraints carried through each one.

6.2 Action chunks and feedback timing

Let $A_t^H = (u_t, \ldots, u_{t+H-1})$ contain H predicted steps, of which $K \leq H$ are executed before the generating policy incorporates a new observation. Faster decoders or controllers may receive feedback within that interval. If an observation is acquired at t_o and its first resulting command issued at t_c, its age is already $t_c - t_o$; actuator response adds further delay. Commands issued through model services incur request and response delay. Astra's local programs execute feedback loops between model calls; Jev's torque-step duration specifies simulated motion time [38, 73].

These timescales matter independently of action encoding. Cosmos Policy's direct ALOHA evaluation has a 0.95-second blocking inference pause on one H100, followed by 50 executed steps at 25 Hz, or two seconds of execution [42]. FAST compresses action sequences into tokens but does not determine the feedback interval [77]. Shorter execution chunks help only when new actions arrive in time.

Real-time chunking (RTC) overlaps inference with execution, preserves the actions committed during inference delay, and inpaints the remaining chunk with softly weighted overlap [78]. Its Kinetix ablation favors soft over hard masking particularly at lower delays; robot tests use terrestrial bimanual manipulation.

Table 3: Physical assumptions and feedback responsibilities across model interfaces. Orbital adaptations are proposed extensions of the methods discussed in Sections 5–6; prediction can support any action interface.

Output and examples	Source of physical behavior	Feedback responsibility	Orbital adaptation
Numerical targets: Astra [73, 74]	Robot API and servo; model uses observed state	Local servo/program; observations return to model	Base-aware frames; coupled limits; observation age
Skill or program calls; typed choices: SayCan, Code as Policies, Jev [34, 37, 38]	Existing skills/controllers; external simulation in Jev skill tests	Executor closes local loop and returns status	Coupled-dynamics skills; checked preconditions and recovery
Spatial objectives or constraints: VoxPoser, OrbitTAMP [17, 65]	Explicit robot/orbital dynamics in planner	VoxPoser replans; cited OrbitTAMP trajectories are open-loop	Target prediction; whole-body constraints; execution feedback
Learned action policies: OpenVLA, Octo, π_0 [33, 39, 40]	Robot trajectories and lower-level control	Policy updates; lower-level control between updates	Base/contact state; adapted training; constrained execution
Latent/video futures: V-JEPA 2-AC, Cosmos Policy [42, 62]	Transitions learned from robot trajectories	Planner/policy selects; observations update prediction	Momentum/contact representation; calibration under changed dynamics

RTC improves continuity between chunks, whereas SpikingVLA overlaps computation inside a network [79]. Physical execution additionally needs an interrupt when contact assumptions fail.

6.3 Incorporating space dynamics

Free-floating arm motion reacts on the spacecraft base, as demonstrated by ETS-VII's generalized-Jacobian and reaction-related control experiments [16]. Reproducing a terrestrial end-effector trajectory therefore changes base motion and control-resource use. Capture adds the target to the coupled system: contact can dissipate kinetic energy, but internal forces cannot remove its total momentum [1]. Retaining a grasp and stabilizing the combined motion consequently require more than reaching the predicted contact pose.

Adaptation can enter at three locations [1]. *State conditioning* gives the model relative motion, base attitude and velocity, and joint state. *Training adaptation* exposes it to changes in inertia, momentum, and support conditions. *External control* interprets or modifies proposed targets using coupled dynamics and constraints. For explicit targets and skills, physical adaptation can modify state descriptions and executors; learned action mappings and transition models must also be checked against the new dynamics.

SpaceOctopus combines multi-arm policies with a lower-level PD controller in MuJoCo [19]. Prior-policy-guided learning uses inverse-kinematics knowledge for dual-arm free-floating motion planning while varying base mass and target rotation [20]. These specialist systems show where relevant state and dynamics enter the policy–controller combination. The latter stops short of post-contact force control, so extending its approach to stabilized capture requires a further contact-control stage.

6.4 Contact sensing, action updates, and compliant control

Contact information can influence execution at several levels. ForceVLA conditions π_0 action decoding through a force-aware mixture of experts, using tool-center-point pose, gripper width, and a six-axis external wrench expressed in the world frame [25]. FTP-1 lets action queries attend to tactile and vision–language streams, with shared tactile processing and sensor-specific encoders trained on target demonstrations [80]. These mechanisms make contact information available to action prediction.

Reactive Diffusion Policy (RDP) updates actions during a chunk: slow latent-action diffusion is paired with a fast GRU decoder receiving the latest tactile or force observation [81]. This specialist imitation-learning method demonstrates feedback within ground contact tasks, complementing RTC's continuity between chunks. It does not impose a force bound.

Feedback control can complement these policies by shaping the robot's response to contact. Impedance control relates motion error to wrench, while admittance control converts wrench into motion adjustments [1]. For floating robots, both must account for base response and changing contact constraints. ETS-VII used delayed-image compensation and speed limits for approach, followed by force-feedback compliance, capture, and braking [82]. The target carried visual markers and moved within the clearance of a partially released

docking mechanism; chaser attitude control was disabled. This orbital experiment illustrates the transition from visual approach to controlled contact.

6.5 Constrained execution and runtime monitoring

The execution layer checks command frames, units, limits, and observation age before acting. GR-2's ground deployment feeds optimized Cartesian trajectories to a 200 Hz whole-body controller [63]; this rate is distinct from model inference frequency. In an ISS experiment with Astrobee, a specialized neural network initializes trajectories and sequential convex programming handles the adopted dynamics and safety constraints [36].

Runtime assurance adds a mechanism for intervention. Abate et al. combine reachable-set overapproximations with a backup controller to enforce a line-of-sight constraint, validated on the ground ASTROS platform [43]. The guarantee depends on the adopted dynamics, bounded disturbances, actuator limits, and verified backup set. Estimator covariance [18] does not itself provide the required disturbance bound. Applying this approach to satellite capture would require corresponding contact and momentum models and validated recovery.

For an approach, clearance must accommodate pose error, closing motion during sensing and computation, tracking error, and braking distance. Faster event acquisition helps only if encoding, inference and command execution preserve that timing benefit. As the remaining margin shrinks, the supervisor must reduce speed or obtain another observation while recovery remains feasible. After contact, the relevant conditions change to grip retention, contact load and combined motion. Figure 5 connects these decisions across one capture attempt; Section 8 turns them into staged measurements and transfer tests.

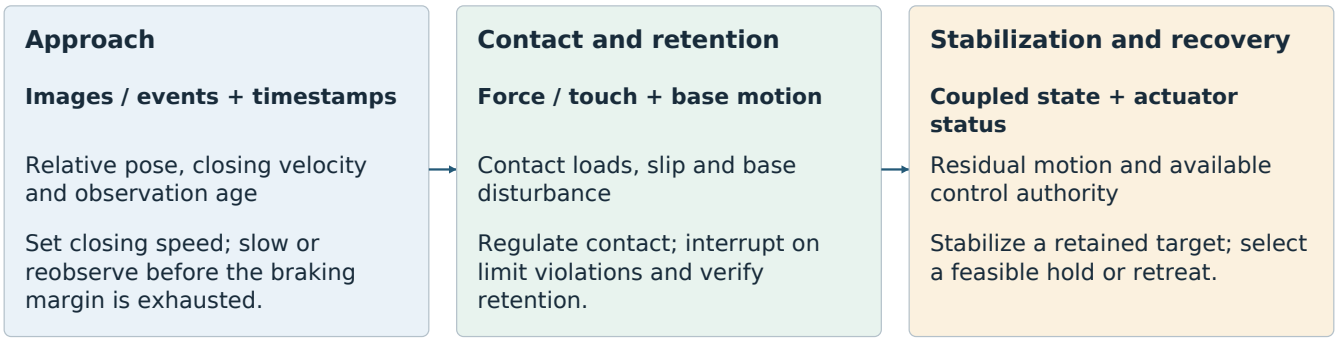

Figure 5: A proposed capture example linking observations to execution and recovery. The phases draw on orbital approach/contact control and supervised execution [18, 43, 82]; the combined foundation-model system is a design synthesis. Image semantics can guide target selection, while geometric and contact estimates determine admissible motion. A hold or retreat must remain feasible under the current dynamics and contact constraints.

