


Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería Aeronáutica y del Espacio

Máster Universitario en Sistemas Espaciales

Mejora de simulador y analisis de viabilidad de misiones espaciales dedicadas a Space Situational Awareness (SSA)

Caso de estudio 2

January 20, 2024

Autora:
Inés Arauzo Andrés

Contents

1 Introduction

The continuous growth in the number of objets (active satellites and space debris) has significantly raised the probability of collisions, endangering the safety of spacecrafts and satellite constellations. Hence, in order to ensure the safety and organization of activities in space, it has become crutial to quickly develop and enhance Space Situational Awareness (*SSA*).

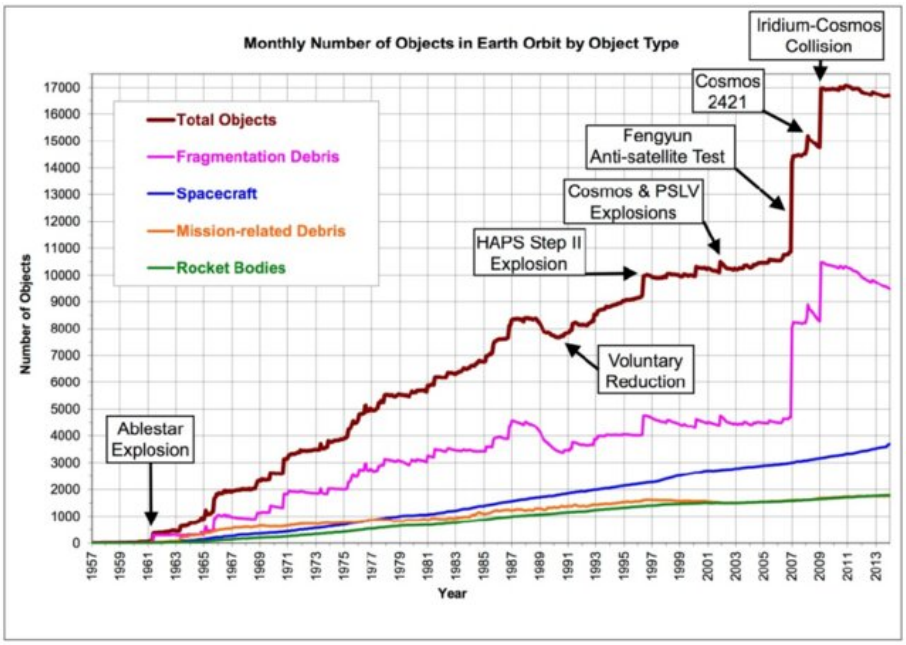

Figure 1.1: Timeline of orbital space debris growth.

In this context, *SSA*, which is defined as the knowledge and characterization of space objects and their operational environment, has evolved into a pivotal solution to mitigate the growing hazards in space. While terrestrial *SSA* has worked as the foundational framework, the dynamic evolution of the space environment needs a paradigm shift towards space-centric *SSA*. This strategic transition entails considerable advantages in precision, operational efficiency, and system stability as compared to its terrestrial counterpart.

The area under surveillance now extends from low Earth orbits (LEO) to a terrestrial radius of at least 100,000 km. This broader coverage essentially includes all objects in orbit, encompassing both artificial and natural entities that may pose potential risks. However, there is a remarkable interest in the Low Earth Orbits due to the increasing density of satellite deployments and space activities within this region.

With the objective of reaching the desired scope of monitoring, the adoption of advanced remote sensing and continuous monitoring technologies has been proved essential. These technologies enable the precise identification and analysis of space threats, including debris and asteroids, contributing to a strong and proactive *SSA*.

In this context, there is an imperative need for the development of specialized simulators tailored to the precise detection and tracking of space debris. These simulators must be able to provide a faithful representation of the space environment, allowing the training and validation of *SSA* strategies to work.

