GPU-Accelerated Astrodynamics World Models for Spacecraft Rendezvous and Proximity Operations

Duncan Eddy
Stanford University
Isaac R. Ward, Grace Ra Kim, and Mykel J. Kochenderfer
Stanford University

ABSTRACT

World models are an emerging paradigm in representation learning in which an agent jointly learns state-action dynamics and observation models from offline trajectory data, enabling multi-step planning and trajectory prediction with uncertainty estimates. They have demonstrated strong results in robotics and game environments, but, to the best of our knowledge, they have not previously been applied to the space domain. This paper introduces a world model-based approach to cooperative and non-cooperative spacecraft rendezvous and proximity operations, and makes three contributions. First, we introduce an open-source, JAX-based International Space Station (ISS) docking simulation environment that supports parallel GPU-based simulation of spacecraft orbit and attitude dynamics, generating the thousands of state-action transitions that world model training requires. Second, we introduce *Out-of-this-World-Model*, a transformer-based world model that encodes relative kinematic states and body-fixed camera imagery into a latent state and predicts its evolution under commanded thrusts and control torques using one-step flow matching. The model produces a distribution over future observations, capturing stochastic dynamics and providing per-timestep uncertainty estimates, and achieves greater predictive performance than DreamerV3-style posterior-correction baselines with fewer trainable parameters and hyperparameters. Third, we apply the approach to a capsule autonomously docking with the ISS under keep-out-zone constraints, demonstrating improved sample efficiency and task performance over reinforcement learning baselines (53% versus 29% docking success across ports), substantially better out-of-distribution generalization (on held-out ports the world model more than doubles baseline success, 40% versus 17%), and detection of anomalous objects encountered during the docking approach with 98% classification accuracy. We open-source the simulation environment and model architecture to enable further study of this paradigm.

1. INTRODUCTION

Spacecraft rendezvous and proximity operations (RPO) are transitioning from a rare, human-supervised procedure to a routine element of civil, military, and commercial space operations. Orbital rendezvous was first demonstrated in 1965 when Gemini 6A closed and held close proximity with Gemini 7. Rendezvous and docking remain essential to human space exploration today, with every crew and cargo vehicle visiting the International Space Station (ISS) performing one. These capabilities are currently central to future exploration architectures such as the Human Landing System, which requires multiple automated dockings and propellant transfers per mission [1]. The technology is also dual-use. There is also a growing tempo of military RPO activity in both low Earth orbit and the geostationary belt, with inspection and shadowing maneuvers by multiple nations becoming a recurring feature of the space domain [2]. Beyond orbital games, on-orbit servicing is seen as a key enabler for reducing the cost of space operations by extending the life of satellites limited by consumables rather than hardware failures. The Orbital Express program demonstrated autonomous servicing in-orbit [3], and Northrop Grumman's Mission Extension Vehicles have since docked with operational geostationary communications satellites to provide life-extension services [4].

Despite this growing operational tempo, rendezvous remains a complicated and risky procedure. Two spacecraft must be brought from kilometers of separation to physical contact with centimeter-level precision, and any collision risks permanent damage to both vehicles. A collision during approach is not only a mission failure—it is a potential debris-generating event that places other satellites in the orbital population at risk. Safe autonomous rendezvous therefore requires methods that can reason about the consequences of control actions before execution and recognize off-nominal scenarios early enough to initiate an abort before the operation leads to catastrophic failure.

For an autonomous rendezvous agent, this reasoning must span both uncertain dynamics and uncertain sensing. The agent observes the world through a range of potential sensors—GNSS receivers, cameras, star trackers, sun sensors,

LIDAR, and ground-based tracking—and must fuse these measurements into an estimate of the relative state before deciding on a course of action. Operations can be cooperative, where the target vehicle exchanges state information and measurements with the chaser, or non-cooperative, where the chaser relies entirely on its own sensing. Cooperative measurements, typically GNSS observables, reduce relative state uncertainty significantly, but they can never remove the aleatoric noise of the underlying sensors. Cameras can provide measurements of relative position and attitude without relying on cooperative information exchange, making them robust to communications failures and supporting scenarios where cooperation is not possible, but provide less accurate relative state measurements. The central question for autonomy in either case is how to fuse the available, heterogeneous measurements and reason over their meaning when selecting actions.

Historically, this question has been answered by decomposing the problem into separate guidance, navigation, and control (GNC) functions. A filter—often a Kalman filter variant [5, 6]—combines an analytic dynamics model with analytic measurement models to produce a state estimate, then a guidance and control layer plans maneuvers against that estimate. Model predictive control has been applied to the docking problem with explicit constraint handling [7], and artificial potential function methods provide safety guarantees under motion constraints [8]. These decompositions work well when the dynamics and measurement models are accurate, and the observations are low-dimensional. They offer no direct mechanism, however, for ingesting rich sensing modalities such as camera imagery, which must first be preprocessed through a computer vision pipeline to extract relative pose measurements before the filter can consume them.

Learning-based methods promise to close this gap by learning system behavior directly from data, but existing approaches fall short of the full problem. Directly regressing state dynamics or next observations with feedforward networks degrades rapidly over multi-step prediction horizons as errors compound [9]. Physics-informed neural networks encode dynamical constraints, such as conservation of energy or angular momentum, into the training loss, but they are brittle to train and model only the dynamics portion of the system evolution—they provide no mechanism for predicting the sensor observations that the agent actually receives. Martin and Schaub study physics-informed gravity modeling and document these training challenges [10]. Reinforcement learning has been applied to spacecraft guidance more broadly [11, 12], but learns a reactive policy tied to the reward and scenario distribution it was trained on rather than a reusable model of the system.

World models are an emerging paradigm in representation learning that addresses this full problem. First introduced by Ha and Schmidhuber [13], a world model co-learns the joint state-action dynamics and the observation model of a system from trajectory data, producing a learned simulator that predicts future observations conditioned on actions. For readers accustomed to spacecraft GNC, a world model can be understood as a learned system model—a fusion of the Kalman filter's dynamics and measurement models into a single network that ingests raw measurements and actions and predicts the distribution over future measurements directly. Fig. 1 illustrates this correspondence.

World models have demonstrated strong results in robotics and game domains, enabling multi-step reasoning, model-based planning, and quantification of predictive uncertainty [14, 15]. Unlike the Kalman filter, world models currently carry no optimality guarantees, but they inherit none of the filter's structural assumptions either: no linearization, no Gaussian noise model, and no requirement that observations be reduced to low-dimensional residuals before use.

This paper introduces a world model approach to spacecraft rendezvous and proximity operations. Our contributions are threefold:

1. We introduce *AstroJAX*, an open-source astrodynamics framework written in JAX that supports massively parallel, GPU-accelerated simulation of orbit and attitude dynamics, along with an open-source ISS-docking simulation environment built on it for generating world model training data at scale.

2. We introduce *Out-of-this-World-Model* (OWM), a transformer-based world model architecture that fuses kinematic and visual observations into a shared latent space and predicts distributions over future observations with a one-step flow-matching head. To the best of our knowledge, this is the first application of world models to space operations.

3. We demonstrate that the learned world model composes with classical sampling-based planning, specifically model predictive path integral control [16], to dock with the ISS, raising docking success across all eight ports from 29% with reinforcement learning to 53%, an 84% relative gain, with the margin widest at berthing ports never seen in the training data, where it more than doubles the success rate of goal-conditioned baselines (40%

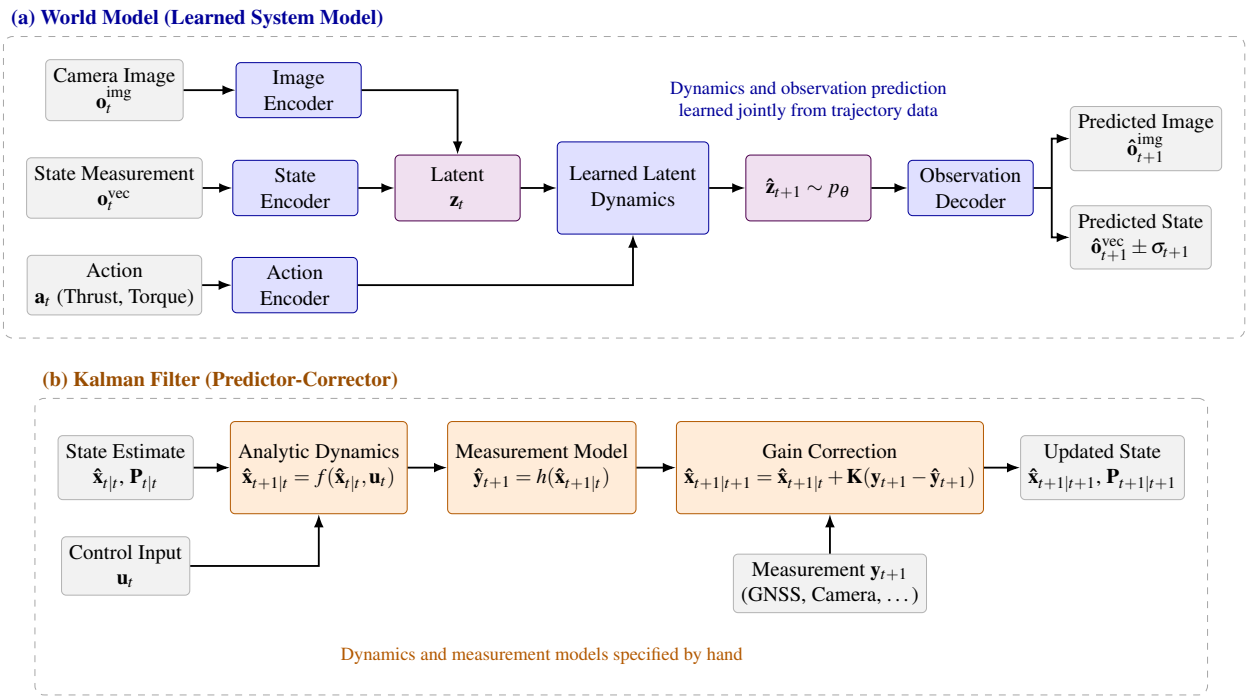


Fig. 1: World models compared with the classical predictor-corrector approach to state estimation and prediction. (a) A world model learns encoders, latent dynamics, and an observation decoder jointly from trajectory data. It ingests raw measurements—camera imagery and noisy kinematic state—together with the applied action, and predicts a distribution over future observations, from which per-timestep uncertainty estimates follow directly. (b) A Kalman filter propagates a state estimate through analytic dynamics and measurement models, correcting the prediction with a gain-weighted measurement residual.

versus 17%) and reaches ports the baselines fail to acquire at all. At the same time the model's predictive uncertainty detects anomalies at runtime, correctly classifying approach sequences with docked vehicles not seen during training from approach sequences without such vehicles 98% of the time.