7 Learning, adaptation, and onboard deployment

7.1 Choosing what to adapt

Space adaptation addresses appearance, sensing and action consequences, with uneven data coverage: Open X-Embodiment contains diverse terrestrial trajectories but does not cover floating-base interaction [31]. Space-LLaVA adapts visual–language knowledge and SPADES supplies spacecraft events; neither provides complete servicing trajectories [45, 48]. DreamGen illustrates how generated videos broaden visual scenarios, while physical consistency and executable actions still require checking [61].

Octo changes observation and action interfaces; OpenVLA limits trainable parameters through low-rank adaptation [33, 39]. New sensors or command conventions may require interface updates; changed inertia or contact may require dynamics or policy adaptation. Section 8 describes comparison controls. Ground training, pre-mission adaptation and onboard updates also have different computing and data budgets.

7.2 Reducing deployment cost

A VLM can supply spacecraft pseudo-labels for a smaller detection or segmentation model, leaving only the student at deployment [83]. The cited study evaluates image datasets. Retaining the backbone preserves its

inference and activation-memory costs, even with low-rank adaptation. Astrobee's specialist flight policy uses two 64-unit hidden layers within a sub-1 MB policy budget, separate from total avionics memory [84].

Physical Jetson AGX Orin profiling gives QwenVLA/OpenVLA-OFT peak memory of 7.39/19.20 GB, with separately evaluated LIBERO success of 78.8/95.3% [85]. Chunk latency is measured over 100 fixed-image/prompt queries under several power modes. BitVLA combines a pretrained ternary language backbone with quantization-aware visual distillation, reporting 1.4 GB model memory and 96.0% LIBERO success [86]. Its 73 ms per 25-action chunk is measured on A100; physical manipulation uses remote A800 inference. Thus compression, runtime memory and response time require separate comparisons on the intended hardware.

Inference optimization introduces a further tradeoff between output throughput and fresh feedback. In the OpenVLA-OFT ablation, parallel decoding raises A100 throughput from 4.2 to 15.9 generated actions/s; adding eight-action chunking raises it to 108.8 actions/s [87]. Query latency nevertheless increases from 62.9 to 73.5 ms when chunking is added. These measurements average 100 queries and concern ablation variants, not the final continuous-action L1 configuration. Because each chunk executes fully before replanning, more generated actions do not imply faster responses to new observations. FAST likewise improves action encoding without measuring onboard energy savings [77].

7.3 Spiking computation for local autonomy

SNNs are relevant when a robot must sustain local sensing or control within power and heat limits during communication gaps. Stateful neurons and sparse communication offer a route to compact temporal processing [9, 10]. One possible design couples a local spiking estimator or controller to a slower foundation model, using each at the rate its task requires.

Implementation can convert ANN activations into spikes or train spiking dynamics directly with surrogate gradients [10]. Both routes require matching neuron operations, precision, connectivity and communication to the processor [9].

Quantized keypoint networks on Akida feed geometric PnP pose recovery using synthetic SPADES data and ground-truth crops [49]. For event-to-frame (E2F) input, physical V1 reports 0.919 W dynamic power at 16 FPS, versus 2.519 W at 330 FPS for the floating-point Jetson Orin Nano baseline. Dividing power by throughput gives 57.4 versus 7.63 mJ/frame: lower power accompanies higher dynamic inference energy at these operating points. On the full 35,889-input test set, quantization raises the E2F mean pose score (lower is better) from 0.0361 to 0.1009 and rejected poses from 1 to 162; those means exclude rejections. Hardware benchmarking uses 4,000 inputs per representation, distinct from the full-test accuracy comparison. V2 uses a heatmap architecture evaluated in cloud simulation, with higher-frequency throughput projected. Whole-system task accounting must also specify sensing, target detection, frame accumulation, PnP, host and idle costs.

For a wall-mounted cabin manipulator, a directly trained spiking actor–critic combines geometric states, tactile features and semantic cues; a PPO curriculum progresses from approach to grasp and lift [88]. These Isaac Gym experiments test the complete policy, while energy remains an operation-count estimate.

Terrestrial VLAs extend this approach. SpikeVLA converts vision and language components and trains a population-coded action network with PPO and surrogate gradients [89]. Spike-driven VLA combines trainable spiking vision, frozen spiking BERT, token fusion and state-conditioned action chunking [90]. With fusion and decoding fixed, replacing its ANN encoders with spiking encoders lowers LIBERO success from 96.3% to 92.4% and estimated arithmetic energy from 179.7 to 20.9 mJ per twelve-action chunk. The estimate includes the frozen encoder but excludes memory and other hardware overhead; encoder capacity and pretrained priors also change.

SpikingVLA converts NaVILA using dendritic channel mixing and adaptive firing [79]. Internal steps propagate asynchronously but process the same observation; eight rather than four improve navigation while increasing estimated energy. All three use simulation and operation-based estimates, requiring hardware measurements.

7.4 Continual learning and recovery

MuSe uses replay during force-feedback adaptation, with storage and data-provenance costs [91]. The space approach in Section 5 selectively updates components after simulated failures; its authors report that online optimization exceeds the power constraints of space-grade embedded computing [64].

A changed payload may require only a revised estimate, without backbone retraining. Ground updates allow regression testing, while onboard training competes with inference. After rollback, restored parameters, recurrent memory and state estimates must be reconciled with the robot's current physical state. Evaluation should measure retained and regained performance, update cost and switching behavior, beyond eventual policy reward.

7.5 Thermal operation and energy measurement

In vacuum, heat leaves onboard electronics through conduction and radiation [14]. Thermal throttling or reduced duty cycles can slow inference and leave controllers acting on older observations. Deployment therefore requires sustaining the task's feedback rate within thermal limits.

Memory access, routing, neuron-state maintenance and input bursts also consume energy [9]. NeuroBench separates algorithm complexity from system measurements, including preprocessing and postprocessing [92]. Table 4 distinguishes these measurement boundaries, from arithmetic estimates to proposed complete-task comparisons.

Table 4: Resource and energy evidence with different quantities and measurement scopes, not a cross-platform ranking. The final row proposes complete-task accounting. Akida comparison values use dynamic power divided by throughput and exclude idle energy.

Evidence type	Example	Quantities and units	Exclusions or conditions
Runtime profiling	Jetson AGX Orin [85]	Peak memory (GB); chunk latency; 15/30/50 W and MAX modes	Power modes specify limits, not measured draw
Arithmetic estimate	Manipulation SNN [88]	Operations; assumed J/operation; estimated J/inference	No measured memory, routing, or idle cost
Power-derived inference energy	Akida V1 / Jetson Orin Nano [49]	E2F: 57.4 / 7.63 mJ/frame, from dynamic power / FPS	AKD1000 chip / Jetson board; different precision and operating points
Measured board power	Nahuku, preprint Table 2 [93]	Idle: 0.935–0.938 W; running-minus-idle: 7–12 mW	Preloaded events; excludes streaming I/O and preprocessing
Proposed task accounting	Matched-task test	Integrated energy (J); J/completed task; completion rate	Declare sensors, host, recovery and actuators included

The hardware configuration matters. Nahuku measurements use preloaded inputs, whereas the flight system used Kapoho Bay; the running-minus-idle increment is distinct from total power and the study's separately defined dynamic-power component [93]. Task-level accounting should include failures and idle periods and specify actuator coverage.

7.6 Radiation tolerance and model robustness

Radiation can corrupt computation even when the input is familiar. Avionics protections address different failures and impose different costs [15]. Error-correcting codes cover bounded storage-error patterns; scrubbing consumes bandwidth and leaves intervals between checks; watchdogs detect stalled execution but may miss incorrect outputs. Replication adds power, synchronization and voting costs while remaining vulnerable to common failures. RedNet instead allocates protection according to layer sensitivity and spatially correlated upsets in satellite DNN workloads [94]. Its payload and radiation-emulator results motivate selective protection as an alternative to uniform redundancy.

Vision-transformer studies identify vulnerable modules through soft-error experiments, including bit-flip injection [95, 96]. Applying these results to control requires matching the fault model to the device and tracing output errors into actions. Compression or adaptation can change numerical sensitivity and fault propagation, requiring protection reassessment. Section 8 connects irradiation and closed-loop tests to the operational questions: is corruption detected before actuation, and does restart restore coherent control?