Space Situational Awareness, particularly in the realm of debris detection and tracking, stands as a critical component for space safety and sustainability. The adoption of advanced technologies, the transition to space-based *SSA*, and the development of specialized simulators are essential steps toward the effective management of risks in orbit.

In conclusion, the increasing challenges arising from the growing number of objects in space underscore the need for a comprehensive approach to Space Situational Awareness (SSA). To address this issue, the integration of advanced technologies and the development of specialized simulators become pivotal elements. These measures collectively play a crucial role in ensuring the safety and sustainability of activities within Earth's orbit. By combining cutting-edge technologies with purpose-built simulators, we can effectively navigate the evolving complexities associated with managing risks in space.

1.1 Motivación para Mejorar el Simulador de *SSA*

Recognizing the urgent need to systematically register and locate space debris, an initial version of a Space Situational Awareness (SSA) simulator was developed. This simulator, characterized by a basic design and a notably disorganized code structure (explained in Section 2), is being requested for improvement to achieve preliminary results.

The main motivation for enhancing the SSA simulator comes from the necessity to integrate a specialized photometry module. Photometry, which involves precise measurements of light emitted by celestial entities, is of utmost importance for the detailed characterization of space debris. The addition of this module is expected to enable a more accurate simulation of the visual attributes of celestial objects in space, thereby strengthening the simulator's capacity to faithfully replicate real-world conditions.

In the process of integrating the new photometry module, the primary objective is to determine the effective range of the sensors. This step is pivotal as it serves as the foundation for the development of essential positioning functions. By accurately establishing the sensor range, these functions will facilitate the strategic positioning of observers in specific orbits. The goal is to ensure optimal visibility of the target by a minimum of 1 observer. This meticulous approach not only enhances the simulator's spatial accuracy but also improves its ability to replicate real-world observational scenarios with greater precision.

Moreover, when considering the evaluation of simulator accuracy, the focus extends beyond a general assessment based on observer numbers and average visibility times. The intention is to delve into the intricate interplay between these factors, seeking to identify patterns and dependencies that may impact the simulator's predictive capabilities. Through an analysis of how the simulator responds to variations in observer counts and average visibility times, valuable insights into the subtleties of its performance under diverse conditions can be obtained. This comprehensive examination is indispensable for validating the simulator's effectiveness in providing reliable space situational awareness, thereby establishing a robust foundation for advancements in the understanding and monitoring of space debris.

2 Code development

2.1 Code Refactoring for Improved Modularity in a Space-Based SSA Simulator

As discussed in the introduction, the initial codebase of the simulator comprised a single folder containing a highly extensive main code. This code encompassed simulation parameters, the structure and parameters of the estimator, simulation errors, and other essential components. Within this folder, there were some other functions used by the main program and another subfolder housing the simulation program itself, invoked from the main program. In essence, the initial simulator had been developed following a spaghetti-code philosophy, lacking a structured approach that would facilitate both code comprehension and the implementation of potential enhancements.

Motivated by the need for a more organized and maintainable codebase, a decision was made to refactor the code into modules, adhering to an imperative programming paradigm. This restructuring aimed to enhance code readability and facilitate the subsequent implementation of various functions and modules, providing a foundation for future improvements.

The refactoring process involved the systematic division of the monolithic code into distinct modules, each addressing specific functionalities or components of the simulator. This modular approach not only promotes code clarity but also streamlines the integration of new features and the maintenance of existing ones.

The refactored code now embraces a more modular and organized structure, aligning with best practices in software engineering. This restructuring sets the stage for further advancements and improvements in the space-based Space Situational Awareness (SSA) simulator, fostering a more sustainable and extensible development process.