We open-source the simulation environments, model architecture, and training datasets to enable further study of this paradigm.[1]

The paper proceeds as follows. Section 2 provides background on world models and reviews prior work on learning-based rendezvous and proximity operations. Section 3 presents the world model formulation, the OWM architecture and training procedure, and the GPU-accelerated astrodynamics simulation used to generate training data. Section 4 describes the experimental setup, baseline algorithms, and evaluation criteria. Section 5 presents results on rendezvous performance, generalization to unseen docking ports, and anomaly detection. Section 6 concludes, summarizing the work and identifying areas for future development.

2. BACKGROUND

This section reviews the two bodies of work that this paper brings together: world models from the representation learning literature, and methods for rendezvous and proximity operations from the spacecraft GNC literature.

2.1 World Models

Ha and Schmidhuber introduce the modern world model formulation as a generative recurrent network that learns a compressed spatial and temporal representation of an environment from recorded trajectories. They show that policies

[1]The model architecture and simulation environments are available at https://github.com/sisl/outofthisworldmodel and https://github.com/sisl/outofthisworldmodel-envs. Training datasets are published at https://huggingface.co/sislaboratory.

trained entirely inside the learned model transfer back to the real environment [13]. The Dreamer line of work extends this formulation into a recurrent state-space model that learns a posterior over latent states from observations and a prior that predicts the latent forward in time, correcting the prior toward the posterior during training. The third generation of this architecture masters a wide range of game environments from Atari to Minecraft with a single set of hyperparameters [14]. These posterior-correction architectures are the closest learned analog to the predictor-corrector structure of a Kalman filter, and we use a Dreamer-style model as an architectural point of comparison in this work.

A second family of world models discards observation reconstruction entirely. LeCun proposes the joint-embedding predictive architecture (JEPA), which predicts future representations in latent space rather than future observations in data space [17], and Assran et al. demonstrate the approach at scale for images [18]. Latent-space prediction is cheaper and avoids modeling pixel-level detail irrelevant to control, but it admits a degenerate solution in which the encoder collapses to a constant. The literature offers several remedies. Grill et al. stabilize latent prediction with a slowly updated exponential-moving-average target encoder [19], and Bardes et al. regularize the batch latent distribution directly with variance and covariance penalties [20]. Maes et al. show that a sketched isotropic-Gaussian regularizer yields stable end-to-end joint-embedding world models from pixels [21]. The architecture we introduce in Section 3 supports these collapse-prevention strategies as interchangeable components of a shared transformer backbone.

The final ingredient of our approach comes from generative modeling. Flow matching learns a velocity field that transports samples from a noise distribution to a data distribution along near-straight paths [22, 23], and shortcut models add a self-consistency objective that lets the same network take one large integration step instead of many small ones [24]. Applied to latent prediction, these methods produce a distribution over next latent states that can be sampled in a single network evaluation—fast enough to sit inside a sampling-based planner—while retaining the ability to represent multimodal and stochastic dynamics that a point-prediction architecture cannot.

Learned system models have begun to see application in domains adjacent to spacecraft operations. An early line of work applies the paradigm to high-dimensional fluid dynamical simulation: Morton et al. learn the forced and unforced dynamics of unsteady fluid flows directly from computational fluid dynamics data and suppress vortex shedding by performing model predictive control over the learned model [25], with subsequent work inferring distributions over the learned dynamics for uncertainty-aware control [26] and pairing learned compressions with learned dynamics to recover missing data from high-order flow simulations [27]. Closer to our setting, Ward et al. train a probabilistic world model in the latent space of a pretrained video tokenizer and show that its predictive uncertainty reliably detects failures in real-world robotic bimanual manipulation tasks [15]—evidence that world model uncertainty is a practical runtime monitoring signal, a property we exploit for anomaly detection in orbit. To the best of our knowledge, however, no prior work applies world models to problems in the space domain.

2.2 Rendezvous and Proximity Operations

Reinforcement learning is the most studied learning-based approach to RPO. Broida and Linares apply proximal policy optimization to rendezvous guidance in cluttered orbital environments [28], and Gaudet et al. develop reinforcement meta-learning for angles-only intercept and adaptive guidance problems [29, 30]. Federici et al. compare deep learning techniques for autonomous proximity operations guidance under image-based navigation [31], and Fereoli et al. extend meta-reinforcement learning to proximity operations in cislunar space [32]. Allen et al. release SpaceGym, a suite of non-cooperative space game environments intended to spur reinforcement learning research in the domain [33].

Much of this work is enabled by dedicated simulation infrastructure. Stephenson and Schaub build the BSK-RL library of reinforcement learning environments on top of the Basilisk astrodynamics library [34]. Stephenson et al. use these tools to learn autonomous inspection policies with optimization-based safety guarantees [35], and Herrmann and Schaub apply reinforcement learning to small-body science operations [36]. Huterer Prats et al. apply reinforcement learning to space-to-space surveillance scheduling [37]. These simulation frameworks execute on CPUs. Our work is complementary: by expressing the full dynamics, sensing, and reward pipeline in JAX, we can generate the hundreds of thousands of transitions that world model training requires directly on GPU hardware in parallel.

Docking has also been studied extensively through the lens of classical control. Weiss et al. apply model predictive control to rendezvous and docking with explicit handling of thrust limits, approach-cone, and soft-docking constraints [7]. Dong et al. construct artificial potential functions that guarantee safety under motion constraints [8], and Breeden and Panagou provide docking safety guarantees with control barrier functions [38]. Maestrini and Di Lizia develop guidance strategies for inspecting unknown non-cooperative objects [39]. These methods provide strong guarantees against the models they are given, but they inherit the limits of those models: rich observations such as camera imagery

enter only after being processed by a computer vision pipeline to derive relative-state estimates.

A smaller body of work learns spacecraft dynamics directly. Silvestrini and Lavagna survey neural-network approaches to spacecraft dynamics, navigation, and control, and document the difficulty of maintaining accuracy over long propagation horizons [40]. Martin and Schaub develop physics-informed neural network gravity models and detail the training challenges of the approach [10]. These efforts model the dynamics alone. A world model differs in scope: it jointly learns the dynamics and the observation process, which is what allows it to consume camera imagery and kinematic measurements directly and to predict both forward in time.

3. METHODS

Our approach has two components: a world model that learns the joint evolution of spacecraft state and sensor observations from trajectory data, and a GPU-accelerated simulation stack that generates that data at the scale world model training requires. This section presents the problem formulation, the Out-of-this-World-Model architecture and its training procedure, and the astrodynamics simulation environments.

3.1 Problem Formulation

We model the docking scenario as a discrete-time partially observable decision process [41]. At each timestep t, the environment occupies a state $\mathbf{s}_t$, the agent applies an action $\mathbf{a}_t$, and the state evolves according to a transition function

$$\mathbf{s}_{t+1} = T(\mathbf{s}_t, \mathbf{a}_t) \tag{1}$$

where T integrates the vehicle dynamics over one control interval. The agent does not observe $\mathbf{s}_t$ directly. Instead it receives an observation

$$\mathbf{o}_t = \left(\mathbf{o}_t^{\text{vec}}, \mathbf{o}_t^{\text{img}}\right) \tag{2}$$

composed of a kinematic measurement $\mathbf{o}_t^{\text{vec}}$, the relative state corrupted by sensor noise, and a body-fixed camera image $\mathbf{o}_t^{\text{img}}$. The environment also emits a scalar reward r_t used by the reinforcement learning baselines; the world model itself never consumes it.

The world model's task is to approximate the distribution over the next observation conditioned on a history of past observations and actions. Given a history window of length H discrete time steps, the model input is a sequence of past observations and actions

$$\mathbf{h}_t = (\mathbf{o}_{t-H+1}, \mathbf{a}_{t-H+1}, \ldots, \mathbf{o}_t, \mathbf{a}_t) \tag{3}$$

The model then learns

$$p_\theta(\mathbf{o}_{t+1} \mid \mathbf{h}_t) \tag{4}$$

from a dataset of recorded transitions. This is the learned analog of the paired dynamics and measurement models of a navigation filter: a single network that ingests raw measurements and actions (control inputs) and predicts the distribution over what the sensors will report next. Because the prediction is a distribution rather than a point, the model represents both the stochasticity of the dynamics and the noise of the sensors, and the spread of its samples provides a per-timestep uncertainty estimate.

3.2 World Model Architecture

Fig. 2 presents the Out-of-this-World-Model architecture. The design is modality-parameterized: each input stream contributes a fixed number of tokens to a per-timestep token vector, and the same backbone serves any combination of streams. A lightweight multilayer perceptron projects the kinematic state measurements, and a vision transformer encodes each camera frame into image tokens through a learned attention bottleneck that cross-attends a small set of latent queries against the frame's patch embeddings. The concatenation of the state and image tokens for each timestep is the model's latent state $\mathbf{z}_t$. A second multilayer perceptron projects the action into an additional token that is appended to the same per-timestep vector as conditioning; it is fused with the state and image tokens by the backbone's spatial attention.

The backbone is a factorized space–time transformer in the style of video architectures. Each block applies spatial attention—bidirectional attention among the tokens of a single timestep, which lets state, action, and image information fuse—followed by temporal attention, which attends causally across timesteps within a sliding window using

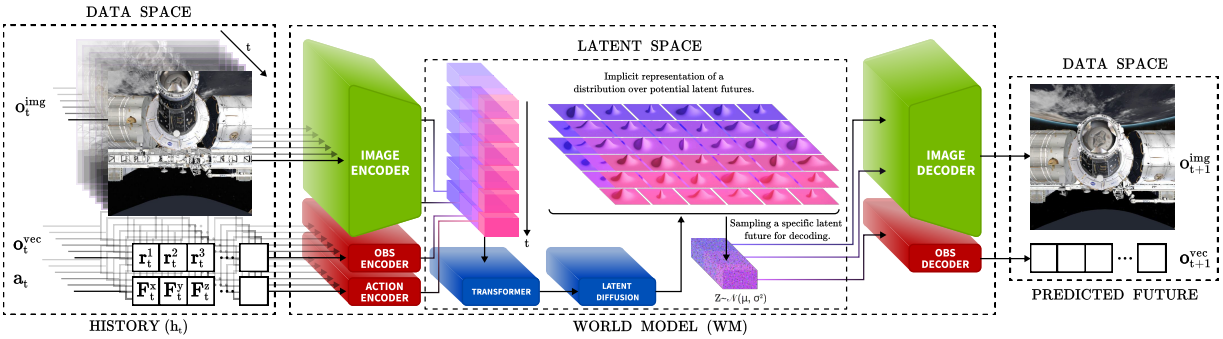


Fig. 2: The Out-of-this-World-Model architecture. A history window $\mathbf{h}_t$ of camera frames, kinematic state measurements, and actions is encoded into a vector of latent tokens per timestep. A factorized space–time transformer attends across tokens within each timestep and causally across timesteps. A flow-matching head transports Gaussian noise to a sample of the latent residual $\Delta\hat{\mathbf{z}}_{t+1}$; added to the current latent this gives the next latent state $\hat{\mathbf{z}}_{t+1} = \mathbf{z}_t + \Delta\hat{\mathbf{z}}_{t+1}$, and repeated draws with different noise realizations produce a distribution over futures. The sampled latent is appended to the token history and the model rolls forward autoregressively in latent space; an observation decoder maps any latent back to a predicted image and state with an associated uncertainty.

rotary position embeddings. Factoring the attention this way reduces the cost from quadratic in the full token sequence to quadratic in each axis separately. A key–value cache makes autoregressive rollouts linear in horizon length, which addresses throughput bottlenecks when the model is queried inside a sampling-based planner.