8 Simulation, physical validation, and evaluation

Validation must connect model behavior to the hardware, environment and task stage.

8.1 From benchmark performance to operational claims

Existing benchmarks cover different parts of the autonomy problem. CALVIN tests language-conditioned manipulation sequences and LIBERO studies transfer and continual learning [97, 98]; neither represents free-floating reaction motion. SIMPLER examines whether simulation reflects physical policy performance, a relationship that needs reassessment when inertia or contact changes [99]. Space perception resources provide complementary tests: SPEED+ addresses synthetic-to-laboratory visual transfer, SHIRT supports temporal pose tracking, and SpaceSense-Bench broadens synthetic multimodal perception [18, 22, 100]. For task-level simulation, Space Robotics Bench supplies learning environments and Basilisk supplies modular astrodynamics [101, 102]. Their usefulness depends on whether the implemented mechanisms reproduce those governing the chosen task.

Event simulation also needs sensor models. ESIM couples adaptive rendering with event generation [103]; v2e models threshold mismatch, finite intensity-dependent bandwidth and noise [104]. ELOPE combines PANGU imagery and v2e events with range and inertial measurements for lunar-descent estimation before touchdown [105]. A closed-loop alternative regulates thrust using optical-flow divergence from tracked events [106]. Success means reaching 50 m altitude; near-surface oscillations and unrealistic thrust requests remain. Both routes depend on rendering cadence, illumination, thresholds, noise and timestamps; event generation cannot recover information absent from source images.

Sensor tests measure event statistics and motion-estimation error; task tests reveal their effects on approach or recovery. Timing must include accumulation, buffering and execution, which can consume asynchronous sensing's latency advantage.

Reaching a grasp pose, establishing contact, retaining a target and stabilizing the coupled system are distinct achievements. The dual-arm learning study stops short of post-contact force control [20], while Lunar-Bench tests reasoning about described scenarios [107].

The spacecraft label-distillation study splits SPEED+'s original test domains for adaptation and evaluation [83], changing the benchmark's untouched-domain protocol. Interpreting transfer therefore requires knowing which images, objects and trajectories were available during pretraining, pseudo-label generation, adaptation and model selection.

8.2 From simulation through ground tests to space

Ground testbeds preserve selected physical mechanisms while introducing their own constraints. SPICE-HL3 provides teleoperated rover recordings under LunaLab's lunar-analogue lighting, supporting optical validation separately from reduced-gravity dynamics [53]. Planar air bearings permit relative motion and in-plane coupling; spherical arrangements permit attitude motion within balancing and travel limits [1]. Support hardware affects the resulting dynamics. Correcting its added inertia improves manipulator vibration identification, while friction and air resistance still affect damping [108]. A transferred policy may consequently learn to compensate for the apparatus as well as the intended robot dynamics.

Gravity offloading supports space mechanisms during ground tests by compensating for their weight. The cable-driven Motion Suspension System supports three-dimensional CAESAR arm motion and contact experiments [109, 110], but suspension forces, attachment geometry, tracking error and bandwidth still shape the motion. Robotic hardware-in-the-loop (HIL) testing combines modeled motion with physical sensing or interaction: EPOS supports proximity-operation testing and OOS-SIM includes a real manipulation arm [111, 112]. Here contact stability depends on the rendered dynamics and delay; the cited EPOS stability experiment is one-dimensional [113]. Table 5 identifies calibration quantities that help assess these differences for a chosen task.

We propose an iterative sim-to-ground-to-space pathway (Figure 6). Ground analogues and HIL test selected dynamics, contact, sensor timing and computing costs against simulation. Compare generated events with physical recordings under matched motion and illumination, including buffering and noise; processor HIL connects SNN execution latency and energy to control behavior [114]. Initially freeze the policy to expose transfer discrepancies before adaptation.

Table 5: Validation media, preserved mechanisms and residual differences. Calibration quantities are proposed reporting items for the chosen task; environmental tests address separate conditions.

Medium	Task or mechanism	Calibration quantities	Residual difference	Sources
Digital simulation	Approach, sensing, contact models	Integrator step; sensor clocks; contact parameters	Model omissions and numerical error	[101–103]
Air bearings	Planar capture or attitude tests	Support inertia; drag; level/centre-of-mass offset	Constrained degrees of freedom	[1, 108]
Gravity offloading	Spatial arm motion and contact	Residual load; tracking error; bandwidth	Suspension constraints and forces	[109, 110]
Robotic HIL	Relative motion and physical contact	Delay; tracking; stiffness/damping	Rendered dynamics and workspace limits	[112, 113]
Processor HIL	Real inference with simulated robot	Execution latency; I/O; numerical format	Simulated embodiment and environment	[114]
Environmental tests	Thermal load or radiation response	Temperature/power; particle exposure/dose; recovery time	Configuration-specific, not task assurance	[14, 28, 115]
Flight experiments	ETS-VII orbital arm; ISS cabin free-flyer	State-estimation error; control timing; disturbances	Different environments and tested tasks	[16, 36]

Diagnosis precedes the update choice: align commands, observations, actuator states and timestamps to separate dynamics mismatch from sensing, timing or facility effects. Distinguish simulator recalibration, ground retraining followed by upload, and onboard weight updates. Historical telemetry replay checks consistency with recorded trajectories; once a changed policy selects different actions, those trajectories cannot establish its closed-loop behavior. Revalidate the revised model or controller in calibrated closed-loop simulation and relevant HIL against both the observed discrepancy and the original operating envelope, using evaluation trajectories separate from calibration or adaptation.

Space demonstrations would progressively test representative task stages. Ground agreement during approach cannot establish contact or stabilization performance in space. Following NASA-STD-7009B, distinguish numerical verification, calibration and empirical validation; report residual uncertainty and extrapolation beyond tested inertia, contact and delay ranges [116].

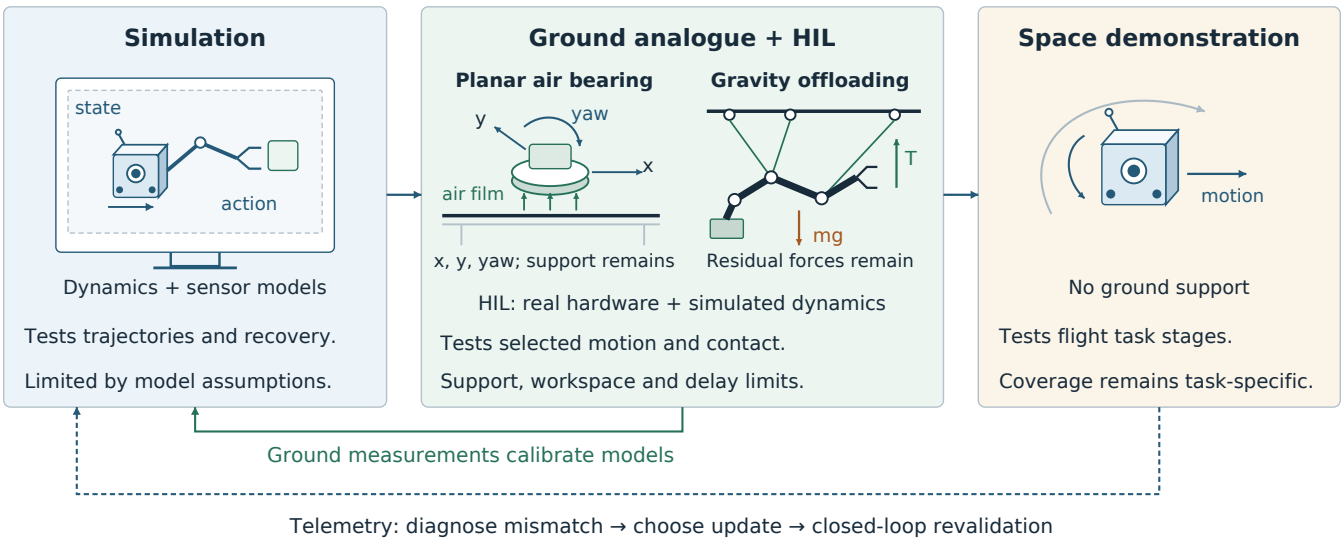

Figure 6: Proposed sim-to-ground-to-space workflow and illustrative test mechanisms. Planar air bearings permit x/y translation and yaw while retaining support; cable tension T counteracts weight mg, with residual forces [1, 109, 110]. Robotic HIL couples simulated motion to physical sensing or contact; processor HIL runs inference hardware with a simulated robot [112–114]. Ground measurements calibrate models; telemetry informs diagnosis, update choice and closed-loop revalidation. Flight evidence remains task-specific.