The restructured code now features a main code accessed through a more user-friendly interface (*SimConfig.m*). The code is organized into several modules residing in subfolders:

- **Estimation Module:** This module houses functions employed as estimators. Currently, the sole proposed estimator is the Unscented Kalman Filter (UKF) which implies several functions. However, any additional estimators introduced in the future would find their place within this module.
- **Physics Module:** This module contains functions that mathematically represent the physics of the problem intended for simulation. It encompasses aspects such as orbit propagation, solar ephemerides, and other relevant physical phenomena.
- **Simulation Module:** The core simulation code (*Data_Simulation.m*) resides in this module, along with the code responsible for generating the desired type of satellite (*Satellite.m*).
- **Photometry Module:** This module incorporates functions developed in Section 3, addressing photometric considerations within the simulation.
- **Utilities Module:** Essential functions for the proper execution of the program are stored here. These include functions for coordinate transformations, quaternion-to-rotation matrix conversions, and other utility functions.

This modular organization enhances code manageability, readability, and extensibility. It establishes a clear separation of concerns, allowing for straightforward integration of new functionalities and easing the maintenance process. The user-friendly entry point (*SimConfig.m*) serves

as a central interface, providing a more intuitive and accessible means to configure and run the simulator.

2.2 Program Workflow and Execution Sequence

As discussed earlier, the program operation unfolds through two primary components: the simulator and the estimator. These modules sequentially execute upon being invoked from the main program, where the main parameters of the simulation are introduced, as defined in *Sim_config.m*. These parameters encompass the dimensions, attitude, and initial position of the target debris, observer number and altitude, as well as other simulation parameters such as duration and start date. The following code excerpt illustrates the parameters fed into the main program:

```
% Global Constants
mu=3.986044418e14;
Rt = 6878000;

% Debris Parameters
sim_params.sat_size = 2;
sim_params.sub_cat = 3;
sim_params.deploy_panels = 0;
sim_params.rt_init = [500, 0, 0]*1000 + Rt;
sim_params.qt =
[-0.401437297624753;-0.578215887659600;0.239454457519217;0.668712229668289];

% 7Observers Parameters
sim_params.n_observers = 5;  %
sim_params.d_1st0bs_target = 200000;
sim_params.ro_init = [750, 0, 0] * 1000 + Rt;

% Simulation Parameters
sim_params.total_sim_time = 2*pi*sqrt(sim_params.ro_init(1)^3/mu);
sim_params.Year = 2023;
sim_params.Month = 2;
sim_params.Day = 25;
sim_params.Hour = 15;
sim_params.Min = 0;
sim_params.Sec = 1;
```

Listing 2.1: Parameters introduced into the *Main_Program.m* through *Sim_config.m*.

2.2.1 Simulation Algorithm

Upon the invocation of the simulator (*Data_Simulation.m*), the pre-established parameters, coupled with the defined simulation time steps are incorporated to the simulation workspace. The "*Debris Parameters*" defined in the code above, are used as an input to a MATLAB predefined function named "Satellite" that defines the target as a satellite of the desired dimensions. Subsequently, leveraging the target's position and observer data, two functions are employed to position the observers in orbital locations conducive to observing the target.

The strategic decision was made to position all observers initially in a shared coplanar orbit with the target, visually represented in Figure 2.1. This aligns with the constellation structure outlined in [**constellations**], optimizing the observational efficiency of the network. To define this configuration, two distinct positioning functions were developed.

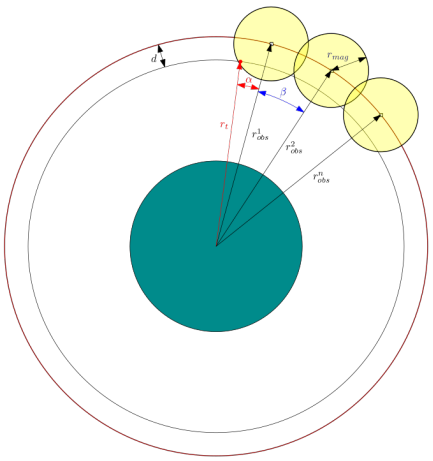

Figure 2.1: Observers and target orbits configuration

The first of these functions, namely *generate_position_1st_obs.m*, takes the lead in constructing the first observer initial position within the specified orbit, drawing parameters from the configuration file. The observer is placed at an angular distance denoted as α from the target.