Prediction is performed in latent space with a flow-matching head [22, 23], applied independently to each token of the vector. Conditioned on the backbone output for the current timestep, the head learns a velocity field that transports a standard Gaussian sample to the residual between the next latent state and the current one. The sampled latent is appended to the token history and the model rolls forward—latent-space autoregression—without ever decoding to observations in the loop. A decoder head per modality maps latents back to predicted images and states when observation-space output is needed.

3.3 Training

The model trains on windows of H context steps followed by F rollout steps sampled from the recorded trajectories. The flow head is trained with the rectified flow objective on the latent residual $\Delta\mathbf{z}_{t+1} = \mathbf{z}_{t+1} - \mathbf{z}_t$. For a noise sample $\varepsilon \sim \mathcal{N}(0, I)$ and a noise level $\tau \sim \mathcal{U}(0, 1)$, where 0 is no added noise and 1 is pure Gaussian noise, the noised input

$$\mathbf{x}_\tau = (1-\tau)\,\Delta\mathbf{z}_{t+1} + \tau\,\varepsilon \tag{5}$$

lies on the straight path between data and noise, and the head f_θ minimizes

$$\mathcal{L}_{\text{flow}} = \left\| f_\theta(\mathbf{x}_\tau, \tau, \mathbf{h}_t) - (\varepsilon - \Delta\mathbf{z}_{t+1}) \right\|^2, \tag{6}$$

which regresses the constant velocity of that path.

Two families of reconstruction terms ground the latent space in observations. Let g_m denote the decoder head for modality m, $\hat{\mathbf{z}}_t$ the latent the model predicts at step t, and $\mathbf{z}_t = E(\mathbf{o}_t)$ the encoding of the true observation $\mathbf{o}_t$, whose modality-m component is $\mathbf{o}_t^m$. The per-modality decode loss reconstructs each observation stream from the *predicted* latent,

$$\mathcal{L}_m^{\text{dec}} = \frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \left\| g_m(\hat{\mathbf{z}}_t) - \mathbf{o}_t^m \right\|^2, \tag{7}$$

where $\mathcal{S}$ is a random subset of the rollout steps drawn each iteration to bound decoder compute (the camera image is decoded by a deterministic mean-squared-error decoder; the kinematic state by its decode head). The latent round-trip anchor instead reconstructs the *true* observation from its own encoding, with no dynamics in the loop,

$$\mathcal{L}_m^{\text{rt}} = \frac{1}{T} \sum_t \left\| g_m(\mathbf{z}_t) - \mathbf{o}_t^m \right\|^2, \tag{8}$$

which keeps the token codec close to an identity map so the shared latent stays faithful to the raw observations; we apply it to the image stream.

All terms combine as a weighted sum,

$$\mathscr{L} = \lambda_{\text{flow}}\,\mathscr{L}_{\text{flow}} + \sum_m w_m\,\mathscr{L}_m^{\text{dec}} + \sum_m \beta_m\,\mathscr{L}_m^{\text{rt}}, \tag{9}$$

with $\lambda_{\text{flow}} = 1$ and unit decode weights $w_m = 1$; the image round-trip anchor is weighted $\beta = 10$ (streams without an anchor take $\beta_m = 0$), since at unit weight it was a negligible fraction of the objective.

Training rolls the model forward autoregressively on its own predictions, with a teacher-forcing probability annealed from one to zero over the first epochs of training so the model learns to consume its own outputs. We train with AdamW at a learning rate of 10^{-3} with weight decay of 10^{-4}, a linear warmup over the first 300 steps, mixed `bfloat16` precision, and gradient-norm clipping at 1.0. Validation runs in fully autoregressive mode with deterministic sampling, so the monitored metrics reflect deployment conditions rather than teacher-forced ones. The models trained in this work use a token dimension of 128, a four-block space–time transformer backbone with eight attention heads and a 32-step sliding attention window, together with a separate two-block, four-head rectified-flow latent denoiser, totaling approximately 6.4M trainable parameters. On a single NVIDIA H100 an epoch takes 1.5 hours, with model convergence usually happening within the first 16 epochs.

3.4 GPU-Accelerated Astrodynamics Simulation

World models typically require hundreds-of-thousands to millions of state-action transitions for successful training, which places a premium on simulation throughput. We generate training data with `outofthisworldmodel-envs`[2], an open-source ISS-docking simulation environment whose dynamics, sensing, reward, and episode logic are written entirely in JAX [42], so that environments can execute in parallel on CPUs or GPUs and entire trajectory rollouts compile to single fused programs. GPU parallelization enables significantly more throughput than is easily achievable with CPU-only simulation. The environment exposes a standard Gymnasium interface [43] for direct integration with the wider reinforcement learning ecosystem, including a natively vectorized variant whose step, reset, observation, and reward paths are all batched rather than dispatched to parallel copies of a single environment.

3.4.1 AstroJAX

Many learning-based approaches rely on simple Hill–Clohessy–Wiltshire (HCW) dynamics to reduce computation and maximize simulation throughput, however this removes the more complex perturbations that shape relative motion over longer periods. Models that capture these perturbations may generate better policies, and, as a result, an astrodynamics simulation framework capable of high-throughput, high-fidelity parallel simulation is a key enabling technology for policy improvement and learning. To address this need, we developed *AstroJAX*[3], an open-source astrodynamics library that expresses standard astrodynamics models as pure, differentiable JAX functions. Table 1 inventories the implemented models.

We validate AstroJAX against the Rust-backed brahe astrodynamics library [44]. The frame transformations reproduce IAU SOFA reference values to 10^{-8}, and the gravity, third-body, solar radiation pressure, and drag accelerations agree with brahe to relative tolerances between 10^{-9} and 5×10^{-4} depending on the model. The one exception is NRLMSISE-00 atmospheric density model implementation, needed for drag perturbation modeling, where the comparison is currently bounded near 15% agreement.

3.4.2 ISS Docking Environment

The environment suite models a Dragon-class chaser maneuvering in the vicinity of the ISS. Three environments share a single task layer—the reward, docking criteria, collision geometry, sensor models, and reset distributions are identical—and differ only in the fidelity of their equations of motion. The results in this paper use the highest-fidelity member, `iss-numerical`, a full numerical simulation of the two-vehicle system.

[2] Available at https://github.com/sisl/outofthisworldmodel-envs
[3] Available at: https://github.com/duncaneddy/astrojax

Table 1: Models implemented in AstroJAX.

Category	Models
Gravity	Point-mass, spherical harmonic, polyhedral
Perturbations	Atmospheric drag (Harris–Priester, NRLMSISE-00), third-body Sun and Moon, SRP with cylindrical or conical shadow
Ephemerides	Analytic Sun and Moon, JPL approximate planetary positions
Attitude	Quaternion kinematics, rigid-body Euler dynamics, gravity-gradient torque
Relative motion	Hill–Clohessy–Wiltshire dynamics, ECI–RTN transforms, quasi-nonsingular relative orbital elements
Frames & time	IAU 2006/2000A GCRF–ITRF with full Earth orientation data support, TEME, geocentric, and geodetic coordinates,
Propagation	RK4, RKF4(5), Dormand–Prince 5(4), RKN12(10) integrators; SGP4/SDP4

In `iss-numerical`, the chief and chaser are propagated as independent state vectors in the Earth-centered inertial (ECI) frame through the same perturbed force model, which comprises zonal gravity harmonics through degree four (terms through J_6 are available), third-body accelerations from the Sun and Moon, and atmospheric drag with Harris–Priester density. The chaser's attitude is propagated inertially with quaternion kinematics and rigid-body Euler dynamics. The full simulation state is 21-dimensional—the epoch (defined by Modified Julian Day epoch and elapsed seconds), the chief and chaser ECI positions and velocities, the body-to-inertial quaternion, and the body rates—and integrates with a fourth-order Runge–Kutta scheme at a timestep of 0.05 s in double precision, with episodes running at most 7200 steps (360 s). The chief flies an ISS reference orbit with a 6795 km semi-major axis at 51.64° inclination. The task layer operates on a canonical 13-dimensional relative view derived from this state,

$$\mathbf{s} = (\mathbf{r}, \mathbf{v}, \mathbf{q}, \boldsymbol{\omega}) \tag{10}$$

comprising the relative position $\mathbf{r}$ and velocity $\mathbf{v}$ of the chaser with respect to the station origin in the rotating station-centered frame, the body-to-world attitude quaternion $\mathbf{q}$, and the body-frame angular velocity $\boldsymbol{\omega}$. The control

$$\mathbf{a} = (\mathbf{F}, \tau) \tag{11}$$

is six-dimensional, comprising a body-frame force $\mathbf{F}$ and torque τ. Actuator limits are sized to Draco-class reality: a proximity-operations translation fires roughly four of Dragon's sixteen 400 N thrusters along a body axis, giving a per-axis force limit of 1600 N, and a thruster couple at a 2.5 m arm gives a torque limit of 2000 N m. The chaser has a mass of 12,000 kg and principal inertias of 80,000 kg m^2, 80,000 kg m^2 and 50,000 kg m^2.

Collision checking sweeps the chaser's 2.25 m bounding sphere along its full step displacement against a 313-box axis-aligned decomposition of the ISS geometry, so a fast chaser cannot tunnel through thin structure between timesteps. Docking succeeds when the chaser holds within 0.1 m of the port pose at under 0.5 m/s, within 5° of the port attitude, and under 0.5 °/s of body rate. The station model is a NASA-derived ISS asset in its February–May 2015 configuration, on which eight visiting-vehicle ports are reachable targets, spanning all three docking mechanism families: the IDSS port at Harmony forward (PMA-2), the CBM berths at Harmony zenith, Harmony nadir, and Unity nadir, and the SSVP probe-and-drogue ports at Zvezda aft, Poisk zenith, Pirs nadir, and Rassvet nadir. The reward is a weighted sum of position, velocity, attitude, and body-rate errors, each passed through a smoothed pseudo-Huber cost, with large terminal payments for collision, successful docking, and escape; Section 4.3 gives the functional form in Equation (12) together with the weights used for data generation and for reinforcement learning training. The attitude and rate terms are gated by distance so that pointing barely matters at range and matters fully at the port, and the position term targets the assigned dock pose rather than the station origin. The world model never trains on the reward.