LunaLab's fully randomized octree-based RL policy achieved 8/25 physical grasps versus 2/25 with reduced randomization [117]. Each was selected from three training seeds, so these episodes do not measure variation across trained deployments. Full randomization reduced training stability; reduced randomization more often produced repeated poorly positioned grasps and timeouts. Simulation used terrestrial gravity

(9.807 m/s^2): this tests visual and manipulation transfer, leaving lunar gravity and free-capture momentum management unresolved.

8.3 Interpreting flight and hardware evidence

Flight and processor tests answer different questions. ETS-VII provides orbital manipulation evidence [16], while Astrobee demonstrates task-specific learned control and trajectory warm starts aboard the ISS [36, 84]. Falcon Neuro adds sensor flight evidence: its ISS-mounted nadir camera recorded Earth's surface in January 2022, allowing coastlines and clouds to be reconstructed from events [118].

LeLaR provides a specialist flight example of discrepancy diagnosis and supervised control. Its InnoCube controller issued three reaction-wheel torque commands at 1 Hz using gyro-integrated relative attitude [119]. Telemetry exposed startup delays up to approximately 20 s and intermittent unresponsiveness near zero speed; comparing commands, wheel speeds and attitude distinguished delayed actuation from measurement spikes. The authors retained the training wheel model to assess robustness against these effects. An independent monitor was configured to stop control and enter command-free idle on limit violations, followed by delayed wheel power-off and operator reset. No monitor intervention occurred in the reported experiments. The idle design relied on InnoCube remaining power-positive while tumbling; a robot in contact may instead need controlled holding or retreat.

The Loihi 2 control study converts a ReLU policy into a quantized sigma–delta network that transmits activation changes [114]. A physical processor exchanges observations and actions with simulated Astrobee in Isaac Lab. It reduces measured inference energy relative to the GPU ANN, but increases final position and orientation errors. This connects deployment efficiency to control quality, while the related ANN's ISS flight remains a separate experiment. ISS cabin operation also differs from external vacuum and contact with uncontrolled satellites.

Figure 7 separates language-task variation, skill reuse and physical execution across tested configurations. SELENE, for example, evaluates physical manipulation and simulated traversal through separate pathways. These capabilities do not form a single readiness score. No thermal-vacuum or irradiation tests of the configurations shown are reported.

8.4 Environmental and resource testing

Thermal-vacuum tests should run representative workloads with the intended conductive and radiative interfaces, recording temperature, power, sustained latency and task behavior to establish how heat affects feedback timing.

Radiation studies illustrate why the device and fault mechanism must also be specified. RedNet uses a hardware-in-the-loop radiation-error emulator [94]; proton irradiation of Tensil reveals service interruption and silent incorrect outputs [28]; ODIN combines neutron exposure with calibrated injection [29]. Loihi has proton single-event-effect and total-dose measurements [115], but these cannot establish tolerance of the Loihi 2 configuration used in the control study [114]. To connect such computing tests to an embodied system, experiments must trace fault detection, rejected commands, recovery duration and the physical consequences of interruption.

Use irradiation data to calibrate device-specific fault models, then inject representative faults during execution. Report exposure, numerical configuration, monitored outputs and interventions: unchanged success can conceal repeated supervisory recovery, while rare errors may cause consequential actions.

8.5 A reusable evaluation protocol

We propose a satellite-capture experiment crossing seen/unseen task compositions with seen/unseen physical conditions. Group tasks by target, grasp site and approach–retreat–recapture sequence, and physics by inertia, target rotation and contact parameters. Evaluate four cells: both groups seen, unseen tasks only, unseen physics only, and both unseen. These labels describe the initial documented development split; report subsequent adaptation exposure and known pretraining overlap or provenance uncertainty separately. Calibrate physical ranges and thresholds against the intended mission.

Use two evaluation modes. *Held-out generalization* fixes the checkpoint across test conditions: report the original model and a version adapted only on the seen/seen set, reserving unseen groups for evaluation. *Target-physics adaptation* allows a separate adaptation set from new physical conditions and measures

Foundation-model applications (first five) and specialist learning configurations (last three)

Configuration	Reuse / task variation	Modeled physics	Robot experiment	Inference / compute evidence	Operational environment
Space-LLaVA	Held-out image-language task	Imagery; no dynamics	Offline model evaluation	Offline inference; flight hardware NR	Offline data
SpaceMind	Task-bound skill reuse	UE5; orbital dynamics omitted	Ground mobile robot (planar)	Ground setup; processor NR	Laboratory
OrbitTAMP	Held-out commands	Numerical orbital motion	Numerical planning	Model inference; flight hardware NR	Simulation
SELENE	Shared skill contract	Lunar-gravity traversal	Ground contact manipulation	Ground-linked inference	Ground + simulated Moon
Claude-assisted Perseverance	Driving workflow	Digital-twin route checks	Mars rover driving	Model on Earth; rover executes	Mars surface
Learned warm start	Trajectory / obstacle changes	Free-flyer dynamics	Ground + ISS free flight	Onboard ARM network	ISS cabin
Astrobee Loihi 2	Pose variations; undocking	Zero-g robot simulation	Simulated Astrobee	Physical Loihi 2	Ground compute + simulation
Manipulation SNN	Reach / grasp / lift stages	Wall-mounted cabin arm	Simulated contact	Operation-based energy estimate	Simulation

Figure 7: Evidence for tested model–hardware configurations. Physical manipulation, free flight and rover driving are distinguished. Mars planning ran on Earth; Space-LLaVA's separate GPT-4o demonstration is excluded. NR denotes unreported information. Cell-level sources and qualifications accompany the manuscript [17, 36, 45, 67–69, 88, 114].

performance against interaction and compute budgets. Split adaptation, model-selection and final evaluation trajectories before augmentation; final evaluation data must influence neither adaptation nor model selection. Log model/simulator versions, changed components, inference location and numerical format.

Within a route, isolate model changes with sensors, action interfaces, executor and supervision fixed; test controller changes separately. Compare adaptation methods from the same pretrained checkpoint; isolate pretraining with a same-architecture random-initialization control and matched downstream data and optimization budgets where feasible. Across routes, compare foundation-model, specialist and conventional systems on the same robot, tasks, available observations, physical conditions, intervention criteria and resource budgets, using paired initial conditions. Retain each route's necessary interfaces and account for its controller, latency and recovery costs.

Score approach, contact, retention and stabilization separately, with task-calibrated limits on relative-state error, contact peaks, retention duration and residual motion. Report uninterrupted completion separately from recovery, including fault onset, deadlines, rejected actions, retreats, operator corrections and resets. Measure base disturbance, constraint violations, latency distributions and energy over declared components, including idle periods and failed attempts. Separate independent training/adaptation runs from evaluation episodes; retain task/scenario groups and use interval estimates suited to the sampling design [120]. Successive actions are not independent trials. Retain unrecovered trials.

9 Synthesis and research priorities

Table 6 separates reused components, target-domain updates, selected comparison conditions, and the interventions whose benefits were measured.

Table 6: Within-study comparisons of reuse and measured contributions. Conditions refer to selected experiments. NR: not reported.

Study	Reuse and target update	Comparison conditions	Finding and attribution
Cosmos Policy [42]	Video-model initialization; robot policy training	LIBERO; pretrained versus random initialization; same gradient steps	98.5 versus 94.6% success. Video-pretraining benefit at the tested training budget
SpaceMind [67]	Frozen VLM; skill instructions updated over five rounds per group	Six task-bound groups; no cross-task skill transfer	Four groups improve; instruction-based recovery within repeated tasks
OrbitTAMP [17]	Frozen LLM; prompt developed on 100 commands	Held-out instance/radio-style sets, 100 commands each; same ontology	Exact partial MissionConfig: rules 92/57%; Qwen3.5-9B 78/70%; Terra 98/98%. Parser contribution
Spike-driven VLA [90]	Pretrained vision/language encoders; language frozen; 80k policy updates	LIBERO tasks used in training; demonstration count NR	ANN/SNN: 96.3/92.4% success; estimated 179.7/20.9 mJ/chunk. Priors and capacity differ
OpenVLA-OFT [87]	OpenVLA checkpoint; LoRA on filtered trajectories from 500-demo suite pools	LIBERO execution after adaptation; fixed-query A100 timing	Adding chunking: 15.9 → 108.8 actions/s, 62.9 → 73.5 ms/query. Decoding benefit

9.1 What the current results support

Terrestrial comparisons support specific pretraining benefits: video initialization improves Cosmos Policy at matched gradient steps [42], while OpenVLA's robot-trajectory pretraining improves over directly adapting Prismatic, with vision–language pretraining retained in both [33]. These benefits need testing under space dynamics and scarce interaction data.