Afterwards, a complementary function, *generate_position_N_obs.m*, was conceptualized to handle the placement of the remaining observers within the same orbit. This function defines an angular separation denoted as β between the observers. It is worth emphasizing that both α and β distances are inherently linked to the observer's effective sensor range, defined in the simulation as the radius of the visibility sphere.

It is crucial to highlight that the developed functions are designed to provide both Above the Horizon (ATH) and Below the Horizon (BTH) coverage, as elucidated in [**ATH-BTH**]. The distinction in positioning arises from the observer's orbital semi-major axis concerning that of the target. When the observer is in an orbit with a semi-major axis greater than that of the target, it will exhibit a lower orbital velocity. To extend the observation time, the observers are strategically positioned with a true anomaly (ν) greater than that of the target, given by $\nu_{obs_1} = \nu_t + \alpha$. Conversely, when the observer is in a lower orbit (providing ATH coverage), it will have a higher orbital velocity. To maximize the observation window in this scenario, the observers are situated with a true anomaly given by $\nu_{obs_1} = \nu_t - \alpha$. This strategic placement ensures comprehensive coverage by adapting to the inherent differences in orbital dynamics.

The nuanced approach to observer positioning becomes evident in the following code snippet:

```
function [r, v] = generate_1st_obs_pos(rt, vt, d)
% Initialize output arrays
r = zeros(3, 1);
v = zeros(3, 1);

% Extract Keplerian elements from the target state vectors
[a_t, ecc_t, inc_t, RAAN_t, argp_t, nu_t] = ijk2keplerian(rt, vt);

% Constants, define as wanted
radius_coverage_sph = 1000000; % [m]this is a function of the magnitude
initial_offset_o1_t = 10000;

% Compute adjusted semi-major axis for observers
a_obs = a_t + d;

% Compute initial angular offset between the target and the first
% observer
```

```
    alpha = acosd((a_obs^2 + a_t^2 - initial_offset_o1_t^2) / (2 * a_obs * a_t));
    if ~isreal(alpha)
        alpha = 0;
    else
        alpha = mod(alpha, 360);
    end
    % Determine true anomaly for observer 1, with the offset stated
    % previously
    if a_obs > a_t
        nu_obs1 = mod(nu_t + alpha, 360);
    else
        nu_obs1 = mod(nu_t - alpha, 360);
    end

    % Convert observer 1's Keplerian elements to Cartesian coordinates
    [r, v] = keplerian2ijk(a_obs, ecc_t, inc_t, RAAN_t, argp_t, nu_obs1);
end
```

Listing 2.2: Excerpt of function *generate_position_1st_obs.m*.

The simulation workflow continues by propagating the initial positions of both the observers and the target along their respective orbits, leveraging the Keplerian orbit equation,

$$\begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{Bmatrix} = \begin{Bmatrix} \dot{\boldsymbol{r}} \\ -\frac{\mu}{r^3}\boldsymbol{r} + a_p \end{Bmatrix}. \tag{2.2.1}$$

In this particular case, the consideration of acceleration attributable to perturbations, denoted as a_p, has not been directly incorporated into the analysis. Rather, the introduced noise has been conceptualized as a surrogate for perturbations in the system.

The numerical Runge-Kutta 4 scheme is employed to ensure accurate and stable propagation, as given by the following equations:

$$\begin{aligned} X_{n+1} &= X_n + \frac{\Delta t}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \\ t_{n+1} &= t_n + \Delta t, \quad , with n = 0, 1, 2, 3..., \end{aligned} \tag{2.2.2}$$

where $X = (x, y, z, \dot{x}, \dot{y}, \dot{z})$ represents the state vector, which comprises the position $r = (x, y, z)^T$ and the velocity $v = (\dot{x}, \dot{y}, \dot{z})$, t denotes time, and k_1, k_2, k_3, and k_4 are intermediary values calculated as per the Runge-Kutta method,

$$\begin{aligned} k_1 &= f(t_n, X_n), \\ k_2 &= f\left(t_n + \frac{\Delta t}{2}, X_n + \Delta t\frac{k_1}{2}\right), \\ k_3 &= f\left(t_n + \frac{\Delta t}{2}, X_n + \Delta t\frac{k_2}{2}\right), \\ k_4 &= f(t_n + \Delta t, y_n + \Delta t k_3). \end{aligned} \tag{2.2.3}$$