Observations follow the formulation of Section 3.1. The kinematic channel reports the 13-dimensional relative view through a configurable Gaussian sensor model described in Section 4, optionally augmented with a 12-dimensional dock-goal error block. The visual channel is a 512×512 first-person view from a camera mounted on the chaser's nose looking along the body approach axis with an 82° vertical field of view, rendered by a physically based renderer that includes the full-globe textured Earth, starfield, Moon, and epoch-dependent Sun lighting, so the model sees the same lighting variation an approach camera would. Frames are rendered from the recorded simulation states when datasets are written, at the 20 Hz simulation rate.

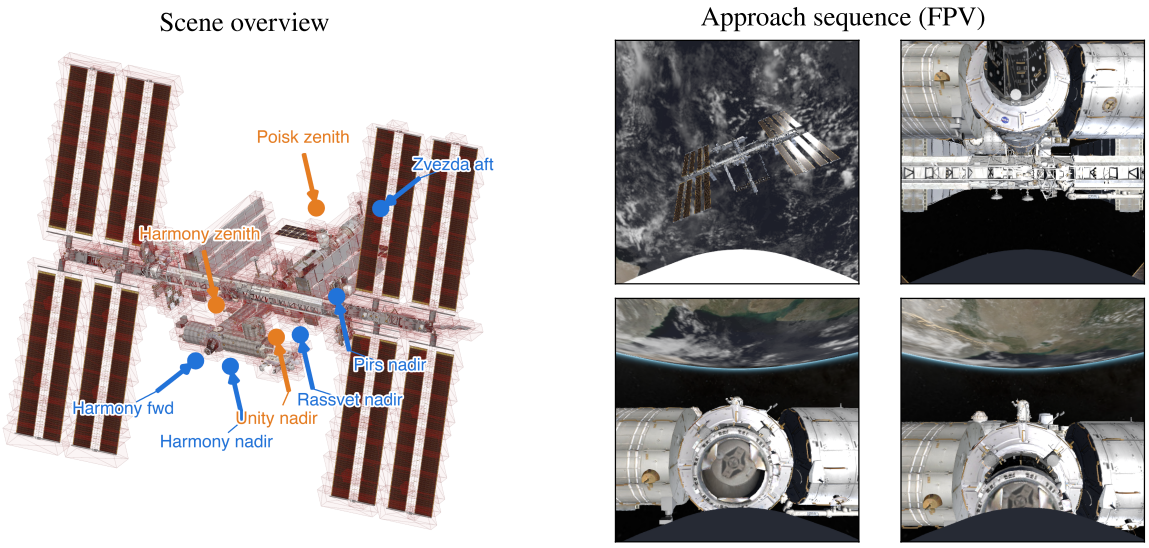


Fig. 3: The ISS docking environment. Left: isometric view of the station model overlaid with the 313 axis-aligned collision boxes (red) used for keep-out checking and the goal poses of the eight docking ports. Blue markers label the five ports flown in the training data and orange markers the three ports held out for evaluation; arrows indicate the approach corridors. Right: the chaser's first-person camera view—the visual observation the world model consumes—at four points along a successful docking approach to the Harmony forward port.

The two remaining environments trade fidelity for speed. The `iss-hcw` environment replaces the numerical propagation with Hill–Clohessy–Wiltshire relative dynamics [45], adding the gravity-gradient torque, and the `iss` environment drops orbital mechanics entirely, propagating a rigid-body free-flyer whose translation reduces to a double integrator under thrust.

4. EXPERIMENTS

We evaluate the approach on the ISS docking scenario across three sensor noise regimes, comparing world model planning against reinforcement learning baselines and measuring prediction quality, docking performance, generalization to unseen ports, and anomaly detection. This section describes the data generation, training configuration, baselines, and evaluation criteria.

4.1 Data Generation

Fig. 3 presents the simulation environment. Each episode initializes the chaser at a uniformly sampled radius between 100 m to 225 m from the station origin with the nose pointed at the station and an epoch drawn uniformly from a seven-day window—which sweeps the solar beta angle and with it the lighting the camera sees—and terminates on docking, collision, departure beyond 750 m, or the 360 s horizon. Trajectories are generated by three behavior policies, chosen so that the training distribution covers both goal-directed and undirected motion. The *random* policy samples uniform force and torque commands, producing undirected drift that covers the state space far from the station. The *orbit* policy commands a proportional-derivative-tracked circumnavigation of the station at a sampled radius between 80 m to 130 m of the station, exposing the model to sustained lateral motion and the full range of viewing geometries. The *dock* policy flies a critically damped proportional-derivative approach to a sampled docking port. About half of these approaches meet the contact conditions and the remainder end in recorded collisions with station structure, so the corpus also covers the contact-adjacent failure modes the model must learn to predict. Fig. 4 shows the training episodes from each policy.

The training split mixes the three policies in a $0.30/0.35/0.35$ ratio and its dock lane flies to five of the eight ports—Harmony forward, Harmony nadir, Zvezda aft, Pirs nadir, and Rassvet nadir. The three remaining ports, Harmony zenith, Poisk zenith, and Unity nadir, appear only in evaluation, giving a held-out generalization test whose goal poses, approach corridors, and visual context the model has never seen. Table 2 summarizes the generated datasets.

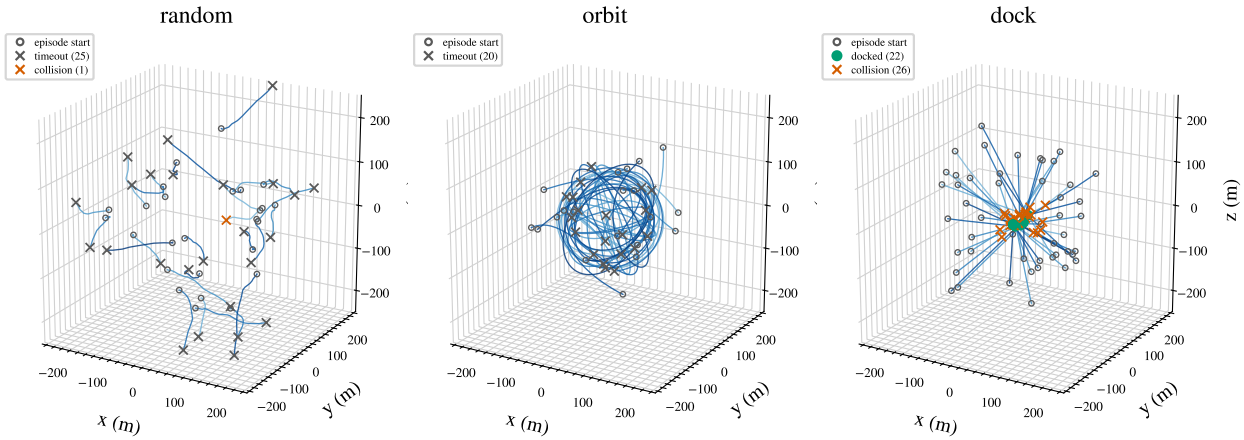


Fig. 4: Training trajectories from the published dataset in the station-centered frame, one panel per behavior policy, with episode starts and outcome-coded episode ends. Random walks drift across the domain without goal direction, orbit trajectories circumnavigate the station across a range of radii and plane orientations, and dock trajectories converge on the sampled port poses, with approaches that clip station structure recorded as collisions.

Table 2: Generated datasets. Transition counts are minimum targets; Training generation runs full episodes to until the target transition count is met, so actual transition count for realized dataset may be slightly larger.

Split	Transitions	Policy mix	Dock ports
train	500,000	random/orbit/dock (0.30/0.35/0.35)	5
val	50,000	dock	8

Table 3: Sensor noise models. Values are total-RMS errors.

Preset	Position	Velocity	Attitude	Body rate
No noise	0	0	0	0
Cooperative	0.05 m	0.002 m/s	5×10^{-5} rad	1×10^{-5} rad/s
Non-cooperative	1% of range	0.03 m/s	5×10^{-5} rad	1×10^{-5} rad/s

Each frame records the true state, the noisy observation vector, the action, the reward, and the rendered first-person camera view, packaged in LeRobot format with normalization statistics computed on the training split alone.

Sensor noise follows one of three models, summarized in Table 3, for training and evaluation. The behavior policies act on the noisy measurement rather than on the true state, so the recorded action-outcome pairs show the aleatoric transition uncertainty of acting on an observed state rather than the true state, which would result in deterministic dynamics. The *cooperative* model represents differential-GNSS-class relative navigation with a fixed position error budget, as flown between vehicles that exchange carrier-phase measurements. The *non-cooperative* model is based on vision-based navigation, in which position error grows with range and the velocity estimate is coarser. Attitude and rate noise, which are assumed to come from the chaser's own star tracker and gyroscopes, are identical across the two presets. Table 3 lists the values. All noise magnitudes are total root-mean-square errors—the Euclidean norm of the error vector—matching the convention of navigation requirements documents. Finally there is a *no noise* configuration in which no kinematic noise is added, enabling isolation and quantification of world-model reconstruction errors.

4.2 Training Configuration

World models train on windows of $H = 8$ context steps and $F = 64$ rollout steps with the procedure of Section 3.3: AdamW at a learning rate of 10^{-3}, 300-step linear warmup, mixed `bfloat16` precision, gradient clipping at 1.0, and teacher forcing annealed to zero over the first four epochs. Batch sizes are tuned automatically to fill available GPU memory. Each model trains for 16 epochs on one NVIDIA H100 GPU with 96 GB of memory, completing in approx-

imately 24 hours; data generation for the 500,000-transition corpus, including rendering, completes in approximately 6 hours on the same hardware.

4.3 Baseline Algorithms

We compare against a model-free reinforcement learning baseline trained directly on the environment: proximal policy optimization (PPO) [46], in its Stable-Baselines3 implementation [47]. The policy is a multilayer perceptron consuming the normalized 27-dimensional vector observation (the epoch, the noisy relative state view, and the dock-goal error block) and trained separately for each the cooperative and non-cooperative sensor noise models, on the five training-data ports and a start shell of 100 m to 125 m. The goal-error block is itself an augmentation of the default model-free formulation: without an input identifying the commanded port, a trained policy has no mechanism for generalizing to a new docking task at all, so we condition the baseline on the goal state to make port transfer expressible in the first place.

Obtaining a reinforcement learning baseline that makes progress on this task at all required significant adaptation and tuning of the training configuration. We document the adaptations for reproducibility and because they measure the challenge with implementing the approach. The environment's native configuration defeats temporal-difference learning: at the 20 Hz simulation step the terminal docking bonus sits 7200 steps out, where at $\gamma = 0.995$ it is discounted by $\gamma^{7200} \approx 2 \times 10^{-16}$, a contribution that vanishes against the accumulated per-step shaping in single-precision arithmetic, so the policy see no contribution from successful docks near the end of the horizon. The baseline therefore trains at a 1 Hz control step, integrating the same dynamics over the same 360 s horizon in 360 decisions rather than 7200, the cadence used in prior reinforcement learning docking work [28, 48], with the keep-out zone made soft, so that collisions are charged per step rather than terminating the episode. Finally, since past works only considered much simpler keep-out zones and restricted dynamics to only orbital (no attitude) dynamics, we had to spend significant effort shaping a reward function that successfully encouraged docking in a correct orientation over loitering at a safe distance.

