doi: 10.82124/CEAS-GNC-2026-084 CEAS-GNC-2026-084

EuroGNC Navigation Guidance Control CEAS Madrid, Spain May 5th-7th uc3m Universidad Carlos III de Madrid AIAA 2026

Orbita: A Novel Space Simulation Framework and its Application to Vision-Based Attitude Control in Close-Proximity Operations

Joaquin G. Lopez-Cepero Co-Founder, Hisperion Aerospace S.L., Seville, Spain. joaquin@hisperion.com

Jose Antonio Rebollo Co-Founder, Hisperion Aerospace S.L., Seville, Spain. jrebollo@hisperion.com

Javier Urrios Co-Founder, Hisperion Aerospace S.L., Seville, Spain. urrios@hisperion.com

ABSTRACT

This paper presents *Orbita*, a novel space simulation framework designed for the holistic modeling and validation of spacecraft dynamics, sensing, and control. Unlike existing approaches that decouple orbital dynamics from visual perception, *Orbita* integrates high-fidelity propagation, synthetic image generation, vision-based navigation, and closed-loop control into a unified software-in-the-loop environment. To demonstrate its capabilities, a Line-of-Sight tracking problem is solved using inputs from an onboard camera. We design, implement and validate a deterministic computer vision pipeline, a modified Multiplicative Extended Kalman Filter for relative attitude estimation, and a Lyapunov-based Backstepping controller for vision-guided spacecraft pointing. A representative case study involving two CubeSats in close-proximity operations is considered, assessing the effect of attitude perturbations on the overall system performance.

Keywords: Space simulation, Software-in-the-loop, Computer-vision, Attitude control, Close-proximity operations, Relative navigation

Nomenclature

B	=	Body axes
T	=	Target attitude axes
N	=	Inertial axes
$\mathbf{h}$	=	Angular momentum vector
$\mathbf{J}$	=	Inertia matrix
ω	=	Angular velocity
q	=	Quaternion
$\mathbf{b}$	=	Magnetic dipole
$\mathbf{I}_n$	=	Identity matrix
$\mathbf{0}_{n\times m}$	=	Zero matrix
$\tilde{\cdot}$	=	Noisy quantity

$\hat{\cdot}$ = Estimated quantity

1 Introduction

Autonomous close-proximity operations such as rendezvous, inspection, servicing, and formation flying are key enablers for future space missions. In these scenarios, the chaser must reliably determine the relative state of a passive target, often without cooperative markers, and maneuver to the desired configuration while satisfying strict constraints and rejecting disturbances. On-board cameras offer a lightweight, passive, and information-rich solution, but vision-based navigation requires robust estimation and control to perform adequately in realistic scenarios.

Validating such closed-loop architectures is challenging because it demands reproducing the coupling between orbital dynamics, imaging, estimation, and control. Prior work relies on simplified orbital models or hardware-in-the-loop (HIL) testbeds. For example, [1] used DLR's EPOS facility, where a robotic manipulator moved a mock-up spacecraft to emulate the final meters of rendezvous. Similar HIL setups with robotic arms and dark rooms were employed in [2], while [3] tested photogrammetric navigation on the SPHERES platform. These platforms are invaluable for hardware testing but remain constrained by scale, simplified dynamics, and the inability to reproduce realistic perturbations and lighting.

To overcome these limitations, software-in-the-loop (SIL) simulations with synthetic imagery have become essential. However, this domain has largely been split into two distinct approaches. On one hand, GNC-focused studies use orbital simulators to test control laws, but often rely on simplified sensor models or abstracted visual data [4, 5]. On the other hand, perception-focused studies leverage powerful rendering engines to create photorealistic datasets for developing advanced computer vision algorithms. A prime example is the work in [6], which used the Unity engine to generate high-fidelity images to train Neural Radiance Fields for 3D spacecraft reconstruction. In such studies, the camera poses are typically pre-scripted or sampled from a geometric distribution rather than being the output of a dynamic orbital simulation. Such approaches decouple perception from GNC, preventing end-to-end evaluation where spacecraft motion shapes the visual input.

This gap highlights the need for a fully integrated environment for end-to-end GNC systems validation that co-simulates dynamics, sensing, and actuation. For this aim, ***Orbita***, can be used: a novel simulation framework designed for the holistic modeling and validation of spacecraft dynamics and operations. While ***Orbita*** supports a wide range of applications, from interplanetary mission design to ground-space communications, this paper focuses on a representative case study: closed-loop vision-based Attitude Determination and Control System (ADCS) during close-proximity operations. By integrating high-fidelity dynamics, synthetic imaging, estimation, and control into a single feedback loop, ***Orbita*** enables realistic analysis of coupled perception and control.

The contributions of this work are as follows:

- The design of a vision-based navigation pipeline combining image processing, a modified multiplicative extended Kalman filter (MEKF), and a Lyapunov-based backstepping controller for close-proximity attitude control.
- The definition of a representative scenario that captures realistic orbital perturbations, lighting conditions, and sensing constraints.
- The evaluation of the closed-loop system through high-fidelity simulation, providing insights into the coupled performance of estimation and control in challenging operational conditions.

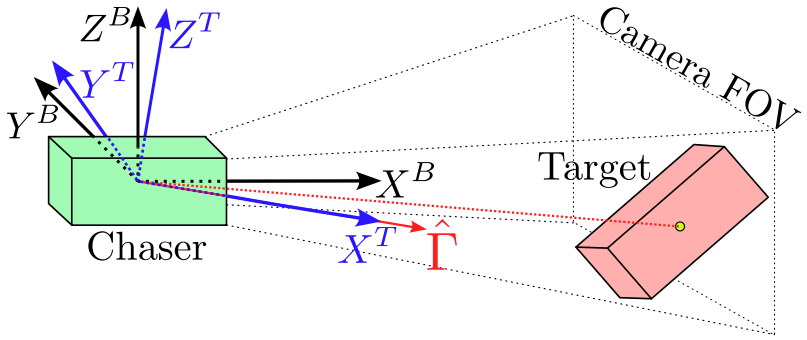

Fig. 1 Problem statement.

2 Problem statement

A chaser spacecraft is flying in close proximity to an uncooperative target spacecraft. The chaser is equipped with a navigation camera able to take images every $T_{s,c}$ seconds with a Field Of View (FOV) δ_c and resolution $m \times n$ (width $\times$ height) in pixels. The camera is aligned with the chaser's x axis, $\mathbf{X}^B$. The chaser also counts with a gyroscope that provides information on its rotational state with respect to the inertial frame N, and is equipped with a set of 3 reaction wheels aligned with each of the principal axes of the spacecraft, whose angular momentum is given in body axes B by

$$\mathbf{h}^B = \begin{pmatrix} h_x & h_y & h_z \end{pmatrix}^T. \tag{1}$$

The objective is to design a system that is able to solve the Line-of-Sight tracking problem by performing two tasks: (1) producing an estimate of the unit vector $\boldsymbol{\Gamma}$ from the chaser to the target, $\hat{\boldsymbol{\Gamma}}$; and (2) controlling the chaser's attitude to align its Line Of Sight (LOS) $\mathbf{X}^B$ with $\hat{\boldsymbol{\Gamma}}$. An illustration of the problem is provided in Fig. 1.

2.1 Frames of references

All frames of reference used in this work are referred to an origin O that coincides with the center of mass of the chaser at all time instants. The following frames of reference are used for this work:

1) N ($O\mathbf{X}^N\mathbf{Y}^N\mathbf{Z}^N$): This frame is aligned at all times with the inertial reference frame.
2) B ($O\mathbf{X}^B\mathbf{Y}^B\mathbf{Z}^B$): Represents the chaser's body fixed frame.
3) T ($O\mathbf{X}^T\mathbf{Y}^T\mathbf{Z}^T$): Represents the target reference frame that is being tracked. It is defined at each time instant based on the measured direction towards the target, given by the unitary vector $\hat{\boldsymbol{\Gamma}}$. Frame T results from a rotation of frame B by $\alpha = \arccos\left(\hat{\boldsymbol{\Gamma}} \cdot \mathbf{X}^B\right)$ radians around the axis

$$\mathbf{e} = \mathbf{X}^B \times \hat{\boldsymbol{\Gamma}}. \tag{2}$$

2.2 Quaternion kinematics

For this work, a passive representation of the attitude is used based on quaternions. In particular, the attitude of a given frame F with respect to the inertial frame N is given by the quaternion

$$q_{F/N} = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}, \tag{3}$$

where q_0 and $\mathbf{q}$ are the scalar and vector part of q, respectively.

The attitude error between two reference frames is given by the quaternion that rotates from one to the other. In particular, given the attitude of the target and body frames with respect to inertial $q_{T/N}$ and $q_{B/N}$, the error quaternion is defined as

$$q_e = q_{B/T} = q_{T/N}^* \otimes q_{B/N}, \tag{4}$$

where operator $(\cdot)^*$ represents the conjugate of the quaternion, and $\otimes$ denotes the quaternion product. Additionally, the angular velocity error between the B and the T frames in B axes is defined as

$$\boldsymbol{\omega}_e \doteq \boldsymbol{\omega}_{B/N}^B - \boldsymbol{\omega}_{T/N}^B. \tag{5}$$

Finally, the attitude kinematics of the error quaternion are given by [7]

$$\dot{q}_e = \frac{1}{2} \begin{bmatrix} -\mathbf{q}_e^T \\ q_{e,0}\mathbf{I}_3 + \mathbf{q}_e^\times \end{bmatrix} \boldsymbol{\omega}_e, \tag{6}$$

where $\mathbf{I}_3 \in \mathbb{R}^{3\times3}$ is the 3×3 identity matrix and $(\cdot)^\times$ produces the cross-product matrix of a vector

$$\mathbf{a}^\times \doteq \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} : \mathbf{a} \times \mathbf{b} = \mathbf{a}^\times \mathbf{b} \quad \mathbf{a}, \mathbf{b} \in \mathbb{R}^3. \tag{7}$$

3 Simulation environment and images generation

Currently under development by Hisperion Aerospace, ***Orbita*** is a commercial Software-as-a-Service (SaaS) platform that integrates modeling, simulation, image generation, vision, navigation, and control into a unified framework. This ecosystem is entirely accessible through a modern, intuitive graphical user interface (GUI) (see Fig. 2). Simulations begin with the user configuring fundamental parameters, including initial and final epochs, data point resolution, integration schemes, and numerical tolerances. Based on these inputs, ***Orbita*** automatically updates the scenario with the latest ephemerides and initializes optimized propagators. Targeted at a broad aerospace audience, the platform is designed to assist academic researchers, industry GNC engineers, mission designers, and satellite operators. ***Orbita*** streamlines simulation workflows by balancing accessibility with maximum flexibility, pairing its robust GUI with an API for extensive scripting. A prime example of this flexibility is the integration of custom Python-based controllers, as demonstrated in this work. What truly separates ***Orbita*** from its competitors is its intuitive, systems-engineering approach, allowing users to rapidly construct highly complex simulations by simply connecting high-level visual blocks (see Fig. 3).