Other studies isolate interface or adaptation effects: OrbitTAMP compares instruction parsers with planning fixed [17], SpaceMind revises task-specific recovery instructions [67], and dynamics updates address changed action consequences [64]. OpenVLA-OFT increases throughput and query latency; spiking-encoder comparisons change capacity and priors alongside arithmetic cost [87, 90]. These interventions require different explanations of the measured gain.

In the capture example (Figure 5), temporal sensing must deliver usable state in time, and computing savings must survive task execution. Astrobee and LeLaR provide specialist flight-control experience; Loihi 2 couples a physical processor to a simulated robot [36, 84, 114, 119]. Joint tests of task reuse, coupled dynamics, sustained computation and recovery remain the main gap for foundation-model systems.

9.2 Research questions that follow from these comparisons

Where should physical adaptation occur? Use the crossed capture tests (Section 8) to separate held-out generalization from target-physics adaptation. Compare state conditioning, policy updates and controller changes at matched budgets as inertia or contact varies.

Does transfer survive contact and recovery? Test held-out task–physics combinations through contact, retention and stabilization, retaining failed attempts to identify whether recovery depends on the policy, controller or operator.

Do efficiency gains survive deployment? Compare sensing and computing variants on matched tasks, including preprocessing, memory, idle periods and recovery. Diagnose transfer discrepancies before choosing an update and test it in the closed loop. Report feedback timing, task energy and sustained thermal performance alongside accuracy.

9.3 Conclusion

Foundation models contribute reusable representations, task interfaces and learned behaviors to autonomous robotics, with controlled ground studies supporting specific pretraining benefits. The central space challenge is preserving those benefits through changed dynamics, contact and limited onboard resources. Event sensing and spiking computation offer complementary design options; sim-to-ground-to-space validation provides a

path to test them. Separating generalization from bounded target-domain adaptation would connect model reuse to complete-task performance, recovery and resource needs.

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] Evangelos Papadopoulos, Farhad Aghili, Ou Ma, and Roberto Lampariello. Robotic Manipulation and Capture in Space: A Survey. *Frontiers in Robotics and AI*, 8:686723, 2021. doi: 10.3389/frobt.2021.686723.

[2] Issa A. D. Nesnas, Lorraine M. Fesq, and Richard A. Volpe. Autonomy for Space Robots: Past, Present, and Future. *Current Robotics Reports*, 2:251–263, 2021. doi: 10.1007/s43154-021-00057-2.

[3] Roya Firoozi, Johnathan Tucker, Stephen Tian, et al. Foundation Models in Robotics: Applications, Challenges, and the Future. arXiv:2312.07843v1, 2023.

[4] Zhiyuan Xu, Kun Wu, Junjie Wen, Jinming Li, Ning Liu, Zhengping Che, and Jian Tang. A Survey on Robotics with Foundation Models: toward Embodied AI. arXiv:2402.02385v1, 2024.

[5] Yafei Hu, Quanting Xie, Vidhi Jain, et al. Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis. arXiv:2312.08782v3, 2023. Revised 2024-10-01.

[6] Zeyuan Huang, Gang Chen, Zixuan Hao, et al. Artificial Intelligence-Enabled Space Robot Operations: Technologies, Challenges and Prospects. arXiv:2609.16880v1, 2026.

[7] Tao Luo, Yahang Zhang, and Yaobing Wang. Research Status and Application Prospects of Multimodal Large Models for Space Robotics. *Spacecraft Engineering*, 35(1):128–136, 2026. doi: 10.3969/j.issn.1673-8748.2026.01.017. In Chinese; title translated.

[8] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, et al. Event-Based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(1):154–180, 2022. doi: 10.1109/TPAMI.2020.3008413.

[9] Jianing Li, Dianze Li, Arren Glover, Xiaopeng Fan, Guoqi Li, Chiara Bartolozzi, Ryad B. Benosman, and Yonghong Tian. Neuromorphic Object Detection: An In-Depth Study and Future Directions. *Proceedings of the IEEE*, 114(5): 712–739, 2026. doi: 10.1109/JPROC.2026.3717677.

[10] Zhengyu Ma and Yonghong Tian. Theories and Methods for Large-Scale Brain-Inspired Neural Networks. *Chinese Science Bulletin*, 68(35):4764–4781, 2023. doi: 10.1360/TB-2023-0775. In Chinese; English title supplied by the authors.

[11] Kento Kawaharazuka, Tatsuya Matsushima, Andrew Gambardella, Jiaxian Guo, Chris Paxton, and Andy Zeng. Real-World Robot Applications of Foundation Models: A Review. arXiv:2402.05741v1, 2024.

[12] Chiara Bartolozzi, Giacomo Indiveri, and Elisa Donati. Embodied neuromorphic intelligence. *Nature Communications*, 13:1024, 2022. doi: 10.1038/s41467-022-28487-2.

[13] Jianing Li and Yonghong Tian. Recent Advances in Neuromorphic Vision Sensors: A Survey. *Chinese Journal of Computers*, 44(6):1258–1286, 2021. doi: 10.11897/SP.J.1016.2021.01258. In Chinese; English title supplied by the authors.

[14] NASA Small Spacecraft Systems Virtual Institute. 7.0 Thermal Control. State-of-the-Art of Small Spacecraft Technology, 2026 web edition, 2026. URL https://www.nasa.gov/smallsat-institute/sst-soa/thermal-control/. Accessed 2026-10-15.

[15] NASA Small Spacecraft Systems Virtual Institute. 8.0 Small Spacecraft Avionics. State-of-the-Art of Small Spacecraft Technology, 2026 web edition, 2026. URL https://www.nasa.gov/smallsat-institute/sst-soa/small-spacecraft-avionics/. Accessed 2026-10-15.

[16] Kazuya Yoshida. Engineering Test Satellite VII Flight Experiments for Space Robot Dynamics and Control: Theories on Laboratory Test Beds Ten Years Ago, Now in Orbit. *The International Journal of Robotics Research*, 22(5):321–335, 2003. doi: 10.1177/0278364903022005003.

[17] Yuji Takubo, Daniele Gammelli, Marco Pavone, and Simone D'Amico. OrbitTAMP: Grounding Language Models for Task and Motion Planning in Spacecraft Rendezvous. arXiv:2610.01093v1, 2026.

[18] Tae Ha Park and Simone D'Amico. Adaptive Neural Network-based Unscented Kalman Filter for Robust Pose Tracking of Noncooperative Spacecraft. *Journal of Guidance, Control, and Dynamics*, 46(9):1671–1688, 2023. doi: 10.2514/1.G007387.

[19] Wenbo Zhao, Shengjie Wang, Yixuan Fan, Yang Gao, and Tao Zhang. SpaceOctopus: An Octopus-inspired Motion Planning Framework for Multi-arm Space Robot. arXiv:2403.08219v1, 2024.

[20] Yuxue Cao, Shengjie Wang, Xiang Zheng, Wenke Ma, Xinru Xie, and Lei Liu. Reinforcement Learning with Prior Policy Guidance for Motion Planning of Dual-Arm Free-Floating Space Robot. arXiv:2209.01434v1, 2022.

[21] Parna Niksirat, Adriana Daca, and Krzysztof Skonieczny. The effects of reduced-gravity on planetary rover mobility. *The International Journal of Robotics Research*, 39(7):797–811, 2020. doi: 10.1177/0278364920913945.

[22] Tae Ha Park, Marcus Märtens, Gérard Lecuyer, Dario Izzo, and Simone D'Amico. SPEED+: Next-Generation Dataset for Spacecraft Pose Estimation across Domain Gap. In *IEEE Aerospace Conference*, 2022. doi: 10.1109/AERO53065.2022.9843439.