The reward used for baseline training is, per step,

$$\begin{aligned} r_t = {} & w_{\text{pos}}\, h(d_t; \delta_{\text{pos}}, \sigma_{\text{pos}}) + w_{\text{vel}}\, h(\|\mathbf{v}_t\|; \delta_{\text{vel}}, \sigma_{\text{vel}}) + g(d_t)\left[w_{\text{att}}\, h(\theta_t; \delta_{\text{att}}, \sigma_{\text{att}}) + w_{\text{rate}}\, h(\|\boldsymbol{\omega}_t\|; \delta_{\text{rate}}, \sigma_{\text{rate}})\right] \\ & + w_{\text{prog}}\, \frac{d_t - d_{t-1}}{\sigma_{\text{pos}}} + e^{-d_t/\sigma_{\text{prox}}} \left(w_{\text{prox}} + w_{\text{align}}\, e^{-\theta_t/\sigma_{\text{align}}}\right) + w_{\text{eff}} \left(\frac{\|\mathbf{F}_t\|}{F_{\max}} + \frac{\|\boldsymbol{\tau}_t\|}{\tau_{\max}}\right) \\ & + w_{\text{col}}\, \mathbf{1}_{\text{col}} + w_{\text{dock}}\, \mathbf{1}_{\text{dock}} + w_{\text{esc}}\, \mathbf{1}_{\text{esc}} \end{aligned} \tag{12}$$

where d_t is the distance to the assigned port position, θ_t the attitude error to the port attitude, $\mathbf{v}_t$ and $\boldsymbol{\omega}_t$ the relative velocity and body rate, and $\mathbf{F}_t$ and $\boldsymbol{\tau}_t$ the commanded force and torque against their actuator limits ($F_{\max} = 1600\,\text{N}$, $\tau_{\max} = 2000\,\text{N m}$). Each error norm passes through the normalized pseudo-Huber loss

$$h(e; \delta, \sigma) = \frac{\sqrt{e^2 + \delta^2} - \delta}{\sigma}, \tag{13}$$

which is quadratic below the knee δ and has unit far-field slope over the scale σ, so the far field keeps a constant pull toward the port while the near field stays smooth. The rotational terms are gated by

$$g(d) = g_\infty + \frac{1 - g_\infty}{1 + (d/d_g)^2} \tag{14}$$

so that pointing matters fully at the port and fractionally at range. The potential-based progress term [48] pays for range closed when it is closed—without it, tuned agents trained for 20 million steps learned to stop escaping but never to approach. The proximity and alignment bonuses are a smooth, discount-reachable precursor of the docking event, paying up to $w_{\text{prox}} + w_{\text{align}}$ per step for sitting on the port pointed correctly. The indicators fire on collision, docking, and departure beyond the domain boundary. Table 4 lists the parameter values.

Hyperparameters were selected by Bayesian optimization with Hyperband early termination [49]; Table 5 lists the frozen configuration, with the entropy coefficient floored at 10^{-3} and a target KL divergence of 0.02 so that exploration does not collapse over a long run. The policy trains for at most 25 million environment steps under each noise preset. The cooperative policy is evaluated at its best-validation-return checkpoint, at 22.5 million steps.

The world model plans with model predictive path integral (MPPI) control [16]. At each replanning step the planner perturbs its nominal action sequence with Gaussian noise to produce K candidate sequences, rolls each through the

Table 4: Reward parameters used for reinforcement learning baseline training (Equation (12)).

Term	Weight	Shaping
Position	$w_{\text{pos}} = -0.9$	$\delta_{\text{pos}} = 1\,\text{m}$, $\sigma_{\text{pos}} = 225\,\text{m}$
Velocity	$w_{\text{vel}} = -0.05$	$\delta_{\text{vel}} = 0.1\,\text{m/s}$, $\sigma_{\text{vel}} = 5\,\text{m/s}$
Attitude	$w_{\text{att}} = -0.2$	$\delta_{\text{att}} = 0.05\,\text{rad}$, $\sigma_{\text{att}} = \pi\,\text{rad}$
Body rate	$w_{\text{rate}} = -0.02$	$\delta_{\text{rate}} = 0.005\,\text{rad/s}$, $\sigma_{\text{rate}} = 0.05\,\text{rad/s}$
Rotation gate	—	$g_\infty = 0.5$, $d_g = 25\,\text{m}$
Progress	$w_{\text{prog}} = -2.0$	shares σ_{pos}
Proximity	$w_{\text{prox}} = +0.5$	$\sigma_{\text{prox}} = 1\,\text{m}$
Alignment	$w_{\text{align}} = +0.5$	$\sigma_{\text{align}} = 0.1\,\text{rad}$
Effort	$w_{\text{eff}} = -0.05$	normalized by actuator limits
Collision	$w_{\text{col}} = -2$ per step	non-terminating
Dock success	$w_{\text{dock}} = +1000$	terminating
Escape	$w_{\text{esc}} = -1000$	terminating

Table 5: PPO baseline configuration.

Parameter	Value
Total environment steps	25M
Parallel environments	32
Start shell	100 m to 125 m
Learning rate	2.2×10^{-4}
Discount γ	0.997
Network	256×3
Batch size	1024
Rollout length	1024
GAE λ	0.945
Clip range	0.1
Epochs per update	20
Value loss coefficient	0.67
Entropy coefficient	10^{-3}
Target KL divergence	0.02

world model in latent space, scores the decoded state predictions against the docking objective, and updates the nominal sequence with the softmax-weighted average

$$\mathbf{a}^\star = \sum_{k=1}^{K} w_k \, \mathbf{a}_k \qquad w_k = \frac{\exp(R_k/\lambda)}{\sum_j \exp(R_j/\lambda)} \tag{15}$$

where R_k is the predicted return of candidate k and λ is a temperature. Because the world model rolls all K candidates as one batch with a shared encoded context and a key–value cache, a full replanning step is a single batched forward pass. The planner executes a short chunk of the optimized sequence open-loop, then re-encodes the new observation and replans. We use a planning horizon of 64 steps, $K = 128$ candidate sequences, a temperature of $\lambda = 0.3$, and execute 8-step chunks between replans.

4.4 Evaluation Criteria

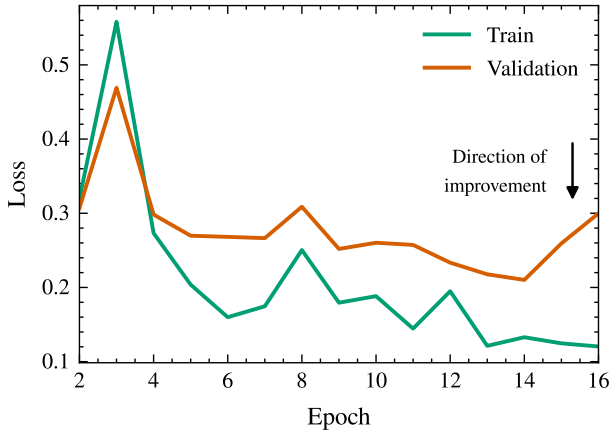
We evaluate along four axes:

1. **Prediction quality.** Open-loop state reconstruction error and image reconstruction quality (mean-squared error and peak signal-to-noise ratio) on held-out trajectories, as a function of training epoch and of prediction horizon, under each of the three noise presets. Comparing the no-noise, cooperative, and non-cooperative presets isolates how much sensor noise—rather than model capacity—limits predictive accuracy, and how noise degrades position and velocity prediction differently.

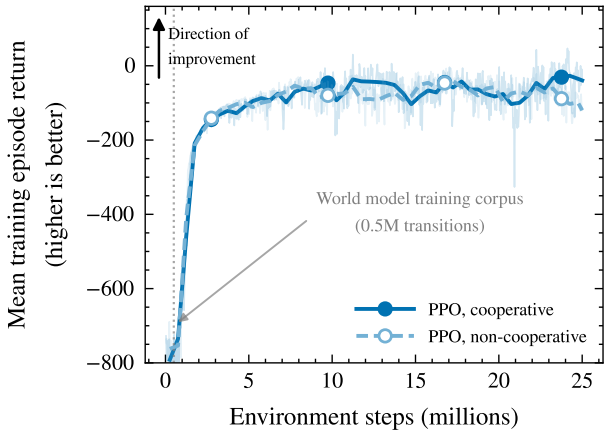
2. **Docking with in-distribution goals.** Each method flies 50 rollouts per port for the five ports in the training data under the cooperative noise preset. Because the full contact conditions of Section 3.4.2 proved out of reach for initial reinforcement learning baseline attempts, we report success at graduated requirement levels: an attempt passes the position-only definition at tolerance d if its closest approach to the goal pose comes within d, and passes the full definition if the velocity, attitude, and body-rate bounds also hold at that approach. Any episode in which the chaser contacts station structure is counted as a failure under every definition. We report the position-only rate at $d = 0.5\,\text{m}$ per port, sweep d from the 0.1 m contact gate outward to trace each method's precision profile under both definitions, and report the collision rate per port alongside.

3. **Docking with out-of-distribution goals.** The same protocol on Harmony zenith, Poisk zenith, and Unity nadir, which no policy visited during training. Reinforcement learning policies must generalize zero-shot through their goal-error input; the world model receives only a new goal pose in the planner's objective.

4. **Anomaly detection.** We render evaluation episodes in which an uninvolved visiting vehicle, a second Dragon spacecraft, is present in the environment, and measure whether the world model's predictive uncertainty flags the anomaly. We run 20 rollouts including the anomalous docked vehicle (even if not physically possible) and report the detection rate defined by the uncertainty exceeding a threshold calibrated on 25 nominal docking episodes.



(a) World model training and validation loss. We note that we stop training as validation loss starts to climb, and use the lowest validation model loss checkpoint for evaluations.



(b) Reinforcement learning baseline training return.

Fig. 5: Training curves. (a) Total training loss by epoch for the flow-matching OWM architecture and the Dreamer-style posterior-correction baseline. (b) Mean training episode return over environment steps for the PPO baseline under cooperative and non-cooperative sensor noise, averaged in bins of 5×10^5 steps over the raw rolling mean; the non-cooperative run is drawn dashed . The dotted line marks the equivalent size of the world model's training corpus on the same axis.