This work employs a fifth-order Runge-Kutta Dormand-Prince integration scheme, with both absolute and relative tolerances set to 10^{-12}. The user can then add and configure a wide range of elements and properties to be simulated. For this study, two spacecraft are defined, one target and one chaser, each with distinct initial conditions tailored to the scenario. Both orbital and attitude dynamics can be configured to achieve the desired level of realism.

The orbital perturbations considered in this work include a gravity field expansion up to degree and order 80, based on the GGM02C model [8], along with third-body effects from the Sun, Moon, and Jupiter, as well as atmospheric drag modeled using the U.S. Standard Atmosphere [9]. The attitude perturbations include gravity-gradient torque and magnetic torque, the latter modeled with a spherical harmonic expansion up to order 10 using World Magnetic Model (WMM) coefficients [10].

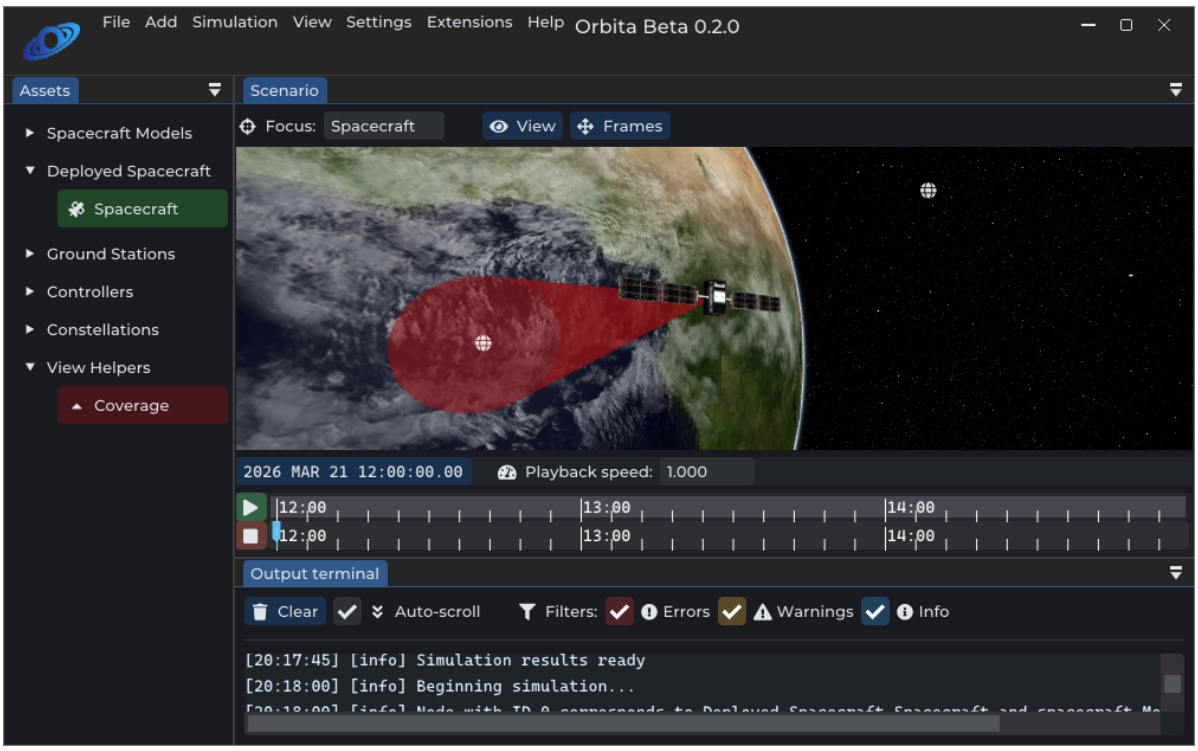

Fig. 2 ***Orbita*** **GUI**

Because the core of ***Orbita*** is implemented entirely in C++ and extensively optimized for performance-critical tasks, such as numerical integration, spherical harmonic expansion, and ephemeris processing, the application is able to simulate several days of the scenario under study, a computationally expensive process, in just a few seconds. For instance, a 3 days simulation[1] takes 10 seconds to run on a basic laptop[2]. This level of performance is typically achievable only with highly specialized code. The actual runtime depends on factors such as available hardware, scenario complexity, and chosen initial conditions.

Spacecraft can be equipped with a variety of sensors and actuators. In this work, the chaser has access to angular velocity measurements and an optical camera. Additionally, the chaser is provided with three reaction wheels placed along orthogonal axes, chosen as the principal axes of inertia for simplicity. By doing so, ***Orbita*** generates the needed environment to implement closed loop algorithms through the *Block Editor*, as depicted in Fig. 3. The controller is built as a custom Python script, which the application naturally integrates within the dynamical model. Images with the configured camera are generated at the defined sampling rate and resolution by efficiently rendering the scene at the propagated state. For this purpose, ***Orbita***'s custom graphic engine combines rasterization techniques with ray tracing in strategic areas. This optimized architecture maximizes performance and frame rate while maintaining a level of realism, allowing for online integration in conventional hardware [11].

The camera is modeled as a projective pinhole system with intrinsic parameters defined by an image resolution equal to the physical detector pixel array, a specified field of view (or focal length), and a principal point at the image center. The rendered image buffer matches the detector resolution exactly, establishing a one-to-one correspondence between frame buffer and sensor pixels, but with additional subpixel resampling. Scene radiance is computed using a hybrid rasterization and ray-tracing pipeline, where rasterization handles primary visibility and standard shading, and ray tracing accounts for secondary effects. The renderer outputs per-pixel RGB intensities as 8-bit unsigned integers, representing discretized radiometric intensity after shading and illumination. For fixed scene and lighting conditions,

[1]Considering all the described tolerances and perturbations, for the simulation conditions in section 5, no actuation.
[2]The used processor is an 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GH.

the pipeline is deterministic; however, the framework allows optional additive or signal-dependent noise injection to approximate realistic sensor behavior.

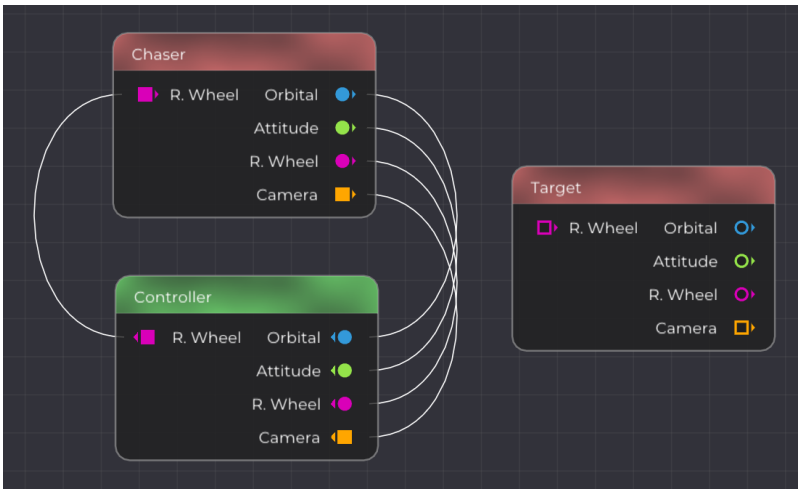

Fig. 3 Block editor with control loop in *Orbita*.

4 System architecture and design

The chaser spacecraft's ADCS operates on a feedback loop principle consisting of three main stages, namely, vision, navigation, and control system, as depicted in Fig. 4. In this section, the implementation of each of the three systems is described and justified.

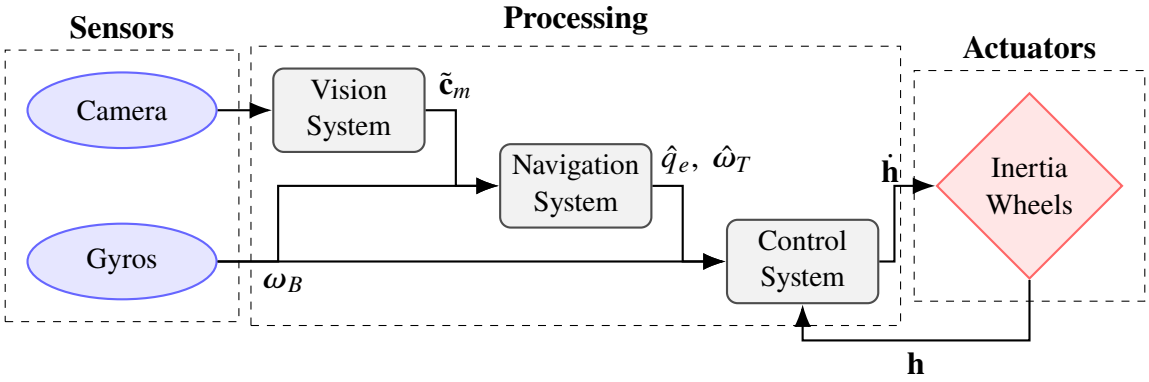

Fig. 4 System architecture with annotated signal flows.