After propagating the orbits, a controlled positioning error, alongside attitude and directional errors, is introduced to the observer. This step is crucial for simulating potential deviations in the measurements that may arise from real-world scenarios. The introduced errors follow a bounded distribution, ensuring a realistic yet manageable level of uncertainty. Both the perturbed and unperturbed datasets are stored as outputs of the simulator.

2.2.2 Estimator Algorithm: Unscented Kalman Filter

As highlighted in the introduction, a fundamental achievement of Space Situational Awareness (SSA) systems revolves around the determination of Resident Space Object (RSO) size and position. This estimation, often articulated through Keplerian parameters, Cartesian coordinates, or alternative representations, plays a pivotal role in predicting the RSO's future position with a desirable degree of precision. In this study, a streamlined adaptation of the algorithm presented in [**linares**] has been employed to focus on estimating the variables inherent in the state vector X.

The process of estimating state variables within the state vector is a nuanced task, and it is typically approached using well-established algorithms. Among these, two extensions of the Standard Kalman Filter (KF), namely the Extended Kalman Filter (EKF) [**EKF**] and the Unscented Kalman Filter (UKF) [**UKF**] stand out as widely adopted methodologies. Both of these algorithms inherently consist of two principal steps:

- **The prediction step**: in the initial phase, the algorithm uses the system's propagation model (in this case RK4) to forecast its future state (referred to as the a priori state). During this step, the associated covariance matrix for the state variables is also projected forward.
- **Correction step**: In this subsequent phase, the algorithm compares the real system's measurement with the predicted one based on the earlier forecasted state. Using the outcome of this comparison and the optimal correction gain, the predicted state is adjusted (referred to as the a posteriori state), and concurrently, the covariance matrix is updated.

Presently, due to considerations outlined in [**ref:UKFvsEKF**], the decision has been made to employ the Unscented Kalman Filter (UKF)whose formulation will be briefly summarized shortly. Nevertheless, owing to the enhanced modular structure of the code, the substitution of the estimator would be a seamless process, contingent upon the introduction of an alternative estimation function into the designated folder.

Formulación del UKF Consider the following non-linear system:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= f(\boldsymbol{x}_k, \boldsymbol{v}_k) \\ \boldsymbol{y}_{k+1} &= h(\boldsymbol{x}_{k+1}, u_k) \end{aligned} \tag{2.2.4}$$

Here, $f(\boldsymbol{x})$ represents the function delineating the evolution of the state variables, where in the present context, it corresponds to Equation 2.2.1. The symbols $\boldsymbol{y}$ denote system measurements, h denotes the measurement function, and v, u signify process and measurement errors, respectively. Furthermore, k is an arbitrary time instant.

Assuming a known (or accurately estimated) covariance matrix of the state variables ($\boldsymbol{P}_k$) at the given time, the calculation of sigma points is as follows:

$$\begin{aligned} \chi_k^0 &= x_k \\ \chi_k^i &= \boldsymbol{x}_k + \left(\sqrt{(n+\lambda)\mathbf{P}_k}\right)_i \quad i = 1...n \\ \chi_k^i &= x_k + \left(\sqrt{(n+\lambda)\mathbf{P}_k}\right)_{i-n} \quad i = n+1 \dots 2n, \end{aligned} \tag{2.2.5}$$

Here, χ denotes the sigma points, n stands for the dimension of the state vector, and λ represents a scale parameter calculated as:

$$\lambda = \alpha^2(n+\kappa) - n, \tag{2.2.6}$$

Parameters α and κ necessitate adjustment based on the filter's requirements. Subsequently, the calculation of weights associated with the sigma points (W) unfolds as follows:

$$\begin{aligned} W^{0,m} &= \frac{\lambda}{\lambda+n} \\ W^{0,c} &= \frac{\lambda}{\lambda+n} + 1 - \alpha^2 + \beta \qquad , \\ W^{i,m} &= W^{i,c} = \frac{\lambda}{2(\lambda+n)} \quad i = 1...2n \end{aligned} \tag{2.2.7}$$

where β emerges as another parameter of the estimator.