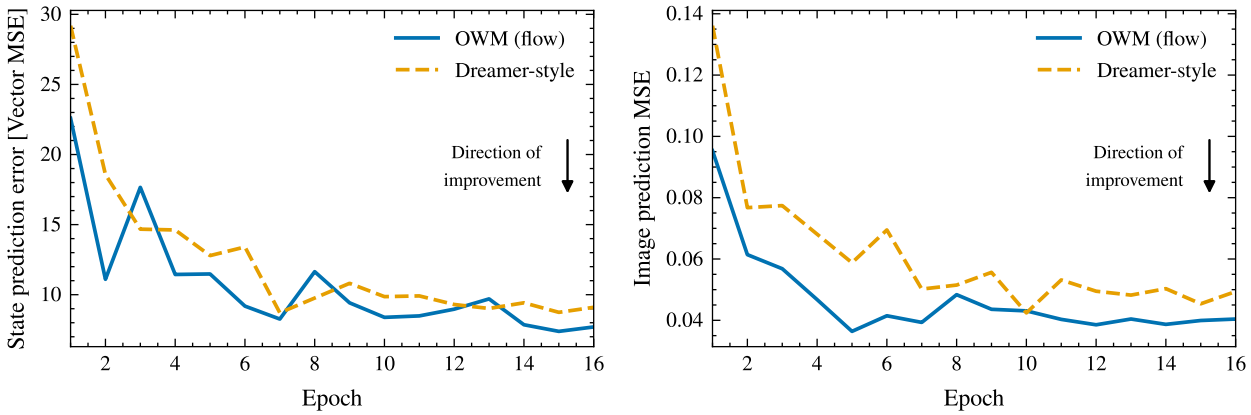


Fig. 6: World model validation performance by training epoch. Left: state reconstruction error on held-out trajectories under fully autoregressive rollout. Right: image reconstruction loss on the same rollouts.

5. RESULTS

5.1 Training and Prediction Quality

Fig. 5 presents the training curves for the world model variants and reinforcement learning baselines, and Fig. 6 presents validation reconstruction quality by epoch. The reinforcement learning baseline (Fig. 5b) makes most learning progress within the first 5 million steps and improves slowly thereafter; the cooperative policy is evaluated at its best-performing checkpoint on the held-out validation episodes (22.5 million steps). Both world model variants converge within 16 epochs. We find that the flow-matching architecture reaches lower validation prediction errors—for both state and image prediction—compared to the Dreamer-style baseline, while using fewer trainable parameters.

Fig. 7 shows prediction error as a function of rollout horizon under each noise preset. We find that the state prediction error is significantly degraded in the non-cooperative setting; it is consistently poorer across the prediction horizon. We expect that this is due to the noise perturbations being applied directly to the state observations. In general,

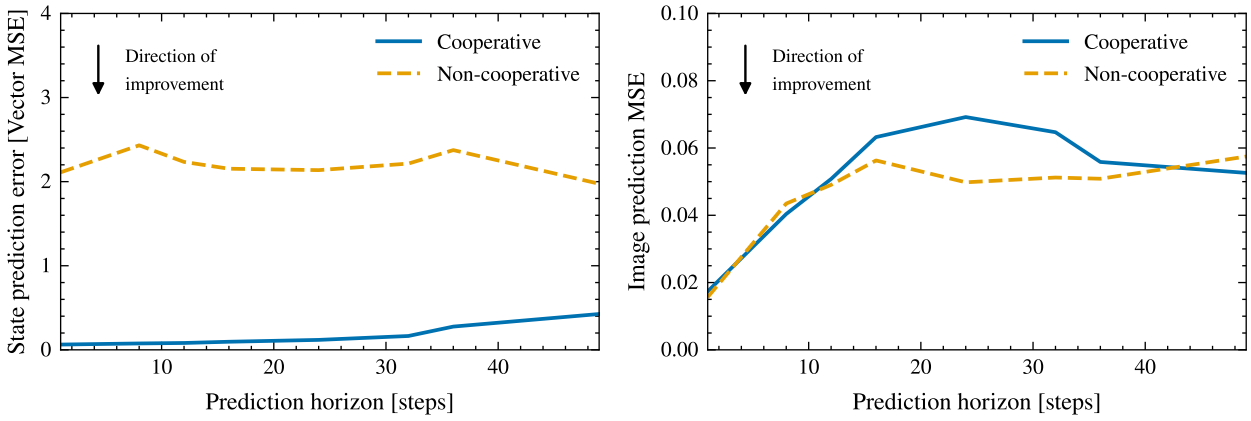


Fig. 7: Prediction error against rollout horizon for each sensor noise preset. Left: state prediction error. Right: image prediction loss. Non-cooperative range-proportional noise degrades long-horizon state prediction more than image predictions.

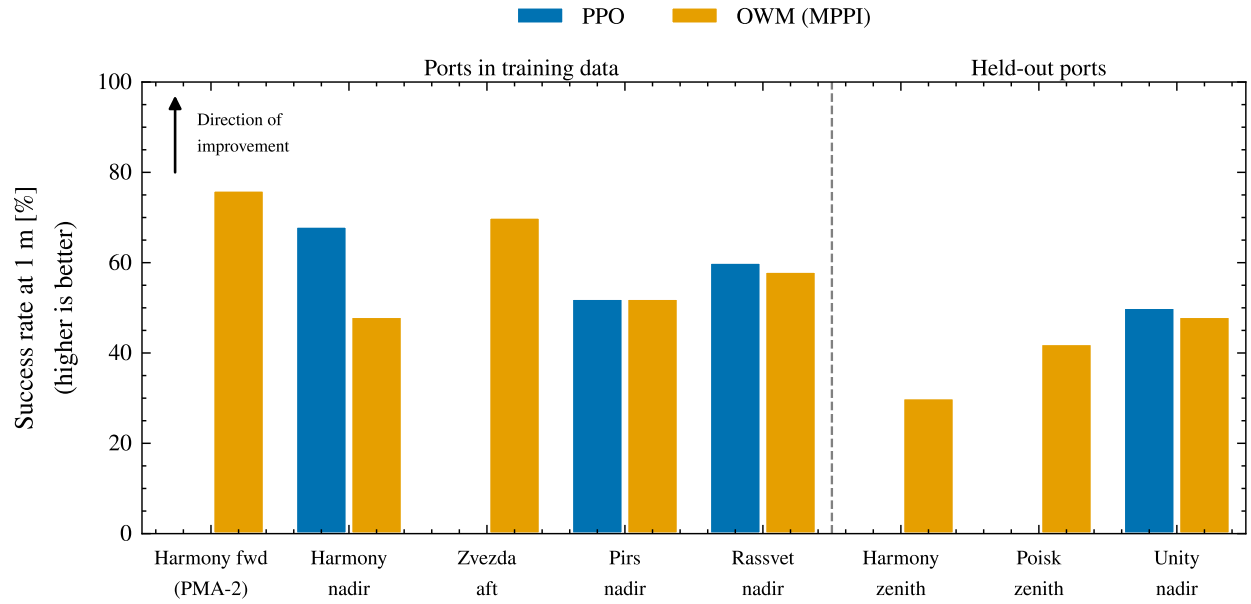


Fig. 8: Docking success rate by port and algorithm under cooperative sensor noise, over 50 rollouts per port. An attempt succeeds if its closest approach to the port comes within 1 m and the chaser never contacts station structure during the episode. Ports left of the divider appear in the training data; Harmony zenith, Poisk zenith, and Unity nadir are held out.

these perturbations greatly increase the MSE of our kinematic state predictions, but image prediction quality is nearly insensitive to the kinematic noise preset, which is expected: the camera channel is noise-free in all presets.

5.2 Docking Performance

Fig. 8 presents docking success rates by port under cooperative noise, with the five ports in the training data and the three held-out ports (Harmony zenith, Poisk zenith, and Unity nadir), which no training trajectory visited. Two patterns stand out. First, given only the port's goal pose in its planning objective and minimal controller tuning, the world-model planner is comparable to PPO for every port and docks at 4 ports the PPO policy never does. The world-model planner's collision-voided success at the 1 m tolerance on the training ports is 61% against 36% for PPO. It docks at Harmony forward (76%) and Zvezda aft (70%), two training ports the RL policy never reaches (its approaches stall 10 m to 12 m out), while PPO docks in earnest only at the three nadir-facing training ports (68%, 60%, and 52% at

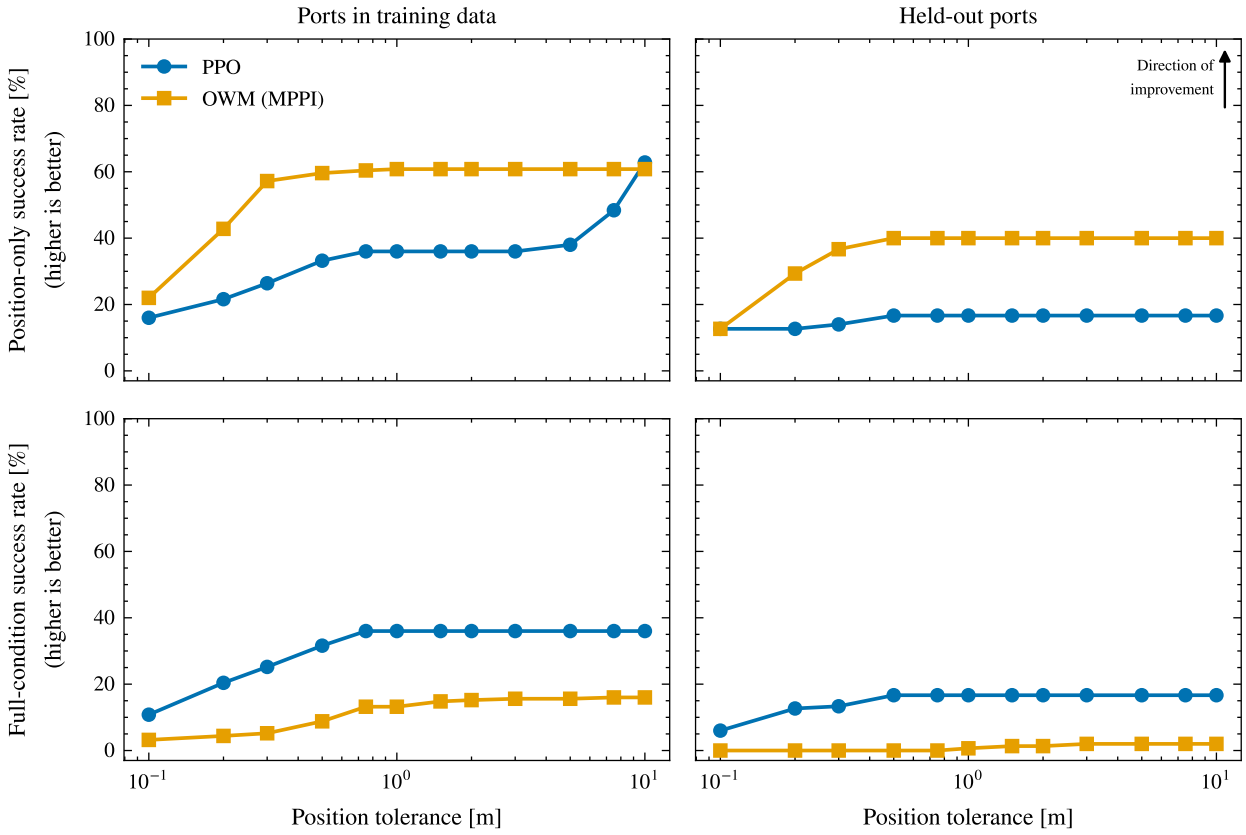


Fig. 9: Collision-voided success rate against position tolerance under cooperative sensor noise, pooled over the ports in the training data (left) and the held-out ports (right), over 50 rollouts per port. Top: position-only success, in which the closest approach comes within the tolerance. Bottom: the full contact conditions, which additionally bound the velocity, attitude error, and body rate at that approach. Curves are shown for the world-model planner and the PPO baseline. The two rows expose the central trade-off; the planner leads on position-only success through reliable acquisition, while PPO leads on the full contact conditions through its slower, better-pointed terminal approach.