4.1 Vision System

Vision-based spacecraft-relative navigation spans classical feature pipelines, model matching, and learning-based methods. Classical pipelines detect and track keypoints, solve Perspective-n-Point with robust fitting, and fuse pose estimates with inertial data [3]. When target geometry is known, model-based approaches match CAD projections or silhouettes to edges, a standard strategy for rendezvous and docking [12]. Recent work leverages deep learning for segmentation, learned keypoints, and direct 6-DoF pose regression, improving robustness to illumination changes and occlusion [13]. Hybrid methods combine deep perception with classical solvers for interpretability, bounded runtime, and verifiability [14].

In this study, deep learning and black-box detectors are avoided in favor of a simpler deterministic image-mask and morphology-based pipeline, prioritizing determinism, interpretability, and bounded complexity, properties crucial for embedded flight systems in practical conditions. While neural methods

excel in complex scenarios, the focus here is system integration and validation, for which a morphological Computer Vision (CV) pipeline offers both simplicity and a practical demonstration of ***Orbita*** 's capabilities. That said, replacing the deterministic centroid pipeline with more robust alternatives could improve resilience to noise, outliers, or complex input distributions, potentially enhancing the accuracy and reliability of downstream processing. Probabilistic or adaptive centroid estimation methods, for instance, can reduce sensitivity to measurement errors or varying input conditions. Importantly, integrating such alternatives into ***Orbita*** requires minimal effort: the propagator interface is algorithm-agnostic and only requires a Python script to accept inputs and produce outputs, so adopting more sophisticated centroiding methods does not necessitate changes to the simulator itself, making their integration straightforward and low-overhead. Consequently, researchers can seamlessly plug their custom algorithms directly into the platform for rapid testing.

Relative target position within the camera FOV is estimated by isolating the target and computing its centroid in each frame. The algorithm applies a sequence of masking and morphological operations, which are efficient and well suited for real-time onboard use [15, 16]. The input image is converted from RGB to HSV to enable filtering by saturation and brightness. Low-saturation, bright regions are retained to suppress background noise and preserve the target, typically a desaturated (gray) region. The result is converted to grayscale, thresholded, and binarized to identify candidate pixels.

Morphological operations then refine the mask: opening (erosion + dilation) removes small artifacts and stars, closing (dilation + erosion) fills gaps, and a final dilation ensures continuity. The binary mask is analyzed with connected-component labeling. Each region is described by bounding box, area, and centroid; the largest is taken as the target. Its centroid $\tilde{\mathbf{c}} = (\tilde{c}_x, \tilde{c}_y)$ provides the target's image coordinates. To distinguish between data stages, we employ $\tilde{\cdot}$ to represent magnitudes derived from raw sensor data, and $\hat{\cdot}$ to denote the filtered stochastic optimal estimate of the system state. Figure 5 shows the full pipeline.

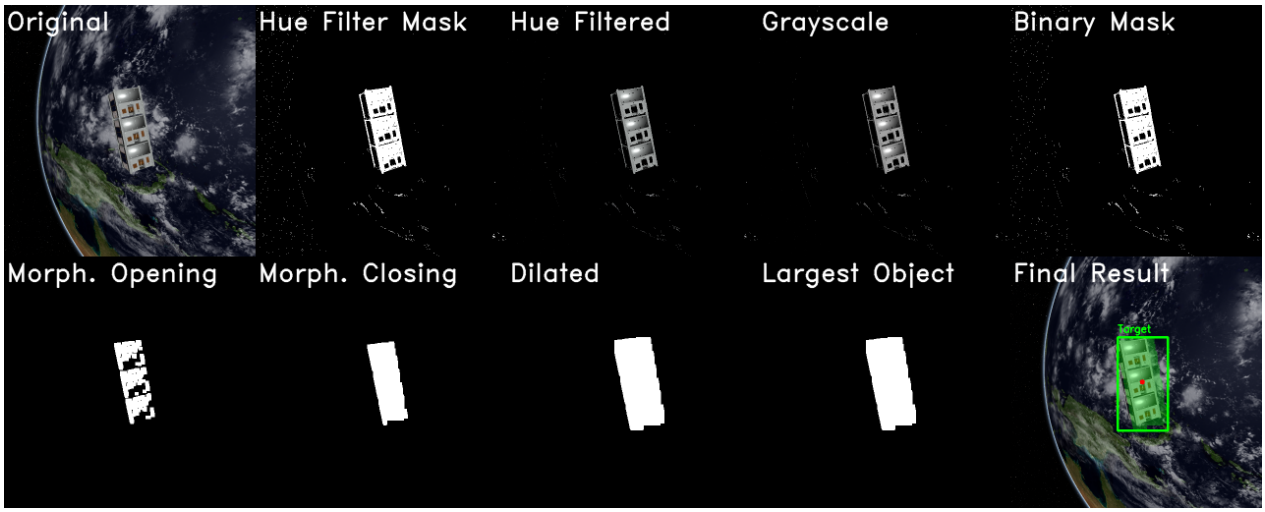

Fig. 5 Image processing pipeline.

To stabilize the centroid estimate and suppress frame-to-frame noise, an exponential moving average filter is applied:

$$\tilde{\mathbf{c}}_m(t + \delta t) = (1 - \alpha)\tilde{\mathbf{c}}_m(t) + \alpha\tilde{\mathbf{c}}(t + \delta t), \tag{8}$$

where $\tilde{\mathbf{c}}$ are raw centroid measurements, $\tilde{\mathbf{c}}_m$ the result of the exponential moving average, and α the update rate. The centroid is measured in pixels, depending on camera resolution and FOV.

4.2 Navigation System

Spacecraft attitude estimation is commonly addressed using variants of the Extended Kalman Filter (EKF), with the Multiplicative EKF (MEKF) being the most widely adopted. The MEKF represents

attitude error as a small quaternion rotation about a nominal quaternion, thereby preserving the unit-norm constraint and avoiding singularities associated with minimal parameterizations [17]. This filter remains standard practice in operational ADCS implementations, including small satellite missions [18]. Extensions of this approach include the Dual Quaternion MEKF (DQ-MEKF), which jointly estimates position and orientation by leveraging the algebra of unit dual quaternions [19].

A closed-loop relative navigation scheme requires modifications to the MEKF to handle a moving, non-inertial target frame and to estimate relative angular velocity for feedback control. The adopted filter formulation retains the nominal quaternion plus small-error representation but incorporates vision-based measurements and relative motion models to produce consistent attitude and velocity estimates. This design ensures tight integration between perception and navigation layers, providing the control system with reliable state information for guidance and tracking.

The first step of the estimator is to determine the relative attitude of the chaser frame with respect to the target frame, q_e, as given by the averaged centroid $\hat{\mathbf{c}}$. A vector pointing from the camera to the target can be computed as

$$\tilde{\mathbf{v}}_T^B = \begin{pmatrix} 1 \\ -\dfrac{\tilde{c}_{m,x} - w/2}{w/2} \tan\frac{FOV}{2} \\ -\dfrac{\tilde{c}_{m,y} - h/2}{h/2} \tan\frac{FOV}{2} \end{pmatrix}, \tag{9}$$

where w and h are the sensor width and height in pixels, while FOV denotes the camera's total field of view in radians. This vector is then normalized to obtain an estimate of $\mathbf{\Gamma}$, $\tilde{\mathbf{\Gamma}}_T^B = \tilde{\mathbf{v}}_T^B / \|\tilde{\mathbf{v}}_T^B\|$. Defining the camera front vector as $\mathbf{X}^B = (1, 0, 0)^T$, the relative rotation of the chaser with respect to the target is given by

$$\tilde{\theta} = -\arccos\left(\mathbf{X}^B \cdot \tilde{\mathbf{\Gamma}}_T^B\right), \tag{10}$$

$$\tilde{\mathbf{e}} = \mathbf{X}^B \times \tilde{\mathbf{\Gamma}}_T^B, \tag{11}$$

$$\tilde{q}_e = \begin{pmatrix} \cos(\tilde{\theta}/2) \\ \sin(\tilde{\theta}/2)\tilde{\mathbf{e}} \end{pmatrix}. \tag{12}$$

The measured relative attitude $\tilde{q}_e$ is the input of the Navigation System. For this purpose, several assumptions are considered. Firstly, the target frame is assumed to move at a constant angular velocity in the inertial frame, that is,

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\omega}_{T/N}^N \approx \mathbf{0}. \tag{13}$$

For simplicity of notation, let the target angular velocity be $\boldsymbol{\omega}_T \doteq \boldsymbol{\omega}_{T/N}^B$ and the body angular velocity be $\boldsymbol{\omega}_B \doteq \boldsymbol{\omega}_{B/N}^B$. The time evolution of $\boldsymbol{\omega}_T$, which is written in the observer axes, is given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{\omega}_T = -\boldsymbol{\omega}_B \times \boldsymbol{\omega}_T. \tag{14}$$

In second place, measurements of spacecraft angular velocity, $\boldsymbol{\omega}_B$, obtained from the on-board gyroscopes, are assumed very accurate. Angular velocities composition allows defining the error angular velocity as

$$\boldsymbol{\omega}_e \doteq \boldsymbol{\omega}_{B/T}^B = \boldsymbol{\omega}_B - \boldsymbol{\omega}_T, \tag{15}$$

where $\boldsymbol{\omega}_T$ is the only source of uncertainty. Finally, measurements of the error attitude q_e are assumed to be affected by a multiplicative error, namely,

$$\tilde{q}_e = q_e \otimes \begin{pmatrix} 1 \\ \boldsymbol{\eta}/2 \end{pmatrix}, \tag{16}$$

where $\boldsymbol{\eta}$ is sampled from a zero-mean multivariate normal distribution, $\boldsymbol{\eta} \sim \mathcal{N}(0, \mathbf{R})$.