After determining the sigma points and their corresponding weights, the next step involves computing the predicted state and covariance matrix:

$$\begin{aligned} \mathbf{x}_{k+1}^{-} &= \sum_{i=0}^{2n} W_i^m f(\chi_k^i) \\ \mathbf{P}_{k+1}^{-} &= \sum_{i=0}^{2n} W_i^c \left(\chi_{k+1}^{-,i} - \mathbf{x}_{k+1}^{-}\right)\left(\chi_{k+1}^{-,i} - \mathbf{x}_{k+1}^{-}\right)^T + \mathbf{Q} \end{aligned} \tag{2.2.8}$$

where $\boldsymbol{Q}$ is a matrix that models the system noise. This first prediction step keeps estimating the estate variables until there is a new experimental (or simulation) measurement available, starting the correction step. When this happens, the expected state is calculated as follows:

$$\begin{aligned} Y_{k+1}^{-,i} &= h(\chi_{k+1}^{-,i}) \\ y_{k+1}^{-} &= \sum_{i=0}^{2n} W_i^m h(\chi_{k+1}^{-,i}). \end{aligned} \tag{2.2.9}$$

Note that $\boldsymbol{Y}$ is the computation of the expected measurement's sigma points. Having the expected measurement, the corrected state can be calculated as:

$$\begin{aligned} \mathrm{P}_{\boldsymbol{yy}} &= \sum_{i=0}^{2n} W_i^c \left(Y_{k+1}^{-,i} - \boldsymbol{y}_{k+1}^{-}\right)\left(Y_{k+1}^{-,i} - \boldsymbol{y}_{k+1}^{-}\right)^{\mathrm{T}} + \mathbf{R} \\ \mathbf{P}_{xy} &= \sum_{i=0}^{2n} W_i^c \left(\chi_k^i - \boldsymbol{x}_{k+1}^{-}\right)\left(Y_{k+1}^{-,i} - \boldsymbol{y}_{k+1}^{-}\right)^{\mathrm{T}} \\ K &= \mathbf{P}_{xy}\mathbf{P}_{yy}^{-1} \\ x_{k+1} &= x_{k+1}^{-} + K(y_{k+1} - y_{k+1}^{-}) \\ \mathbf{P}_{k+1} &= \mathbf{P}_{k+1}^{-} - K\mathbf{P}_{yy}K^{\mathrm{T}}. \end{aligned} \tag{2.2.10}$$

Here, $\boldsymbol{R}$ is a matrix refresenting the noise of the measurement and K is the filter gain to update the sytem and covariance matrix

To use the filter in the most versatile manner possible, it is advantageous that the acquired measurements are independent of the sensors capturing them. Otherwise, it would be neccessary to model the errors introduced by each individual sensor, adding a layer of complexity to the estimation process. Consequently, a decision is made to preprocess the actual measurements, obtained in this case from the simulation, transforming them from the pixel values of light that the sensors would capture to more meaningful quantities for the problem at hand. These transformed

measurements specifically entail the unit vector pointing towards the Resident Space Object (RSO), denoted as $\hat{\rho}$, and the apparent magnitude m. By adopting this preprocessing step, not only is the relevance of the measurements enhanced with respect to the problem domain, but it also circumvents the introduction of noise from diverse sensor equipment, ensuring a more robust and accurate estimation process. The process mede to obtain these values is further explained in **??**

3 Photometry

In order to simplify the inputs fed into the filter, it becomes neccessary to obtain simulation-based measurements that faithfully capture the underlying physics of the problem. To achieve this objective, a dedicated photometry module has been implemented, facilitating the computation of key variables such as the apparent magnitude m and the pointing vector towards the Resident Space Object (RSO), denoted as $\mathbf{d}$. These variables essentially serve as quantifications of the solar illumination reflected by the facets of the RSO, encapsulating both the magnitude and directional characteristics of the reflected light.