Harmony, Rassvet, and Pirs nadir). Second, and more consequentially, the held-out ports expose a sharp difference in out-of-distribution generalization. The planner, which receives only a new goal pose and is otherwise unchanged, generalizes to all three: 48% at Unity nadir, 42% at Poisk zenith, and 30% at Harmony zenith. PPO The baseline, despite its goal-state conditioning, generalizes to exactly one held-out port, Unity nadir (50%), which shares the nadir approach direction of the Pirs and Rassvet ports in the training set, and fails outright at the two zenith ports, whose approach geometry resembles no training task. Reinforcement learning, that is, transfers only where a held-out port looks like a task it has already trained on; the world model transfers to every port it is pointed at, despite having seen none of them.

Fig. 9 sweeps the position tolerance of the success definition at two grades of strictness. Under position-only success (top row), the planner outperforms the baseline handily at every tolerance, on training and held-out ports alike, reflecting its reliable acquisition. Under the full contact conditions (bottom row), which additionally bound the velocity, attitude error, and body rate at closest approach, the ordering reverses: PPO's slower, better-pointed terminal approaches meet all four conditions in 36% of training-port rollouts at 1 m against the planner's 13% (concentrated at Harmony forward, 66%), and 11% against 3% at the strict 0.1 m contact gate. We attribute much of this gap to tuning effort rather than to the paradigm: the RL reward shaping was iterated over many training runs to produce slow, pointed arrivals, whereas the MPPI cost received far less tuning and favors a collision-free approach over arriving pointed. More careful weighting of the terminal attitude and body-rate costs should improve attitude-rate convergence and lift the planner's success under the full contact conditions.

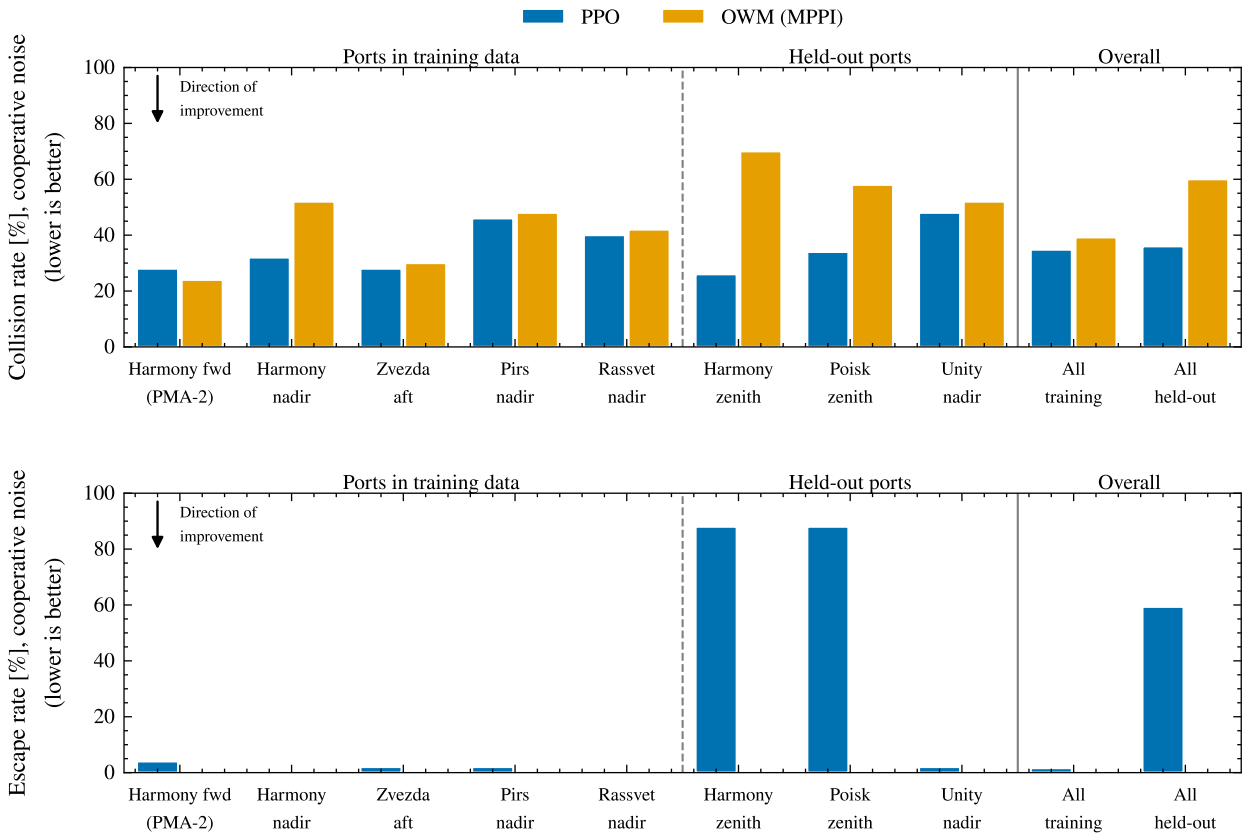
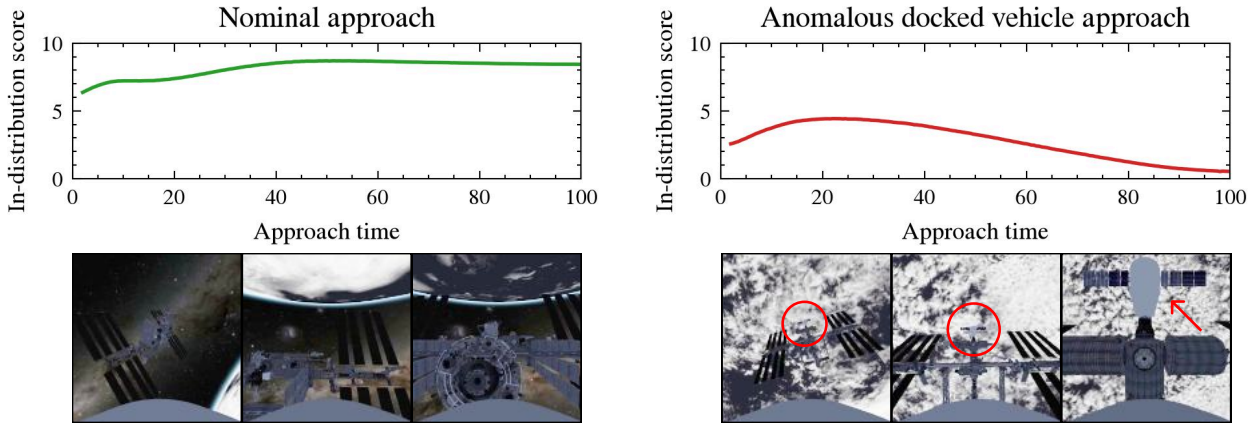


Fig. 10: Collision rate (top) and escape rate (bottom) by port and algorithm under cooperative sensor noise, over 50 rollouts per port. The collision rate is the fraction of rollouts in which the chaser entered a station keep-out box at any point; the escape rate is the fraction in which the chaser left the 750 m operational domain. The world-model planner never escapes, so every one of its rollouts presses a full approach and attempts a dock.

Fig. 10 reports the collision rate (top panel) and escape rate (bottom panel) at each port; the two panels must be read together. In isolation the planner's collision rate is the higher one (39% of training-port rollouts against PPO's 35%, and 60% against 36% on the held-out ports), but the escape rates show why. The planner never leaves the operational domain: 0% of rollouts across all eight ports, versus the baseline's 88% at each zenith port. Rollouts that escape before any close approach can never register a collision, so the baseline's low held-out collision rate reflects approaches it never made. The planner instead presses a full approach on every rollout, attempting many more docks precisely where the station geometry is tightest, and its residual failure mode is contact with structure on the zenith approaches rather than lost acquisition. Reducing this collision rate, whether through the cost tuning above or by feeding the model's predictive uncertainty back into the planner as a runtime safety signal, remains open work.

5.3 Anomaly Detection

Finally, we evaluate the world model's predictive uncertainty as a runtime anomaly monitor. Fig. 11 shows two representative episodes, and Fig. 12 the aggregated results across the full anomaly detection dataset. The world model, having never seen a second visiting vehicle during training, predicts a nominal scene, and the disagreement between prediction and observation concentrates precisely on the anomalous object. The model correctly classifies 20 approach sequences including an anomalous docked Dragon spacecraft from 25 approach sequences without any anomalous spacecraft with a classification accuracy of 98%.



(a) A nominal approach: the in-distribution score remains high throughout, indicating the world is evolving as the model predicted.

(b) An approach with an anomalous docked vehicle: the model, which never saw a docked vehicle in training, cannot predict its presence, and the score drops as the vehicle grows in the perceptual field.

Fig. 11: In-distribution scores as an anomaly signal on two representative approaches.

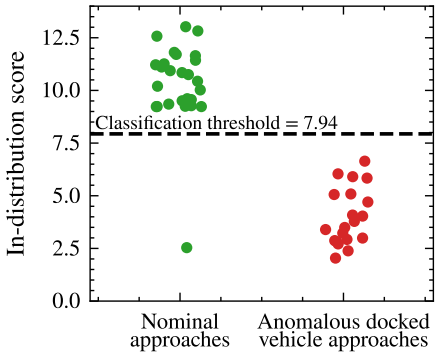


Fig. 12: Maximum in-distribution score per approach over the anomaly detection dataset. Approaches with an anomalous docked vehicle score consistently lower, and a scalar threshold classifies 98% of approaches correctly.

6. CONCLUSIONS

This paper introduces a world model approach to spacecraft rendezvous and proximity operations. We present Out-of-this-World-Model, a transformer-based architecture that fuses camera imagery and kinematic state measurements into a shared latent space and predicts distributions over future observations with a one-step flow-matching head, and we present AstroJAX, a JAX-based astrodynamics framework whose massively parallel GPU simulation generated the 500,000-transition training corpus used in this work. Applied to the problem of a capsule docking with the ISS, the learned world model composes with model predictive path integral control to reach berthing ports both included and excluded from the training distribution, where it more than doubles the docking success rate of reinforcement-learning baselines on held-out ports (40% versus 17%), succeeds at ports the baselines fail to acquire at all, and never leaves the operational domain. The same model's predictive uncertainty serves as a runtime anomaly monitor, correctly classifying approach sequences with and without previously unseen vehicles 98% of the time. Together, these results represent a step toward autonomous spacecraft that reason about the consequences of their actions before executing them.