[23] S. Burmeister, S. Elgner, F. Preusker, A. Stark, and J. Oberst. Thermal Effects on Camera Focal Length in MESSENGER Star Calibration and Orbital Imaging. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-3:103–105, 2018. doi: 10.5194/isprs-archives-XLII-3-103-2018.

[24] Aubin Antonsanti, Cédric Virmontois, Jean-Marie Lauenstein, Alexandre Le Roch, Hugo Dewitte, and Vincent Goiffon. Probing Dark Current Random Telegraph Signal in a Small Pitch Vertically Pinned Photodiode CMOS Image Sensor After Proton Irradiation. *IEEE Transactions on Nuclear Science*, 69(7):1506–1514, 2022. doi: 10.1109/TNS.2022.3160056.
[25] Jiawen Yu, Hairuo Liu, Qiaojun Yu, et al. ForceVLA: Enhancing VLA Models with a Force-aware MoE for Contact-rich Manipulation. arXiv:2505.22159v3, 2025.
[26] Ruoxuan Feng, Jiangyu Hu, Wenke Xia, Tianci Gao, Ao Shen, Yuhao Sun, Bin Fang, and Di Hu. AnyTouch: Learning Unified Static-Dynamic Representation across Multiple Visuo-tactile Sensors. arXiv:2502.12191v3, 2025.
[27] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling Proprioceptive-Visual Learning with Heterogeneous Pre-trained Transformers. arXiv:2409.20537v1, 2024.
[28] Saad Memon, Rafal Graczyk, Jan Swakoń, Leszek Grzanka, Sebastian Kusyk, and Mike Papadakis. Proton Irradiation Characterization of an Open-Source ML Accelerator on a Zynq UltraScale+ MPSoC. arXiv:2609.05249v1, 2026.
[29] Wim Nijsink, Bruno Endres Forlin, Amirreza Yousefzadeh, and Marco Ottavi. Shooting Neutrons at Neurons: Radiation Testing of a Spiking Neural Network on Flash-Based FPGAs. arXiv:2605.00030v1, 2026.
[30] Peter Schmaus, Daniel Leidner, Ralph Bayer, Benedikt Pleintinger, Thomas Krüger, and Neal Y. Lii. Continued Advances in Supervised Autonomy User Interface Design for METERON SUPVIS Justin. In *2019 IEEE Aerospace Conference*, 2019. doi: 10.1109/AERO.2019.8741885.
[31] Open X-Embodiment Collaboration. Open X-Embodiment: Robotic Learning Datasets and RT-X Models. arXiv:2310.08864v9, 2023. Revised 2025-05-14.
[32] Wenlong Huang, Fei Xia, Ted Xiao, et al. Inner Monologue: Embodied Reasoning through Planning with Language Models. arXiv:2207.05608v1, 2022.
[33] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, et al. OpenVLA: An Open-Source Vision-Language-Action Model. arXiv:2406.09246v3, 2024.
[34] Michael Ahn, Anthony Brohan, Noah Brown, et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances. arXiv:2204.01691v2, 2022.
[35] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, et al. PaLM-E: An Embodied Multimodal Language Model. arXiv:2303.03378v1, 2023.
[36] Somrita Banerjee, Abhishek Cauligi, and Marco Pavone. Deep Learning Warm Starts for Trajectory Optimization on the International Space Station. arXiv:2505.05588v3, 2025.
[37] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control. arXiv:2209.07753v4, 2022. Revised 2023-05-25.
[38] Yaodong Yang, Hongyao Tang, Yi Ma, Xingyu Fan, Weixun Wang, Jinpeng Li, and Tianpei Yang. Code Owns the Simulation, Jev Owns the Evaluation. arXiv:2610.01834v2, 2026. Revised 2026-10-10.
[39] Octo Model Team, Dibya Ghosh, Homer Walke, et al. Octo: An Open-Source Generalist Robot Policy. arXiv:2405.12213v2, 2024.
[40] Kevin Black, Noah Brown, Danny Driess, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. arXiv:2410.24164v4, 2024. Revised 2026-01-08.
[41] Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning Interactive Real-World Simulators. arXiv:2310.06114v3, 2023. Revised 2024-09-26.
[42] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, et al. Cosmos Policy: Fine-Tuning Video Models for Visuomotor Control and Planning. arXiv:2601.16163v1, 2026.
[43] Matthew Abate, Mark Mote, Mehregan Dor, et al. Run Time Assurance for Spacecraft Attitude Control Under Nondeterministic Assumptions. *IEEE Transactions on Control Systems Technology*, 2024. doi: 10.1109/TCST.2023.3340624.
[44] Philipp G. Knestel, Luisa Mayershofer, Anne Köpken, et al. NealAI: A RAG-based Chat Assistant Supporting Command of a Robotic Team from the ISS. In *International Conference on Space Robotics (iSpaRo)*, pages 684–691, 2025. doi: 10.1109/iSpaRo66239.2025.11436289.
[45] Matthew Foutter, Daniele Gammelli, Justin Kruger, Ethan Foss, Praneet Bhoj, Tommaso Guffanti, Simone D'Amico, and Marco Pavone. Space-LLaVA: a Vision-Language Model Adapted to Extraterrestrial Applications. arXiv:2408.05924v2, 2024. Revised 2025-01-18.
[46] Cristina Pérez-Ramos, Leopoldo Altamirano-Robles, and Miguel Chávez-Dagostino. Zero-shot semantic landmark-based visual odometry using foundation models for unstructured planetary exploration. *Frontiers in Robotics and AI*, 13: 1819714, 2026. doi: 10.3389/frobt.2026.1819714.
[47] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. FoundationPose: Unified 6D Pose Estimation and Tracking of Novel Objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17868–17879, 2024. URL https://arxiv.org/abs/2312.08344v2. Reviewed arXiv:2312.08344v2, revised 2024-03-26.
[48] Arunkumar Rathinam, Haytam Qadadri, and Djamila Aouada. SPADES: A Realistic Spacecraft Pose Estimation Dataset using Event Sensing. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024. doi: 10.1109/ICRA57147.2024.10611231.
[49] Arunkumar Rathinam, Jules Lecomte, Jost Reelsen, Gregor Lenz, Axel von Arnim, and Djamila Aouada. Efficient Onboard Spacecraft Pose Estimation with Event Cameras and Neuromorphic Hardware. arXiv:2604.04117v1, 2026.
[50] Seth Roffe, Himanshu Akolkar, Alan D. George, Bernabé Linares-Barranco, and Ryad Benosman. Neutron-Induced, Single-Event Effects on Neuromorphic Event-Based Vision Sensor: A First Step and Tools to Space Applications. *IEEE Access*, 9:85748–85763, 2021. doi: 10.1109/ACCESS.2021.3085136. Author manuscript: arXiv:2102.00112v1.
[51] David Rodríguez-Martínez and C. J. Pérez del Pulgar. Fast Vision in the Dark: A Case for Single-Photon Imaging in Planetary Navigation. In *18th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2025. URL https://arxiv.org/abs/2510.10597v1. Reviewed arXiv:2510.10597v1.
[52] Yixuan Wang, Jianing Li, Lin Zhu, Xijie Xiang, Tiejun Huang, and Yonghong Tian. Learning Stereo Depth Estimation with Bio-Inspired Spike Cameras. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2022. doi: 10.1109/ICME52920.2022.9859975.
[53] David Rodríguez-Martínez, Dave van der Meer, Junlin Song, Abhishek Bera, C. J. Pérez del Pulgar, and Miguel Angel Olivares-Mendez. SPICE-HL3: Single-Photon, Inertial, and Stereo Camera dataset for Exploration of High-Latitude Lunar Landscapes. *Scientific Data*, 13:374, 2026. doi: 10.1038/s41597-026-06668-8.
[54] Jiazhou Zhou, Xu Zheng, Yuanhuiyi Lyu, and Lin Wang. EventBind: Learning a Unified Representation to Bind Them All for Event-based Open-world Understanding. In *European Conference on Computer Vision (ECCV)*, 2024. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/html/8931_ECCV_2024_paper.php.
[55] Shaoyu Liu, Jianing Li, Guanghui Zhao, Yunjian Zhang, Xin Meng, Fei Richard Yu, Xiangyang Ji, and Ming Li. EventGPT: Event Stream Understanding with Multimodal Large Language Models. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 29139–29149, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Liu_EventGPT_Event_Stream_Understanding_with_Multimodal_Large_Language_Models_CVPR_2025_paper.html.