Using the standard filtering notation, in what follows, for any variable $\mathbf{f}$, the associated estimate is defined as $\hat{\mathbf{f}}$. Additionally, the superscript $\hat{\mathbf{f}}^-$ denotes a-priori estimation, that is, before the filter, while $\hat{\mathbf{f}}^+$ is the a-posteriori estimation, after the filter. The output of the Navigation System is the estimated pair $\hat{q}_e$, $\hat{\boldsymbol{\omega}}_T$. As $\boldsymbol{\omega}_B$ is assumed to be accurately known, the error angular velocity is also directly estimated, as given by Eq. (15).

The estimator is designed as a modified Multiplicative Extended Kalman Filter (MEKF) with both attitude and angular velocity estimation. For this purpose, the estimated error attitude quaternion $\hat{q}_e$ is represented by a multiplicative deviation from the true value q_e, namely,

$$\hat{q}_e = q_e \otimes \begin{pmatrix} 1 \\ \hat{\mathbf{a}}/2 \end{pmatrix}. \tag{17}$$

Thus, the error quaternion dynamics in Eq. (6) is linearized by defining the error vector as in Eq. (17), yielding

$$\frac{\mathrm{d}}{\mathrm{d}t}\hat{\mathbf{a}} = -\boldsymbol{\omega}_B \times \hat{\mathbf{a}} - \hat{\boldsymbol{\omega}}_T. \tag{18}$$

Let the estimator state be

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{a}} \\ \hat{\boldsymbol{\omega}}_T \end{pmatrix}, \tag{19}$$

and the associated covariance matrix $\mathbf{P}$. For the propagation of $\mathbf{P}$, as obtained from the local dynamics described in Eqs. (14) and (18), let the Jacobian of the state variables be

$$\mathbf{F} \doteq \begin{bmatrix} -\boldsymbol{\omega}_B^{\times} & -\mathbf{I}_{3\times3} \\ \mathbf{0}_{3\times3} & -\boldsymbol{\omega}_B^{\times} \end{bmatrix}. \tag{20}$$

The Jacobian matrix couples the attitude and angular velocity estimation through the upper diagonal region, which allows estimating $\boldsymbol{\omega}_T$ without direct angular velocity measurements. This non-standard modification of the conventional MEKF is a powerful tool for the improvement of the controller behavior, while providing a robust and computationally efficient implementation. In this scenario, only relative attitude is measured. Thus, the measurement matrix is defined as

$$\mathbf{H} \doteq \begin{bmatrix} \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}. \tag{21}$$

This way, the measurements $\mathbf{z}$ are given by twice the vector part of the error quaternion,

$$\mathbf{z} \doteq 2\tilde{\mathbf{q}}_e. \tag{22}$$

The estimations of angular velocity must then be completely inferred by the filter. For the sake of implementability, assuming step-wise constant values of $\boldsymbol{\omega}_B$ and $\boldsymbol{\omega}_T$, the discrete-time dynamics of the

error variables can be written exactly as

$$\hat{q}_e^-(t+\delta t) = \frac{1}{2}\hat{q}_e^+(t) \otimes \begin{pmatrix} \cos\left(\|\hat{\omega}_e^+(t)\|\delta t/2\right) \\ \sin\left(\|\hat{\omega}_e^+(t)\|\delta t/2\right)\hat{\omega}_e^+(t)/\|\hat{\omega}_e^+(t)\| \end{pmatrix}, \tag{23}$$

$$\hat{\omega}_T^-(t+\delta t) = \hat{\omega}_T^+(t) - \omega_B(t) \times \hat{\omega}_T^+(t)\delta t, \tag{24}$$

$$\mathbf{P}^-(t+\delta t) = \mathbf{\Phi}(t)\mathbf{P}^+(t)\mathbf{\Phi}(t)^T + \mathbf{Q}_d, \tag{25}$$

where $\mathbf{Q}_d$ is the propagation noise covariance and for the State Transition Matrix (STM) defined as

$$\mathbf{\Phi}(t) = \exp\left(\mathbf{F}(t)\delta t\right). \tag{26}$$

When measurements are available, a Kalman filter is applied. Let the Kalman gain be

$$\mathbf{K} = \mathbf{P}^-\mathbf{H}^T(\mathbf{H}\mathbf{P}^-\mathbf{H}^T + \mathbf{R})^{-1}, \tag{27}$$

where $\mathbf{R}$ is the measurement noise covariance. This allows computing the innovation vector, given by

$$\delta\hat{\mathbf{x}}(t+\delta t) = \mathbf{K}\left(\mathbf{z}(t+\delta t) - \mathbf{H}\hat{\mathbf{x}}^-(t+\delta t)\right). \tag{28}$$

The quaternion part of the innovation vector, $\delta\hat{\mathbf{a}}(t+\delta t)$, is applied to the estimated error quaternion as

$$\hat{q}_e^+(t+\delta t) = \hat{q}_e^-(t+\delta t) \otimes \begin{pmatrix} 1 \\ \delta\hat{\mathbf{a}}(t+\delta t)/2 \end{pmatrix}, \tag{29}$$

and the a-posteriori target angular velocity is given by

$$\hat{\omega}_T^+(t+\delta t) = \hat{\omega}_T^-(t+\delta t) + \delta\hat{\omega}(t+\delta t). \tag{30}$$

A convenient final adjustment is to force $\mathbf{X}^B \cdot \hat{\omega}_T = 0$, that is, no relative angular velocity in the first component. This guarantees that the controller will stabilize the camera front axis, for which the dynamics of the target are not observable, avoiding any potential tumbling.

4.3 Control System

Classical spacecraft attitude control is often implemented using PD, PID, or quaternion-feedback laws, providing simple and reliable stabilization [20, 21]. Various robust and nonlinear extensions, such as sliding-mode, adaptive, and backstepping controllers, have been proposed to improve disturbance rejection and accommodate modeling errors [22–24]. In vision-guided relative operations, these requirements become particularly stringent, as the controller must cope with higher estimation uncertainty and limited onboard computational resources.

The controller employed in this study is based on a Lyapunov-based backstepping design that guarantees global asymptotic stability under nominal conditions. By relying solely on the navigation system's relative attitude and angular velocity estimates, the control scheme remains lightweight and implementable in real time. The backstepping framework allows systematic handling of the attitude dynamics while mitigating the effect of estimator noise, providing a suitable solution for vision-guided relative maneuvers in constrained spacecraft missions.

The chaser's rigid body dynamics, written in the body axes B, is given by

$$\mathbf{J}\dot{\omega}_{B/N}^B + \left(\omega_{B/N}^B\right)^\times \mathbf{J}\omega_{B/N}^B = \mathbf{u} + \mathbf{M}, \tag{31}$$

where $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the inertia tensor of the spacecraft, $\mathbf{u} \in \mathbb{R}^3$ is the control input to the system and $\mathbf{M} \in \mathbb{R}^3$ are the disturbing torques. For the controller design, however, disturbing torques are ignored. Note that no assumption is made on the control mechanism of the spacecraft; the controller is designed for an arbitrary set of actuators, and then particularized for the case of a set of reaction wheels.

The target equilibria[3] are given by $q_e \to (\pm1, 0, 0, 0)^T$ and $\boldsymbol{\omega}_e \to (0, 0, 0)^T$, where q_e and $\boldsymbol{\omega}_e$ are given by Eqs. (4) and (5) respectively. The attitude kinematics are expressed in Eq. (6), while the angular velocity error dynamics can be computed by differentiating Eq. (5) with respect to time, pre-multiplying by $\mathbf{J}$, and substituting Eq. (31),

$$\mathbf{J}\dot{\boldsymbol{\omega}}_e = \mathbf{J}\dot{\omega}^B_{B/N} - \mathbf{J}\dot{\omega}^B_{T/N} = \left(\mathbf{u} - \left(\boldsymbol{\omega}^B_{B/N}\right)^\times \mathbf{J}\boldsymbol{\omega}^B_{B/N}\right) - \mathbf{J}\dot{\omega}^B_{T/N}. \tag{32}$$

Following a similar approach to that described in [7] for the backstepping controller design, let the set of states $\mathbf{x}_1 \in \mathbb{R}^4, \mathbf{x}_2 \in \mathbb{R}^3$ be

$$\mathbf{x}_1 = \begin{pmatrix} 1 - |q_{e,0}| \\ \mathbf{q}_e \end{pmatrix}, \quad \mathbf{x}_2 = \boldsymbol{\omega}_e, \tag{33}$$

so that

$$\{\mathbf{x}_1 = \mathbf{0}, \quad \mathbf{x}_2 = \mathbf{0}\} \leftrightarrow \{q_e = (\pm1, 0, 0, 0)^T, \quad \omega_e = \mathbf{0}\}. \tag{34}$$