Promising avenues for future work include more careful tuning of the MPPI cost weights, whose terminal attitude and body-rate terms received far less attention than the reinforcement-learning reward shaping and which we expect to both reduce the planner's collision rate and close its remaining gap under the full contact conditions; leveraging

the anomaly detection capability more directly for collision mitigation, so that rising predictive uncertainty tempers approach aggressiveness before contact occurs; applying the learned system model paradigm to pattern-of-life characterization for space domain awareness, where deviations from a satellite's predicted behavior signal a change in operational mode; and integration of world-model based planners with safety-aware planning techniques to further improve docking success rate.

ACKNOWLEDGMENTS

Duncan Eddy is supported by Stanford Institute for Human-Centered AI. Grace Kim is supported by the Fannie and John Hertz Foundation and the National Defense Science and Engineering Graduate (NDSEG) Fellowship Program. Specifically, this material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-25-C-B010 in the amount of currently negotiated tuition and stipend rates.

REFERENCES

[1] Greg Chavers, Lisa Watson-Morgan, Marshall Smith, Nantel Suzuki, and Tara Polsgrove. NASA's Human Landing System: The Strategy for the 2024 Mission and Future Sustainability. In *IEEE Aerospace Conference*, pages 1–9, 2020. doi: 10.1109/AERO47225.2020.9172599.

[2] Clayton Swope, Kari A. Bingen, Makena Young, and Kendra LaFave. Space Threat Assessment 2025. A report of the CSIS aerospace security project, Center for Strategic and International Studies (CSIS), Washington, DC, April 2025.

[3] Robert B. Friend. Orbital Express Program Summary and Mission Overview. In *Sensors and Systems for Space Applications*, page 695803, 2008. doi: 10.1117/12.783792.

[4] Matt Pyrak and Joseph Anderson. Performance of Northrop Grumman's Mission Extension Vehicle (MEV) RPO Imagers at GEO. In *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*, page 121150A, 2022. doi: 10.1117/12.2631524.

[5] Mohinder S. Grewal and Angus P. Andrews. Applications of Kalman Filtering in Aerospace 1960 to the Present. *IEEE Control Systems Magazine*, 30(3):69–78, 2010.

[6] John L Crassidis, F Landis Markley, and Yang Cheng. Survey of Nonlinear Attitude Estimation Methods. *AIAA Journal of Guidance, Control, and Dynamics*, 30(1):12–28, 2007.

[7] Avishai Weiss, Morgan Baldwin, Richard Scott Erwin, and Ilya Kolmanovsky. Model Predictive Control for Spacecraft Rendezvous and Docking: Strategies for Handling Constraints and Case Studies. *IEEE Transactions on Control Systems Technology*, 23(4):1638–1647, 2015. doi: 10.1109/TCST.2014.2379639.

[8] Hongyang Dong, Qinglei Hu, and Maruthi R. Akella. Safety Control for Spacecraft Autonomous Rendezvous and Docking under Motion Constraints. *Journal of Guidance, Control, and Dynamics*, 40(7):1680–1692, 2017. doi: 10.2514/1.G002322.

[9] Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.

[10] John R. Martin and Hanspeter Schaub. The Physics-Informed Neural Network Gravity Model Revisited: Model Generation III. In *AAS/AIAA Space Flight Mechanics Meeting*, 2023. AAS 23-110.

[11] Kirk Hovell and Steve Ulrich. Deep Reinforcement Learning for Spacecraft Proximity Operations Guidance. *AIAA Journal of Spacecraft and Rockets*, 58(2):254–264, 2021.

[12] Massimo Tipaldi, Raffaele Iervolino, and Paolo Roberto Massenio. Reinforcement Learning in Spacecraft Control Applications: Advances, Prospects, and Challenges. *Annual Reviews in Control*, 54:1–23, 2022. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2022.07.004.

[13] David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018. arXiv:1803.10122.

[14] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Control Tasks through World Models. *Nature*, 640:647–653, 2025. arXiv:2301.04104.

[15] Isaac R. Ward, Michelle Ho, Houjun Liu, Aaron Feldman, Joseph Vincent, Liam Kruse, Sean Cheong, Duncan Eddy, Mykel J. Kochenderfer, and Mac Schwager. Foundational World Models Accurately Detect Bimanual Manipulator Failures. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2026. arXiv:2603.06987.

[16] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information Theoretic MPC for Model-Based Reinforcement Learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721, 2017. doi: 10.1109/ICRA.2017.7989202.

[17] Yann LeCun. A Path towards Autonomous Machine Intelligence, Version 0.9.2. OpenReview, 2022. https://openreview.net/forum?id=BZ5a1r-kVsf.

[18] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15619–15629, 2023.

[19] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pages 21271–21284, 2020.

[20] Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning. In *International Conference on Learning Representations (ICLR)*, 2022.

[21] Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. LeWorldModel: Stable End-to-End Joint-Embedding Predictive Architecture from Pixels. *arXiv preprint*, 2026. arXiv:2603.19312.

[22] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.02747.

[23] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2209.03003.

[24] Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One Step Diffusion via Shortcut Models. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.12557.

[25] Jeremy Morton, Freddie D. Witherden, Antony Jameson, and Mykel J. Kochenderfer. Deep Dynamical Modeling and Control of Unsteady Fluid Flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018. arXiv:1805.07472.

[26] Jeremy Morton, Freddie D. Witherden, and Mykel J. Kochenderfer. Deep Variational Koopman Models: Inferring Koopman Observations for Uncertainty-Aware Dynamics Modeling and Control. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. arXiv:1902.09742.

[27] Kevin T. Carlberg, Antony Jameson, Mykel J. Kochenderfer, Jeremy Morton, Liqian Peng, and Freddie D. Witherden. Recovering Missing CFD Data for High-Order Discretizations Using Deep Neural Networks and Dynamics Learning. *Journal of Computational Physics*, 395:105–124, 2019. doi: 10.1016/j.jcp.2019.05.041. arXiv:1812.01177.

[28] Jacob Broida and Richard Linares. Spacecraft Rendezvous Guidance in Cluttered Environments via Reinforcement Learning. In *29th AAS/AIAA Space Flight Mechanics Meeting*, Ka'anapali, HI, 2019. AAS 19-462.

[29] Brian Gaudet, Roberto Furfaro, and Richard Linares. Reinforcement Learning for Angle-Only Intercept Guidance of Maneuvering Targets. *Aerospace Science and Technology*, 99:105746, 2020. doi: 10.1016/j.ast.2020.105746.

[30] Brian Gaudet, Richard Linares, and Roberto Furfaro. Adaptive Guidance and Integrated Navigation with Reinforcement Meta-Learning. *Acta Astronautica*, 169:180–190, 2020. doi: 10.1016/j.actaastro.2020.01.007.

[31] Lorenzo Federici, Boris Benedikter, and Alessandro Zavoli. Deep Learning Techniques for Autonomous Spacecraft Guidance during Proximity Operations. *Journal of Spacecraft and Rockets*, 58(6):1774–1785, 2021. doi: 10.2514/1.A35076.

[32] Giovanni Fereoli, Hanspeter Schaub, and Pierluigi Di Lizia. Meta-Reinforcement Learning for Spacecraft Proximity Operations Guidance and Control in Cislunar Space. *Journal of Spacecraft and Rockets*, 62(3):803–818, 2025. doi: 10.2514/1.A36100.

[33] Ross E. Allen, Yaron Rachlin, Jessica Ruprecht, Sean Loughran, Jacob Varey, and Herbert Viggh. SpaceGym: Discrete and Differential Games in Non-Cooperative Space Operations. In *2023 IEEE Aerospace Conference*, pages 1–11, Big Sky, MT, 2023. doi: 10.1109/AERO55745.2023.10115968.

[34] Mark A. Stephenson and Hanspeter Schaub. BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking. In *International Astronautical Congress (IAC)*, Milan, Italy, 2024. IAC-24-D1.2.4.

[35] Mark Stephenson, Daniel Huterer Prats, and Hanspeter Schaub. Autonomous Satellite Inspection in Low Earth Orbit with Optimization-Based Safety Guarantees. In *International Workshop on Planning and Scheduling for Space (IWPSS)*, 2025.

[36] Adam Herrmann and Hanspeter Schaub. Autonomous Small Body Science Operations Using Reinforcement Learning. *Journal of Aerospace Information Systems*, 21(10):815–828, 2024. doi: 10.2514/1.I011376.

[37] Daniel Huterer Prats, Hanspeter Schaub, and Chris Wheeler. Reinforcement Learning for Space-to-Space Surveillance: Autonomous Scheduling for Resident Space Object Imaging. In *Advanced Maui Optical and Space Surveillance Technologies (AMOS) Conference*, Maui, HI, 2025.

[38] Joseph Breeden and Dimitra Panagou. Guaranteed Safe Spacecraft Docking with Control Barrier Functions. *IEEE Control Systems Letters*, 6:2000–2005, 2022. doi: 10.1109/LCSYS.2021.3136813.

[39] Michele Maestrini and Pierluigi Di Lizia. Guidance Strategy for Autonomous Inspection of Unknown Non-Cooperative Resident Space Objects. *Journal of Guidance, Control, and Dynamics*, 45(6):1126–1136, 2022. doi: 10.2514/1.G006126.

[40] Stefano Silvestrini and Michèle Lavagna. Deep Learning and Artificial Neural Networks for Spacecraft Dynamics, Navigation and Control. *Drones*, 6(10):270, 2022. doi: 10.3390/drones6100270.

[41] Mykel J. Kochenderfer, Tim A. Wheeler, and Kyle H. Wray. *Algorithms for Decision Making*. MIT Press, Cambridge, MA, 2022.

[42] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: Composable Transformations of Python+NumPy Programs. `http://github.com/jax-ml/jax`, 2018.

[43] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A Standard Interface for Reinforcement Learning Environments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025.

[44] Duncan Eddy and Mykel J. Kochenderfer. Brahe: A Modern Astrodynamics Library for Research and Engineering Applications, 2026. arXiv:2601.06452.

[45] W. H. Clohessy and R. S. Wiltshire. Terminal Guidance System for Satellite Rendezvous. *Journal of the Aerospace Sciences*, 27(9):653–658, 674, 1960. doi: 10.2514/8.8704.

[46] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms, 2017. arXiv:1707.06347.

[47] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021.

[48] Kyle Dunlap, Mark Mote, Kai Delsing, and Kerianne L. Hobbs. Run Time Assured Reinforcement Learning for Safe Satellite Docking. *Journal of Aerospace Information Systems*, 20(1):25–36, 2023. doi: 10.2514/1.I011126.

[49] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, 18(185): 1–52, 2018.