[56] Jiajun Zhai, Hao Shi, Shangwei Guo, Kailun Yang, and Kaiwei Wang. E-VLA: Event-Augmented Vision-Language-Action Model for Dark and Blurred Scenes. arXiv:2604.04834v2, 2026.

[57] Jiaxin Liu, Xun Xu, Zhenhao Zhang, Hanqing Wang, Ruiqi Chen, Shi Chang, Weiyu Guo, and Laurent Kneip. Event-VLA: Action-Conditioned Event Fusion for Robust Vision-Language-Action Model. arXiv:2606.29384v1, 2026.

[58] Jonathan Courtois, Benoît Miramond, and Alain Pegatoquet. Spiking monocular event based 6D pose estimation for space application. In *Conference on AI in and for Space (SPAICE)*, pages 394–399, 2024. doi: 10.5281/zenodo.13885633.

[59] Joshua Jones, Oier Mees, Carmelo Sferrazza, Kyle Stachowicz, Pieter Abbeel, and Sergey Levine. Beyond Sight: Finetuning Generalist Robot Policies with Heterogeneous Sensors via Language Grounding. arXiv:2501.04693v3, 2025.

[60] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. arXiv:2301.04104v2, 2023. Revised 2024-04-17.

[61] Joel Jang, Seonghyeon Ye, Zongyu Lin, et al. DreamGen: Unlocking Generalization in Robot Learning through Video World Models. arXiv:2505.12705v2, 2025.

[62] Mido Assran, Adrien Bardes, David Fan, et al. V-JEPA 2: Self-Supervised Video Models Enable Understanding, Prediction and Planning. arXiv:2506.09985v1, 2025.

[63] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, et al. GR-2: A Generative Video-Language-Action Model with Web-Scale Knowledge for Robot Manipulation. arXiv:2410.06158v1, 2024.

[64] Andrej Orsula, Miguel Olivares-Mendez, and Carol Martinez. Reward-Free Continual Adaptation for Resilient Space Robots. arXiv:2608.23452v1, 2026.

[65] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models. arXiv:2307.05973v2, 2023. Revised 2023-11-02.

[66] HongRun Chen, Li Yang, Long Sun, and HaoQing Yu. LLM-KGFP: A chain-of-thought-driven knowledge graph planner for autonomous rover operations in lunar polar shadowed regions. *Advances in Space Research*, 77(6):7406–7423, 2026. doi: 10.1016/j.asr.2026.01.031.

[67] Aodi Wu, Haodong Han, Xubo Luo, Ruisuo Wang, Shan He, and Xue Wan. SpaceMind: A Modular and Self-Evolving Embodied Vision-Language Agent Framework for Autonomous On-orbit Servicing. arXiv:2604.14399v1, 2026.

[68] Bingqi Huang, Bingchuan Wei, Yingkai Cai, and Zhaokui Wang. Toward Self-Evolving Lunar Robotic Autonomy Through Contract-Governed Skill Registration. *Astronautics*, 1(3):15, 2026. doi: 10.3390/astronautics1030015.

[69] NASA Jet Propulsion Laboratory. NASA's Perseverance Rover Completes First AI-Planned Drive on Mars. NASA/JPL mission report, 2026. URL https://www.nasa.gov/missions/mars-2020-perseverance/perseverance-rover/nasas-perseverance-rover-completes-first-ai-planned-drive-on-mars/. Accessed 2026-10-15.

[70] R. Francis, T. Estlin, G. Doran, et al. AEGIS autonomous targeting for ChemCam on Mars Science Laboratory: Deployment and results of initial science team use. *Science Robotics*, 2(7):eaan4582, 2017. doi: 10.1126/scirobotics.aan4582.

[71] Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, et al. Robots That Ask For Help: Uncertainty Alignment for Large Language Model Planners. arXiv:2307.01928v2, 2023.

[72] Alejandro Carrasco, Victor Rodriguez-Fernandez, and Richard Linares. Fine-tuning LLMs for Autonomous Spacecraft Control: A Case Study Using Kerbal Space Program. arXiv:2408.08676v1, 2024.

[73] Sida He, Lingxi Xie, Yunning Cao, et al. Robot Manipulation with GPT-6-Astra: Body Knowledge, Experience Reuse, Emergent Skills, and Sim2Real Transfer. arXiv:2609.31770v1, 2026.

[74] Galbot Team, Xuchuan Chen, Xiaoqian Cheng, Yu Deng, et al. Systematically Exploring the Capabilities of GPT-6 Astra as Embodied Policies. arXiv:2609.38537v1, 2026.

[75] Anthony Brohan, Noah Brown, Justice Carbajal, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. arXiv:2307.15818v1, 2023.

[76] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. arXiv:2303.04137v1, 2023.

[77] Karl Pertsch, Kyle Stachowicz, Brian Ichter, et al. FAST: Efficient Action Tokenization for Vision-Language-Action Models. arXiv:2501.09747v1, 2025.

[78] Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-Time Execution of Action Chunking Flow Policies. arXiv:2506.07339v2, 2025. Revised 2025-12-05.

[79] Jingya Wang, Dehao Zhang, Shuai Wang, Malu Zhang, Yang Yang, and Haizhou Li. SpikingVLA: Asynchronous Spiking Vision-Language-Action Models. arXiv:2610.09710v1, 2026.

[80] Chengbo Yuan, Zicheng Zhang, Mingjie Zhou, et al. FTP-1: A Generalist Foundation Tactile Policy Across Tactile Sensors for Contact-Rich Manipulation. arXiv:2606.13102v3, 2026.

[81] Han Xue, Jieji Ren, Wendi Chen, Gu Zhang, Yuan Fang, Guoying Gu, Huazhe Xu, and Cewu Lu. Reactive Diffusion Policy: Slow-Fast Visual-Tactile Policy Learning for Contact-Rich Manipulation. arXiv:2503.02881v2, 2025. Revised 2025-04-01.

[82] Noriyasu Inaba, Mitsushige Oda, and Masato Hayashi. Visual Servoing of Space Robot for Autonomous Satellite Capture. *Transactions of the Japan Society for Aeronautical and Space Sciences*, 46(153):173–179, 2003. doi: 10.2322/tjsass.46.173.

[83] Samet Hicsonmez, Jose Sosa, Dan Pineau, Inder Pal Singh, Arunkumar Rathinam, Abd El Rahman Shabayek, and Djamila Aouada. Annotation Free Spacecraft Detection and Segmentation using Vision Language Models. arXiv:2602.04699v1, 2026.

[84] Kenneth Stewart, Samantha Chapin, Roxana Leontie, and Carl Glen Henshaw. Crossing the Sim2Real Gap Between Simulation and Ground Testing to Space Deployment of Autonomous Free-flyer Control. arXiv:2512.03736v1, 2025.

[85] Amir Taherin, Juyi Lin, Arash Akbari, Arman Akbari, Pu Zhao, Weiwei Chen, David Kaeli, and Yanzhi Wang. Cross-platform scaling of vision-language-action models from edge to cloud GPUs, 2026. URL https://arxiv.org/abs/2509.11480v2. Reviewed version 2, revised 26 January 2026; first submitted 2025.

[86] Hongyu Wang, Chuyan Xiong, Ruiping Wang, and Xilin Chen. BitVLA: 1-bit vision-language-action models for robotics manipulation, 2026. URL https://arxiv.org/abs/2506.07530v2. Reviewed version 2, revised 1 March 2026; first submitted 2025.

[87] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. arXiv:2502.19645v2, 2025. Revised 2025-04-28.

[88] Liwen Zhang, Dong Zhou, Shibo Shao, Zihao Su, and Guanghui Sun. Multimodal Spiking Neural Network for Space Robotic Manipulation. arXiv:2508.07287v1, 2025.

[89] Ruiqi Song, Dujun Nie, Siyu Teng, et al. SpikeVLA: Vision-Language-Action Models with Spiking Neural Networks. arXiv:2606.27807v1, 2026.