The dynamics for the newly defined states are

$$\dot{\mathbf{x}}_1 = \frac{1}{2}\begin{pmatrix} \mathrm{sgn}(q_{e,0})\mathbf{q}_e^T \\ q_{e,0}\mathbf{I}_3 + \mathbf{q}_e^\times \end{pmatrix}\boldsymbol{\omega}_e = \frac{1}{2}\mathbf{Q}^T\mathbf{x}_2, \tag{35}$$

$$\mathbf{J}\dot{\mathbf{x}}_2 = \left(\mathbf{u} - \left(\boldsymbol{\omega}^B_{B/N}\right)^\times \mathbf{J}\boldsymbol{\omega}^B_{B/N}\right) - \mathbf{J}\dot{\omega}^B_{T/N}, \tag{36}$$

where $\mathrm{sgn}(x) = 1$ if $x \geq 0, -1$ otherwise. A backstepping controller design aims to stabilize the dynamical system following a recursive, Lyapunov-based non-linear control methodology. For its implementation, one needs to express the system in strict-feedback form [25], which is the case for the system defined by Eqs. (35) and (36). To design the controller, Subsystem (35) is controlled in the first place by taking $\mathbf{x}_2$ as the virtual control input. This is satisfied by finding a control function $\boldsymbol{\Lambda}(\mathbf{x}_1) : \mathbb{R}^4 \to \mathbb{R}^3$ that makes the origin on Subsystem (35) asymptotically stable. Let a function of the type $\boldsymbol{\Lambda}(\mathbf{x}_1) = -\mathbf{K}_1\mathbf{Q}\mathbf{x}_1$, where $\mathbf{K}_1 \in \mathbb{R}^{3\times3}$ is a positive definite matrix. A standard candidate Lyapunov function $V_1(\mathbf{x}_1) : \mathbb{R}^4 \to \mathbb{R}$ is

$$V_1(\mathbf{x}_1) = \frac{1}{2}\mathbf{x}_1^T\mathbf{x}_1 > 0, \tag{37}$$

such that

$$\dot{V}_1(\mathbf{x}_1) = \mathbf{x}_1^T\dot{\mathbf{x}}_1 = -\frac{1}{2}\mathbf{x}_1^T\mathbf{Q}^T\mathbf{K}_1\mathbf{Q}\mathbf{x}_1, \tag{38}$$

where Eq. (35) is used, and $\mathbf{x}_2$ is replaced by the virtual control input $\boldsymbol{\Lambda}(\mathbf{x}_1)$. It can be proven that $\mathbf{Q}\mathbf{x}_1 = \mathrm{sgn}(q_{e,0})\mathbf{q}_e$, leading to

$$\dot{V}_1(\mathbf{x}_1) = -\frac{1}{2}\left(\mathbf{Q}\mathbf{x}_1\right)^T\mathbf{K}_1\mathbf{Q}\mathbf{x}_1 = -\frac{1}{2}\mathbf{q}_e^T\mathbf{K}_1\mathbf{q}_e < 0, \quad \forall\mathbf{x}_1 \in \mathbb{R}^4. \tag{39}$$

As a consequence, the virtual control input $\boldsymbol{\Lambda}(\mathbf{x}_1)$ for $\mathbf{x}_2$ makes the origin asymptotically stable for Subsystem (35). At this point, the system is augmented by defining the error state of the virtual control

[3] Note that two valid representations of the zero-error quaternion can be considered.

as $\mathbf{x}_3 = \mathbf{x}_2 - \mathbf{\Lambda}(\mathbf{x}_1)$, whose dynamics are given by

$$\dot{\mathbf{x}}_3 = \dot{\mathbf{x}}_2 - \dot{\mathbf{\Lambda}} = \boldsymbol{\rho}, \tag{40}$$

where $\boldsymbol{\rho}$ is the control input for $\mathbf{x}_3$. The compound Lyapunov function $V_c(\mathbf{x}_1, \mathbf{x}_3)$, similar to [7], is

$$V_c(\mathbf{x}_1, \mathbf{x}_3) = V_1(\mathbf{x}_1) + \frac{1}{2}\mathbf{x}_3^T \mathbf{J} \mathbf{x}_3 > 0, \tag{41}$$

and its time derivative is

$$\dot{V}_c(\mathbf{x}_1, \mathbf{x}_3) = \dot{V}_1(\mathbf{x}_1) + \mathbf{x}_3^T \mathbf{J} \dot{\mathbf{x}}_3 = \dot{V}_1(\mathbf{x}_1) + \mathbf{x}_3^T \mathbf{J} \boldsymbol{\rho}. \tag{42}$$

Then, if one selects $\boldsymbol{\rho} = -\mathbf{J}^{-1}\mathbf{K}_3\mathbf{x}_3$, with $\mathbf{K}_3 \in \mathbb{R}^{3\times3}$ being a positive definite matrix, V_c becomes

$$\dot{V}_c(\mathbf{x}_1, \mathbf{x}_3) = \dot{V}_1(\mathbf{x}_1) - \mathbf{x}_3^T \mathbf{K}_3 \mathbf{x}_3 < 0 \quad \forall \mathbf{x}_1 \in \mathbb{R}^4, \mathbf{x}_3 \in \mathbb{R}^3. \tag{43}$$

From Eq. (40), it is possible to obtain

$$\mathbf{J}\dot{\mathbf{x}}_2 = \mathbf{J}\dot{\boldsymbol{\omega}}_e = \mathbf{J}\dot{\mathbf{x}}_3 + \mathbf{J}\dot{\mathbf{\Lambda}}, \tag{44}$$

which, introduced in Eq. (36), leads to

$$\mathbf{u} = \mathbf{J}\dot{\mathbf{x}}_3 + \mathbf{J}\dot{\mathbf{\Lambda}} + \left(\boldsymbol{\omega}_{B/N}^B\right)^\times \mathbf{J}\boldsymbol{\omega}_{B/N}^B + \mathbf{J}\dot{\boldsymbol{\omega}}_{T/N}^B. \tag{45}$$

Taking into account that $\mathbf{J}\dot{\mathbf{x}}_3 = -\mathbf{K}_3\mathbf{x}_3$ and $\mathbf{x}_3 = \boldsymbol{\omega}_e + \mathbf{K}_1\mathbf{Q}\mathbf{x}_1$, one obtains

$$\mathbf{u} = -\mathbf{K}_3\left(\boldsymbol{\omega}_e + \mathbf{K}_1\mathbf{Q}\mathbf{x}_1\right) + \mathbf{J}\dot{\mathbf{\Lambda}} + \left(\boldsymbol{\omega}_{B/N}^B\right)^\times \mathbf{J}\boldsymbol{\omega}_{B/N}^B + \mathbf{J}\dot{\boldsymbol{\omega}}_{T/N}^B. \tag{46}$$

To compute the time derivative $\dot{\mathbf{\Lambda}}$ recall that $\mathbf{\Lambda} = -\mathbf{K}_1\mathbf{Q}\mathbf{x}_1$, and that $\mathbf{Q}\mathbf{x}_1 = \mathrm{sgn}(q_{e,0})\mathbf{q}_e$, leading to

$$\dot{\mathbf{\Lambda}} = -\mathbf{K}_1\left(\frac{\partial \mathrm{sgn}(q_{e,0})}{\partial q_{e,0}}\dot{q}_{e,0}\mathbf{q}_e + \mathrm{sgn}(q_{e,0})\dot{\mathbf{q}}_e\right). \tag{47}$$

Note that the first term on the right-hand side is zero, except for $q_{e,0} = 0$, for which it becomes singular. Given the controller design, initial states with $q_{e,0} > 0$ are stabilized towards $q = (1, 0, 0, 0)$ whereas points satisfying $q_{e,0} < 0$ will converge to $q = (-1, 0, 0, 0)$. As discussed next, the removable singularity in $q_{e,0} = 0$ can be ignored if one defines $\mathrm{sgn}(0) \doteq 1$. Thus,

$$\dot{\mathbf{\Lambda}} = -\mathbf{K}_1\mathrm{sgn}(q_{e,0})\dot{\mathbf{q}}_e = -\frac{1}{2}\mathbf{K}_1\mathrm{sgn}(q_{e,0})\left(q_{e,0}\boldsymbol{\omega}_e + \mathbf{q}_e^\times\right)\boldsymbol{\omega}_e. \tag{48}$$

Given the system provided by Eqs. (35), (36) and (40), it is clear that the origin $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = (\mathbf{0}, \mathbf{0}, \mathbf{0})$ is an equilibrium point. Moreover, the proposed Lyapunov functions satisfy $V_1(\mathbf{x}_1) > 0$ and $V_c(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) > 0$, and its time derivatives are such that

$$\dot{V}_1(\mathbf{x}_1) < 0 \quad \forall \mathbf{x}_1 \neq \mathbf{0}, \tag{49}$$

$$\dot{V}_c(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) < 0 \quad \forall (\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \neq (\mathbf{0}, \mathbf{0}, \mathbf{0}). \tag{50}$$

By virtue of the Lyapunov theorem, the equilibrium $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) = (\mathbf{0}, \mathbf{0}, \mathbf{0})$ is globally asymptotically stable. Substituting $q_{e,0} = 0$ in the controlled system dynamics, it seems that the system can be placed in an unstable equilibrium. However, provided that sgn(0) is defined as 1, the symmetry is broken, and

the plant is correctly stabilized towards equilibrium, producing a global asymptotic stability. Moreover, given the linear nature of $\mathbf{\Lambda}(\boldsymbol{x}_1)$, its evaluation and that of $\dot{\mathbf{\Lambda}}$ are more computationally efficient than the controller proposed in [7]. Considering that the controller may operate at a higher frequency than the CV module, and that onboard computational power is limited, this efficiency enhances overall performance, which would be especially pronounced on smaller platforms, such as CubeSats.

Note that no assumptions have been made for the control input $\mathbf{u}$, as it is treated as a generic torque around each of the axes. For this work, a set of reaction wheels are used to control the attitude of the spacecraft. In particular, the input for the control system of the reaction wheels is taken as the time derivative of its angular momentum, $\dot{\mathbf{h}}$, which can be expressed as a function of $\mathbf{u}$ as

$$\dot{\mathbf{h}} = -\mathbf{u} - \left(\boldsymbol{\omega}_{B/N}^{B}\right)^{\times} \mathbf{h} = \mathbf{K}_3 \left(\boldsymbol{\omega}_e + \mathbf{K}_1 \mathbf{Q} \mathbf{x}_1\right) - \mathbf{J}\dot{\mathbf{\Lambda}} - \left(\boldsymbol{\omega}_{B/N}^{B}\right)^{\times} \mathbf{J}\boldsymbol{\omega}_{B/N}^{B} - \mathbf{J}\boldsymbol{\omega}_{T/N}^{B} - \left(\boldsymbol{\omega}_{B/N}^{B}\right)^{\times} \mathbf{h}. \tag{51}$$

It is worth mentioning that a measurement of the relative error q_e will only be available if the target is within the camera FOV, given by δ_c. Since $\delta_c < 180°$, then $q_{e,0} > 0$. As a consequence, the general expressions obtained for the controller in this section may be simplified by removing the $\text{sgn}(q_{e,0})$ function (as it is always equal to 1), as well as the absolute value in the definition of $\mathbf{x}_1$. Whenever a value of q_e is not available, $\mathbf{u}$ is set to $\mathbf{0}$ until the target is detected again. More sophisticated strategies involve executing a pre-programmed search pattern, such as a conical scan or an expanding spiral, rotating the spacecraft about its principal axes to systematically sweep the celestial sphere until the target is detected again.

To compare the performance of this new backstepping controller, simulations are also conducted with a PD attitude controller, considered the baseline. Its control law is:

$$\mathbf{u} = -\mathbf{K}_p \mathbf{q}_e - \mathbf{K}_\mathbf{d} \boldsymbol{\omega}_e + \left(\boldsymbol{\omega}_{B/N}^{B}\right)^{\times} \mathbf{J}\boldsymbol{\omega}_{B/N}^{B} + \mathbf{J}\dot{\boldsymbol{\omega}}_{T/N}^{B}. \tag{52}$$

Note that it has the usual proportional and derivative linear terms, but also the same feed-forward compensation used for the backstepping controller.

5 Simulation results

In order to validate the developed control architecture, a realistic scenario is simulated. In particular, two CubeSats fly in close proximity within an inclined circular LEO orbit. In this situation, the Earth can fill the complete LOS of the chaser's camera, potentially affecting the vision system. Furthermore, eclipses and illumination angles alter significantly the visibility of the target, increasing the overall complexity.

5.1 Spacecraft properties

For testing and validating the control, two 3U CubeSats are taken as chaser and target, respectively. Both have dimensions of $10 \times 10 \times 34$ cm and a total weight of 4 kg. The inertia matrix is given by

$$\mathbf{J} = \begin{bmatrix} 0.0419 & 0 & 0 \\ 0 & 0.0419 & 0 \\ 0 & 0 & 0.0067 \end{bmatrix} \text{kg m}^2. \tag{53}$$

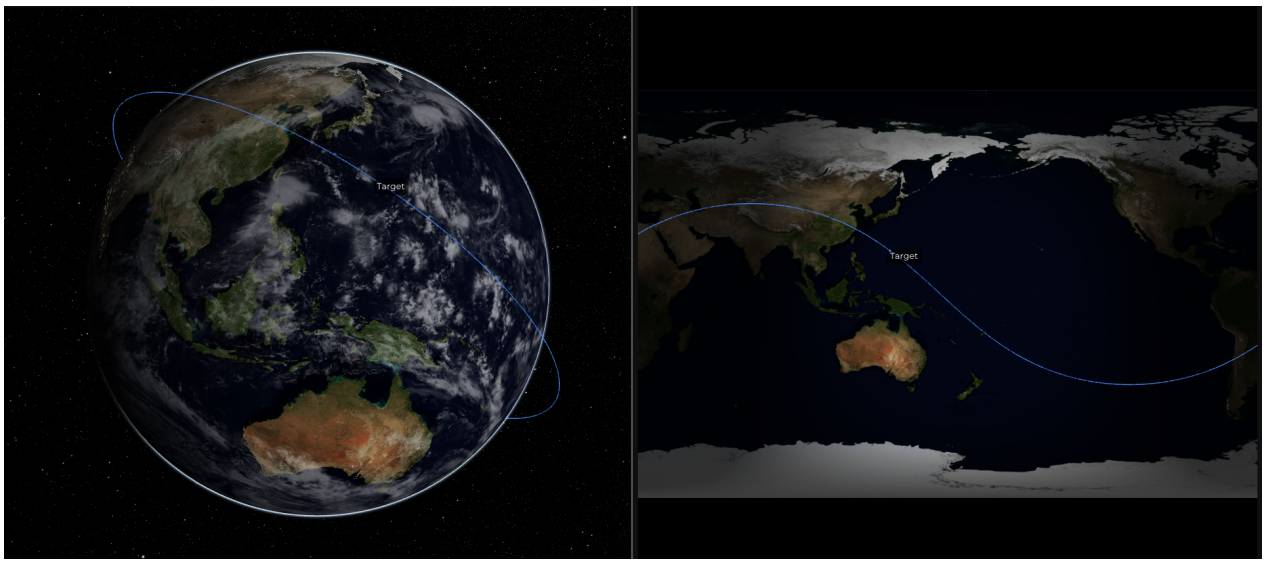

Fig. 6 Target spacecraft orbit and ground track in *Orbita*.

Additionally, the spacecraft are characterized by a magnetic dipole due to onboard electronic components in body axes of

$$\mathbf{b}^B = \begin{pmatrix} 0 \\ 0 \\ 10^{-4} \end{pmatrix} \text{A m}^2. \tag{54}$$

The chaser is equipped with a camera oriented towards the $\mathbf{X}^B$ direction, with the upwards direction towards $\mathbf{Z}^B$.

5.2 Scenario setup within *Orbita*

Simulations are performed starting on 1 October 2025 at 00:00:00 UTC, with a total duration of 1.5 hours. The initial state of the target is computed so that the satellite is within a circular orbit at a height of 800 km, with an inclination of 40°. ***Orbita*** allows to directly provide initial coordinates in orbital elements for simplicity. Note that, as the propagation considers perturbations, the orbit does not remain Keplerian. The resulting trajectory is displayed in Fig. 6, as seen in ***Orbita***.

The chaser spacecraft is initialized in a bounded Hill-Clohessy-Wiltshire (see, e.g [26]) solution around the target, with an initial offset of 8 m towards the Earth and 2 m along-track. In this way, both spacecraft stay in close proximity for the duration of the simulation. Illumination conditions and camera LOS limits are what determine the visibility of the target.

Attitude-wise, the target starts the simulation aligned with the J2000 inertial axes. The initial orientation of the chaser corresponds to a pointing error of 30° with respect to the target relative position. Finally, a general angular velocity is applied to the target, while the chaser starts with a rotation state that guarantees that the target will enter the camera's LOS, as otherwise the navigation system would not obtain measurements. The complete initial conditions are given in Tab. 2, while the parameters that define the camera as well as the constants of the navigation and control algorithms are included in Tab. 3.

The navigation and control parameters are tuned heuristically and validated through simulation. The filter's noise characteristics and covariance matrices are likewise adjusted empirically, since the error distribution produced by the vision algorithm is not explicitly characterized. Consistent with standard control practice, the estimator bandwidth is selected to be significantly higher than that of the controller to prevent the introduction of additional, unmodeled dynamics into the closed-loop system.

The baseline PD controller is also tuned heuristically. The proportional gain $\mathbf{K}_p$ is chosen to obtain the fastest possible response without saturating the control under normal operation, while the derivative gain $\mathbf{K}_d$ aims to minimize the oscillations of the closed-loop system.

Object	Vector	W	X	Y	Z	Units
Target	Position	–	6198.143815	2741.290040	2300.215461	km
	Velocity	–	-3.731411476	4.950940839	4.154332632	km/s
	Attitude	1.00000	0.00000	0.00000	0.00000	–
	Angular velocity	–	0.00300	0.00100	0.00020	rad/s
Chaser	Position	–	6198.135887	2741.288302	2300.214003	km
	Velocity	–	-3.731416550	4.950945974	4.154336941	km/s
	Attitude	0.92904	-0.02310	-0.08622	0.35906	–
	Angular velocity	–	0.00000	0.00000	-0.00100	rad/s

Table 2 Initial position, velocity, attitude, and angular velocity of Target and Chaser in the J2000 frame.

Vision	Width (px)	Height (px)	FOV (°)	Framerate (Hz)
	512	512	20	0.1
Navigation	$\mathbf{Q_d}$ (–, s^{-1}, s^{-2})	$\mathbf{R}$ (–)	$\mathbf{P}^+(\mathbf{0})$ (–, s^{-1}, s^{-2})	α (–)
	$0.05 \cdot \mathbf{I}_6$	$0.1 \cdot \mathbf{I}_3$	$10^{-3} \cdot \mathbf{I}_6$	0.5
Backstepping Control	$\mathbf{K}_1$ (s^{-1})	$\mathbf{K}_3$ (kg m^2 s^{-1})	$\dot{h}_{\max}$ (N m)	Sampling rate (Hz)
	$0.039 \cdot \mathbf{I}_3$	$8.7 \cdot 10^{-5} \cdot \mathbf{I}_3$	10^{-6}	0.1
PD Control	$\mathbf{K}_p$ (kg m^2 s^{-2})	$\mathbf{K}_d$ (kg m^2 s^{-1})	$\dot{h}_{\max}$ (N m)	Sampling rate (Hz)
	$2 \cdot 10^{-6} \cdot \mathbf{I}_3$	$3 \cdot 10^{-4} \cdot \mathbf{I}_3$	10^{-6}	0.1

Table 3 Vision, navigation, and control parameters

5.3 Closed loop performance

The scenario is simulated with and without attitude perturbations, namely, magnetic torque and gravity gradient, to evaluate the robustness of the controller against unmodeled effects. To assess the pointing error, two metrics are considered. Firstly, the chaser's pointing vector, aligned with $\mathbf{X}^B$, is compared to the normalized relative position of the target $\mathbf{\Gamma}$, that is, the desired pointing direction. Representing these vectors in the inertial frame enables the evaluation of the controller and the complexity of the reference trajectory. Secondly, the pointing error angle is computed, that is, the angle measured from $\mathbf{X}^B$ to the actual target direction $\mathbf{\Gamma}$. Finally, the pointing error from the estimator is defined as the angle error between the direction towards the target $\mathbf{\Gamma}$ and the one provided by the modified MEKF $\hat{\mathbf{\Gamma}}$. These magnitudes are depicted in Fig. 7 for the simulation with and without attitude perturbations. The time instants at which the vision algorithm does not detect the target are shaded in red.