[90] Shuai Wang, Malu Zhang, Mingquan Liu, et al. Spike-driven Vision-Language-Action Model. arXiv:2609.39514v1, 2026.

[91] Jaden Clark, Changhao Wang, Yihuai Gao, Seongheon Hong, Hojung Choi, Mark Cutkosky, Yifan Hou, and Shuran Song. Multisensory Continual Learning: Adapting Pretrained Visuomotor Policies to Force. arXiv:2606.30988v4, 2026.

[92] Jason Yik, Korneel Van den Berghe, Douwe den Blanken, et al. The neurobench framework for benchmarking neuromorphic computing algorithms and systems. *Nature Communications*, 16:1545, 2025. doi: 10.1038/s41467-025-56739-4.

[93] Federico Paredes-Vallés, Jesse Hagenaars, Julien Dupeyroux, Stein Stroobants, Yingfu Xu, and Guido de Croon. Fully neuromorphic vision and control for autonomous drone flight. *Science Robotics*, 9:eadi0591, 2024. doi: 10.1126/scirobotics.adi0591. URL https://arxiv.org/abs/2303.08778v1. Energy benchmark inspected in preprint arXiv:2303.08778v1, Table 2.

[94] Meiqi Wang, Han Qiu, Longnv Xu, Di Wang, Yuanjie Li, Tianwei Zhang, Jun Liu, and Hewu Li. A Case for Application-Aware Space Radiation Tolerance in Orbital Computing. arXiv:2407.11853v1, 2024.

[95] Xinghua Xue, Cheng Liu, Ying Wang, Bing Yang, Tao Luo, Lei Zhang, Huawei Li, and Xiaowei Li. Soft Error Reliability Analysis of Vision Transformers. arXiv:2302.10468v3, 2023. Revised 2024-04-26.

[96] Lester Frias-Dominguez, Jose M. Badia, German Leon, Adrian Amor-Martin, and Jose A. Belloch. Dependability analysis and hardening of vision transformers against soft errors. *The Journal of Supercomputing*, 82:331, 2026. doi: 10.1007/s11227-026-08373-0.

[97] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. arXiv:2112.03227v4, 2021. Revised 2022-07-13.

[98] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. arXiv:2306.03310v2, 2023.

[99] Xuanlin Li, Kyle Hsu, Jiayuan Gu, et al. Evaluating Real-World Robot Manipulation Policies in Simulation. arXiv:2405.05941v1, 2024.

[100] Aodi Wu, Jianhong Zuo, Zeyuan Zhao, Xubo Luo, Ruisuo Wang, and Xue Wan. SpaceSense-Bench: A Large-Scale Multi-Modal Benchmark for Spacecraft Perception and Pose Estimation. arXiv:2603.09320v1, 2026.

[101] Andrej Orsula, Matthieu Geist, Miguel Olivares-Mendez, and Carol Martinez. Space Robotics Bench: Robot Learning Beyond Earth. arXiv:2509.23328v1, 2025.

[102] Patrick W. Kenneally, Scott Piggott, and Hanspeter Schaub. Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework. *Journal of Aerospace Information Systems*, 17(9):496–507, 2020. doi: 10.2514/1.I010762.

[103] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. ESIM: An Open Event Camera Simulator. In *Proceedings of the 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 969–982, 2018. URL https://proceedings.mlr.press/v87/rebecq18a.html.

[104] Yuhuang Hu, Shih-Chii Liu, and Tobi Delbruck. v2e: From Video Frames to Realistic DVS Events. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2021. URL https://arxiv.org/abs/2006.07722v2. Reviewed preprint arXiv:2006.07722v2; first posted 2020.

[105] Ondřej Dvořák, Marcus Märtens, and Dario Izzo. Event-based Lunar OPtical flow Egomotion estimation (ELOPE) dataset. Zenodo, version 1, 2025. URL https://zenodo.org/records/15421707.

[106] Olaf Sikorski, Dario Izzo, and Gabriele Meoni. Event-Based Spacecraft Landing Using Time-To-Contact. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1941–1950, 2021. URL https://openaccess.thecvf.com/content/CVPR2021W/AI4Space/html/Sikorski_Event-Based_Spacecraft_Landing_Using_Time-To-Contact_CVPRW_2021_paper.html.

[107] Xin-Yu Xiao, Ye Tian, Erwei Yin, Zhixian He, Shiqi Wang, Yalei Liu, and Qianchen Xia. Lunar-Bench: Towards Evaluating Task-Oriented Reasoning of LLMs in Lunar Exploration Scenarios. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 1668–1705, 2026. doi: 10.18653/v1/2026.findings-acl.83.

[108] Haiquan Li, Qingqing Wei, Jianxun Liang, Weiyan Ren, Zixin Tang, and Delun Li. Estimation of Vibration Characteristics of a Space Manipulator From Air Bearing Supported Test Data. *Frontiers in Robotics and AI*, 8:641165, 2021. doi: 10.3389/frobt.2021.641165.

[109] Ferdinand Elhardt, Roland Boumann, Marco De Stefano, et al. The Motion Suspension System – MSS: A Cable-Driven System for On-Ground Tests of Space Robots. In *Advances in Mechanism and Machine Science: Proceedings of the 16th IFToMM World Congress*, volume 148 of *Mechanisms and Machine Science*, pages 379–388. Springer, 2023. doi: 10.1007/978-3-031-45770-8_38.

[110] Ferdinand Elhardt, Marco de Stefano, Manfred Schedl, et al. The Motion Suspension System for On-Ground Tests of Space Robots: Demonstration and Results. In *IEEE Aerospace Conference*, 2024. doi: 10.1109/AERO58975.2024.10521208.

[111] DLR. European Proximity Operations Simulator (EPOS 2.0). DLR research infrastructure, n.d. URL https://www.dlr.de/en/research-and-transfer/research-infrastructure/european-proximity-operations-simulator-epos. Accessed 2026-10-15.

[112] Jordi Artigas, Marco De Stefano, Wolfgang Rackl, et al. The OOS-SIM: An On-ground Simulation Facility For On-Orbit Servicing Robotic Operations. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2854–2860, 2015. URL https://elib.dlr.de/97179/1/Artigas_OOSSIM_icra15.pdf.

[113] Melak Zebenay, Toralf Boge, Rainer Krenn, and Daniel Choukroun. Analytical and experimental stability investigation of a hardware-in-the-loop satellite docking simulator. arXiv:1309.3512v1, 2013.

[114] Kenneth Stewart, Roxana Leontie, Samantha Chapin, Joe Hays, Sumit Bam Shrestha, and Carl Glen Henshaw. Autonomous Reinforcement Learning Robot Control with Intel's Loihi 2 Neuromorphic Hardware. arXiv:2512.03911v1, 2025.

[115] Ronald Scrofano, Scott Davis, and Jennifer Taggart. Radiation Test Performance of the Intel Loihi Neuromorphic Processor. In *IEEE Space Computing Conference (SCC)*, 2024. doi: 10.1109/SCC61854.2024.00016.

[116] NASA. Standard for Models and Simulations. Technical Report NASA-STD-7009B, NASA, 2024. URL https://standards.nasa.gov/sites/default/files/standards/NASA/B/1/NASA-STD-7009B-Final-3-5-2024.pdf. Approved 2024-03-05.

[117] Andrej Orsula, Simon Bøgh, Miguel Olivares-Mendez, and Carol Martinez. Learning to Grasp on the Moon from 3D Octree Observations with Deep Reinforcement Learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4112–4119, 2022. doi: 10.1109/IROS47612.2022.9981661.

[118] Matthew G. McHarg, Richard L. Balthazor, Brian J. McReynolds, et al. Falcon Neuro: an event-based sensor on the International Space Station. *Optical Engineering*, 61(8):085105, 2022. doi: 10.1117/1.OE.61.8.085105.

[119] Kirill Djebko, Tom Baumann, Erik Dilger, Frank Puppe, and Sergio Montenegro. LeLaR: The First In-Orbit Demonstration of an AI-Based Satellite Attitude Controller. *IEEE Access*, 14:49348–49367, 2026. doi: 10.1109/ACCESS.2026.3678816. Reviewed author version: arXiv:2512.19576v5.

[120] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C. Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320, 2021. URL https://papers.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html. Reviewed arXiv:2108.13264v4, revised 2022-01-05.