As seen in the figures, the control system correctly steers the chaser to keep a small pointing error as soon as the vision system begins reporting the target's relative position. The pointing error is mostly due to the control system, as the modified MEKF outputs a precise estimate as long as the target is visible. The total error is larger in the simulation with perturbations because the effects of gravity gradient and magnetic field have not been considered during controller design. Nonetheless, the pointing architecture performs robustly in the operating condition. Furthermore, note that the controller keeps a small angular error even when target visibility is lost, with a performance decrease under the effect of perturbations.

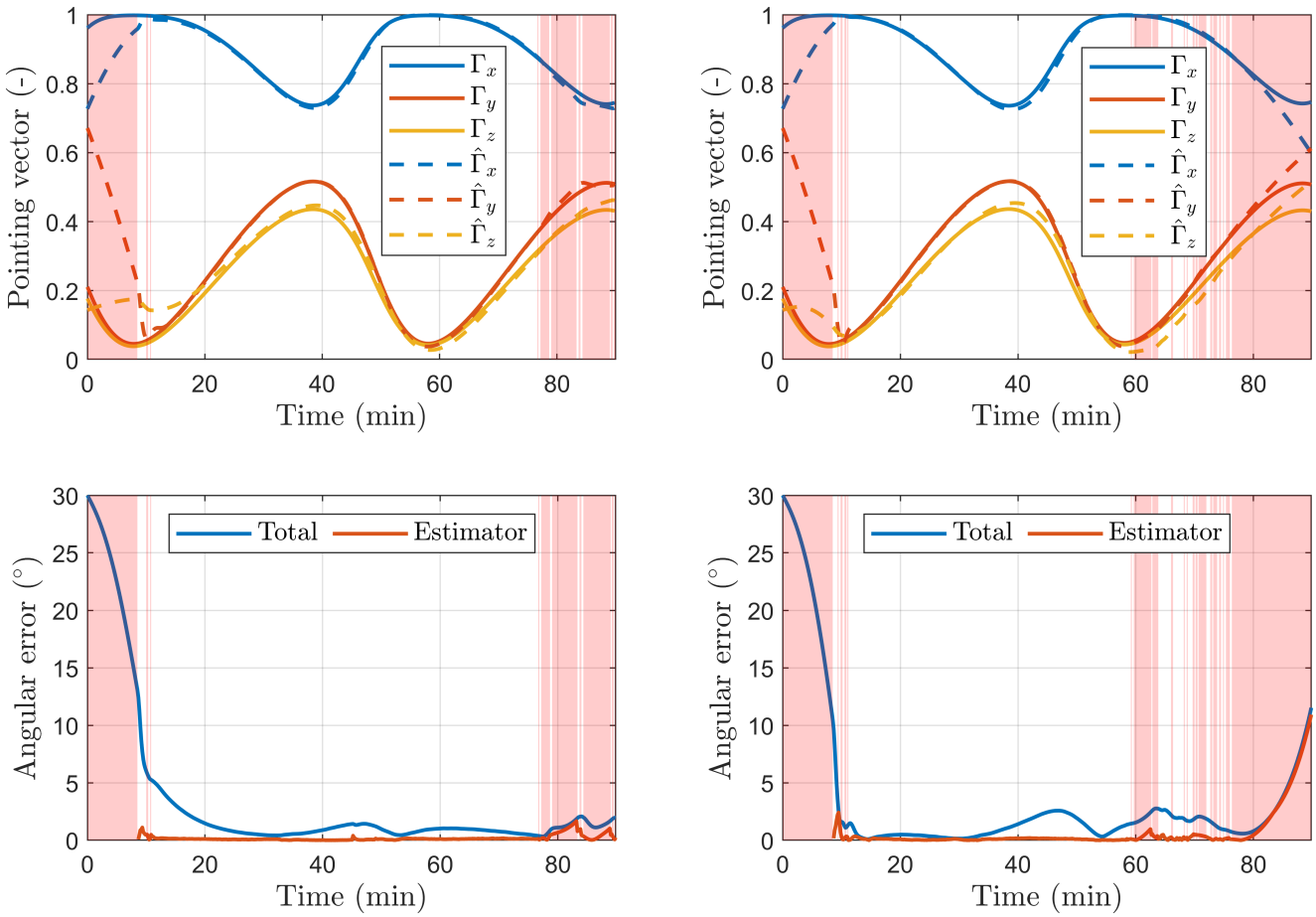

Fig. 7 Results for the pointing errors using the backstepping controller without (left) and with (right) attitude perturbations.

Table 4 summarizes the performance of the estimator and the integrated system (comprising the estimator and controller). The estimator performance is evaluated over the interval spanning from the epoch of the initial estimation to the point of consistent target loss. For the unperturbed scenario, this interval concludes at $t \simeq 60$ min, whereas for the perturbed case, it extends to $t \simeq 77.1$ min. In both instances, the estimator maintains a Root-Mean-Square (RMS) error below 0.3°. The superior performance observed in the unperturbed case is expected, as the linearized dynamical model within the MEKF is formulated based on unperturbed dynamics.

The performance of the integrated system is assessed from the epoch at which the tracking error first drops below 2° until the loss of the target. In the unperturbed case, the tracking error remains below 2° for the duration of the interval. In contrast, the perturbed case reaches a maximum error of 2.414°. Despite this, both scenarios exhibit a total system RMS error of less than 0.28°.

The last column of Tab. 4 shows the performance of the closed-loop system under the actuation of the baseline controller. As expected, it performs worse than the proposed backstepping controller. Nevertheless, it still provides a reasonably accurate pointing, which can be explained because, close to the origin (0° pointing error), the linearized equations of the PD formulation behave well. A more degraded actuation would be observed farther from such origin. However, given that the camera's FOV is 20° in this scenario, the target, if seen, cannot be more than 10° away from the chaser's LOS, a value small enough for the linearized equations to apply.

Regarding actuation, the control signal $\dot{\mathbf{h}}$ is bounded to a maximum value of $|\dot{\mathbf{h}}| < 10^{-6}$ N m in each axis. The time evolution of this magnitude, as well as the total angular momentum in the wheels, is shown in Fig. 8.

		Non perturbed	Perturbed	Perturbed, PD baseline
Estimator	Max. error [deg]	1.108	2.414	2.547
	RMS error [deg]	0.153	0.272	0.291
Total	Max error. [deg]	1.961	2.576	6.9249
	RMS error [deg]	0.919	1.212	3.944

Table 4 Summary of estimator and full system (estimator + controller) performance.

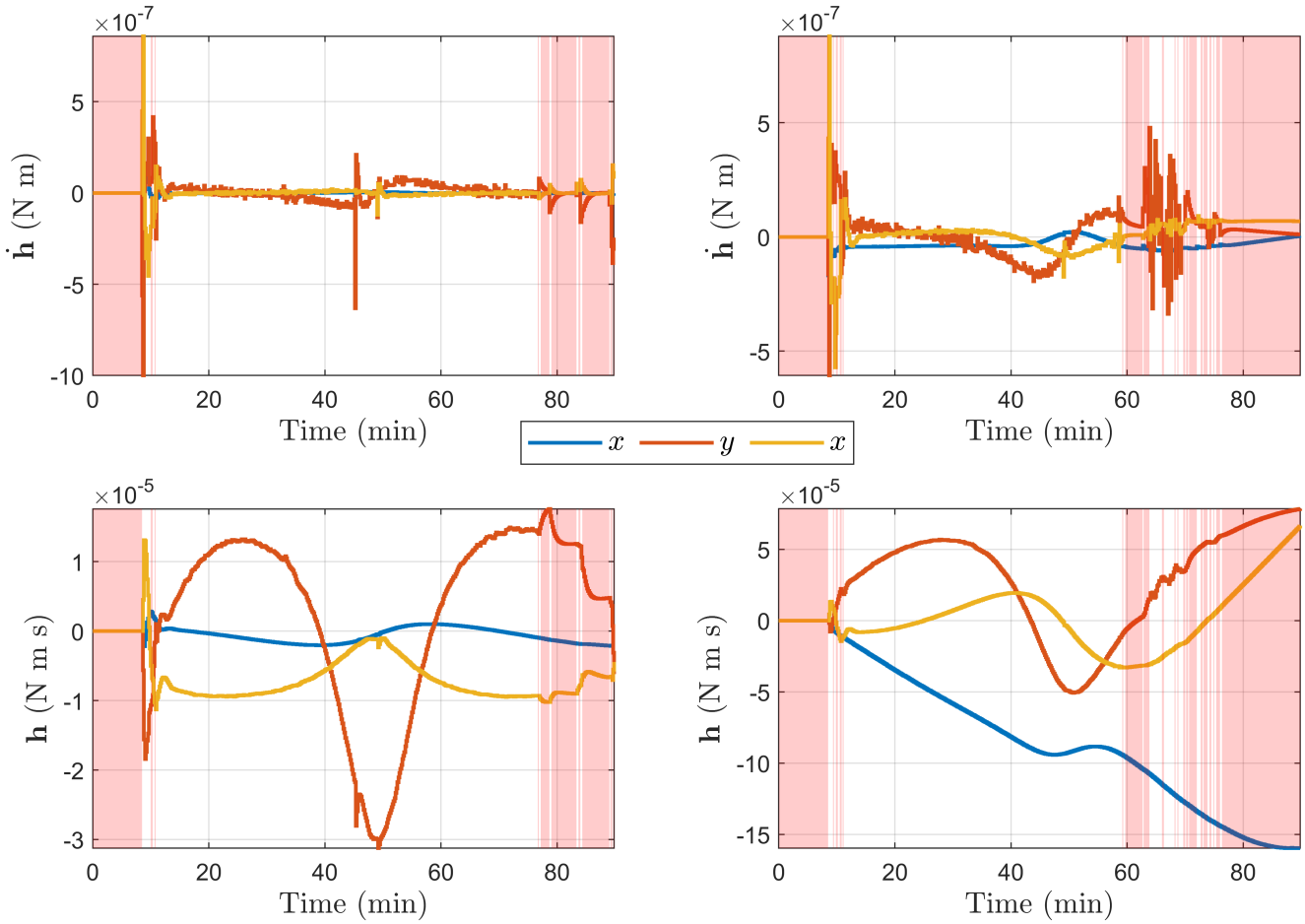

Fig. 8 Control signal and total angular momentum in the reaction wheels for the backstepping controller without (left) and with (right) attitude perturbations.

In general, the presence of perturbations increases the needed control authority, given the total angular momentum increase with time. This behavior is expected, as there is no mechanism for angular momentum dissipation, necessary if a steady perturbation rejection is performed. The simulation of these scenarios allows identifying the control requirements and selecting actuators accordingly. For instance, these performed simulations indicate that perturbations, especially magnetic torque, produce relevant effects on the satellite's dynamics, given the considered magnetic dipole and inertia tensor.

The recordings captured by the chaser camera, directly generated by ***Orbita*** during the simulations, can be seen here [4]. The vision algorithm performs adequately even when the Earth is visible, as shown in Fig. 5. Note that changes in illumination significantly affect the behavior of the target detection.

[4]https://www.youtube.com/watch?v=5gSQOjFR_pg

6 Conclusions

In this work, a vision-based ADCS for attitude close-proximity operations of CubeSats was developed and simulated in ***Orbita***. A simple image processing pipeline was proved to perform adequately in the studied conditions when combined with a nonlinear relative attitude estimator and a backstepping Lyapunov controller. The system was tested in realistic conditions, assessing the effect of attitude perturbations on the pointing error. For the operating conditions and experiment duration, perturbations have a significant effect on the control requirements, namely, total angular momentum accumulated in the inertia wheels.

Future research may consider more advanced vision systems, improving robustness in more scenarios, for varying illumination setups and satellite geometries. Furthermore, the addition of a magnetometer on the chaser spacecraft would allow improving the capabilities of the controller through a more realistic dynamic model, without an increase of computational cost. Indeed, direct measurements of the magnetic field would provide a practical alternative to a dipole approximation or expensive harmonic expansions. The developed system can be combined with a low thrust solution to perform autonomous close-proximity operations or formation flying.

This work serves as a relevant example of the current capabilities of ***Orbita***, a powerful software for space system design, simulation and validation under realistic conditions. Our team is developing relevant functionalities as the ones highlighted in this study to bridge the gap between research on new GNC architectures and high performance, realistic simulation. Additional features, like uncertainty and noise modeling or built-in Monte Carlo methods, are currently under development, and will be considered in future studies.

Declaration of Use of Artificial Intelligence

Artificial intelligence was used only for minor refinement of the English translation.

References

[1] Heike Benninghoff, Toralf Boge, and Tristan Tzschichholz. Hardware-in-the-loop rendezvous simulation involving an autonomous guidance, navigation and control system. *Advances in the Astronautical Sciences*, 2012.

[2] P. Gasbarri, M. Sabatini, and G.B. Palmerini. Ground tests for vision based determination and control of formation flying spacecraft trajectories. *Acta Astronautica*, 102:378–391, 2014. ISSN: 0094-5765. doi: 10.1016/j.actaastro.2013.11.035.

[3] Brent E Tweddle and Alvar Saenz-Otero. Relative computer vision-based navigation for small inspection spacecraft. *Journal of guidance, control, and dynamics*, 38(5):969–978, 2015. doi: 10.2514/1.G000687.

[4] Dehann Fourie, Brent Tweddle, Steve Ulrich, and Alvar Saenz Otero. Vision-based relative navigation and control for autonomous spacecraft inspection of an unknown object. In *AIAA guidance, navigation, and control (GNC) conference*, page 4759, 2013. doi: 10.2514/6.2013-4759.

[5] Jorge Pomares, Leonard Felicetti, Gabriel J García, and José L Ramón. Spacecraft formation keeping and reconfiguration using optimal visual servoing. *The Journal of the Astronautical Sciences*, 71(2):19, 2024. doi: 10.1007/s40295-024-00439-6.

[6] Anne Mergy, Gurvan Lecuyer, Dawa Derksen, and Dario Izzo. Vision-based neural scene representations for spacecraft. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2021. doi: 10.48550/arXiv.2105.06405.

[7] WU Yunhua, Mohong Zheng, HE Wei, WANG Feng, CHEN Zhiming, and HUA Bing. High precision attitude dynamic tracking control of a moving space target. *Chinese Journal of Aeronautics*, 32(10):2324–2336, 2019. doi: 10.1016/j.cja.2019.06.005.

[8] Byron D. Tapley, John C. Ries, Srinivas Bettadpur, Don P. Chambers, Minkang Cheng, Florent Condi, Brian C. Gunter, Z. Kang, Patrick Nagel, R. Pastor, T. Pekker, S. Poole, and F. Wang. GGM02 - An improved Earth gravity field model from GRACE. Technical report, Center for Space Research, The University of Texas at Austin, 2004. Model GGM02C. doi: 10.1007/s00190-005-0480-z, `https://www2.csr.utexas.edu/grace/gravity/`.

[9] National Oceanic and Atmospheric Administration; National Aeronautics and Space Administration; United States Air Force. U.S. Standard Atmosphere, 1976. Technical Report NOAA-S/T 76-1562, Washington, D.C., 1976. `https://ntrs.nasa.gov/citations/19770009539`.

[10] Arnaud Chulliat, Patrick Alken, Manoj Nair, W. Brown, C. D. Beggan, G. A. Cox, A. Woods, S. Macmillan, B. Hamilton, and B. Meyer. The US/UK World Magnetic Model 2020-2025. Technical report, National Centers for Environmental Information, NOAA and British Geological Survey, 2020. doi: 10.25923/ytk1-yx35, `https://www.ncei.noaa.gov/products/world-magnetic-model`.

[11] Randima Fernando, Matt Pharr, and Hubert Nguyen, editors. *GPU Gems Collection: Volumes 1–3*. Addison-Wesley, 2004–2007. A series of three volumes published by NVIDIA and Addison-Wesley.

[12] Connor Beierle, Joshua Sullivan, and Simone D'Amico. Design and Utilization of the Stanford Vision-Based Navigation Testbed for Spacecraft Rendezvous. In *Proceedings of the 9th International Workshop on Satellite Constellations and Formation Flying, University of Colorado*, 2017.

[13] Sijia Qiao, Haopeng Zhang, Fengying Xie, and Zhiguo Jiang. Deep-learning-based direct attitude estimation for uncooperative known space objects. *IEEE Transactions on Aerospace and Electronic Systems*, 60(3):2526–2541, 2023. doi: 10.1109/TAES.2023.3325801.

[14] Alessandro Lotti, Dario Modenini, Paolo Tortora, Massimiliano Saponara, and Maria A Perino. Deep learning for real-time satellite pose estimation on tensor processing units. *Journal of Spacecraft and Rockets*, 60(3):1034–1038, 2023. doi: 10.2514/1.A35496.

[15] Rafael C. Gonzalez and Richard E. Woods. *Digital image processing*. Prentice Hall, 2008.

[16] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2nd edition, 2012.

[17] F Landis Markley. Multiplicative versus additive filtering for spacecraft attitude determination. In *6th International Conference on Control of Systems and Structures in Space*, 2003.

[18] Divya Bhatia. Spacecraft high accuracy attitude estimation: Performance comparison of quaternion based ekf and uf. In *AAS/AIAA Astrodynamics Specialist Conference*, 2017.

[19] Yigal Zivan and Daniel Choukroun. Dual quaternion kalman filtering and observability analysis for satellite relative navigation with line-of-sight measurements. *IEEE Transactions on Aerospace and Electronic Systems*, 58(2):754–765, 2021. doi: 10.1109/TAES.2021.3115584.

[20] Imran Ali, Gianmarco Radice, and Jongrae Kim. Backstepping control design with actuator torque bound for spacecraft attitude maneuver. *Journal of guidance, control, and dynamics*, 33(1):254–259, 2010. doi: 10.2514/1.45541.

[21] Hans-Christian Jensen and Rafal Wisniewski. Quaternion feedback control for rigid-body spacecraft. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 4338, 2001. doi: 10.2514/6.2001-4338.

[22] Fuyuto Terui. Position and attitude control of a spacecraft by sliding mode control. In *Proceedings of the 1998 American Control Conference. ACC (IEEE Cat. No. 98CH36207)*, volume 1, pages 217–221. IEEE, 1998. doi: 10.1109/ACC.1998.694662.

[23] Wencheng Luo, Yun-Chung Chu, and Keck-Voon Ling. Inverse optimal adaptive control for attitude tracking of spacecraft. *IEEE Transactions on Automatic Control*, 50(11):1639–1654, 2005. doi: 10.1109/TAC.2005.858694.

[24] Long-Life Show, Jyh-Ching Juang, Chen-Tsung Lin, and Ying-Wen Jan. Spacecraft robust attitude tracking design: PID control approach. In *Proceedings of the 2002 American control conference (IEEE cat. No. CH37301)*, volume 2, pages 1360–1365. IEEE, 2002. doi: 10.1109/ACC.2002.1023210.

[25] Hassan K Khalil and Jessy W Grizzle. *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ, 2002.

[26] WH Clohessy and RS Wiltshire. Terminal guidance system for satellite rendezvous. *Journal of the aerospace sciences*, 27(9):653–658, 1960.