3.1 Reflexion model

With the aim of getting information about the sunlight reflection, it will be necessary to ascertain its geometry. In this project, the targets are characterized as a discrete set of planar facets, each defined by a basis of orthogonal unit vectors $(\mathbf{u}_n^B, \mathbf{u}_u^B, \mathbf{u}_\nu^B)$, elucidated in Figure 3.1. As depicted in this illustration, the basis is determined by the normal vector to the facet $(\mathbf{u}_n^B)$ and two additional vectors lying in the plane of the facet and orthogonal to each other $(\mathbf{u}_u^B, \mathbf{u}_\nu^B)$, expressed in body-fixed axes denoted by the superscript B. This discrete representation provides a comprehensive characterization of the target's geometry, forming the basis for subsequent computations related to sunlight reflection.

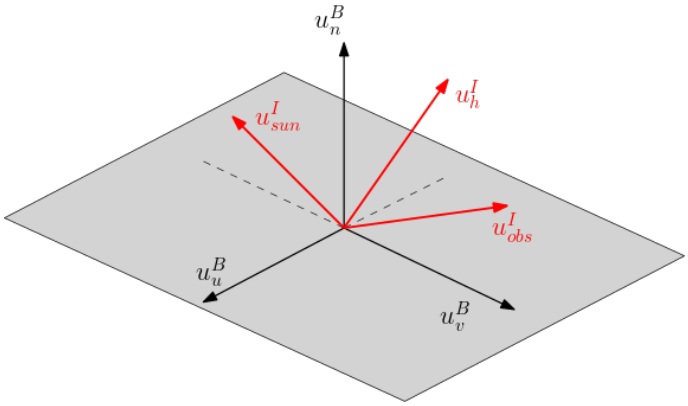

Figure 3.1: Geometría de la reflexión

In addition, in order to obtain the value of the apparent magnitude, it is needed to model the light curve. This entails having the vectors that define the target's facets in inertial coordinates, denoted by the superscript I. To achieve this, the attitude matrix defined by the quaternion associated with the satellite at each moment is employed:

$$\mathbf{u}_k^b = A(\mathbf{q}_\mathbf{I}^\mathbf{B})\mathbf{u}_k^I, \quad k = u, v, n, \tag{3.1.1}$$

Simultaneously, it is fundamental to determine the vector representing the sun with respect to the target, $\mathbf{u}_{\text{sun}}^I$, and the vector depicting the target's orientation concerning the observer, $\mathbf{u}_{\text{obs}}^I$.

With the directions of the reflected light with respect to the inertial frame defined, it is now possible to ascertain the variables that the observer will be able to see.

3.2 Observation model

Consider that the observing satellite is in a position $\boldsymbol{r_{obs}}$, capable of measuring the azimuth and elevation of the target, $\boldsymbol{r_{RSO}}$. The pointing vector from the observer to the RSO is defined as:

$$\boldsymbol{d} = \boldsymbol{r_{RSO}} - \boldsymbol{r_{obs}} \tag{3.2.1}$$

Furthermore, assume that the observer's sensors can measure the magnitude of the reflected light, modeled using the Phong light diffusion model [**Phong**], based on the Bidirectional Reflectance Distribution Function (BDRF). In this reference, the authors decompose the BDRF into a specular part (with a preferred direction), ρ_{spec}, and a diffuse part, ρ_{diff}:

$$\rho_{tot}(i) = \rho_{spec}(i) + \rho_{diff}(i) \quad i = 1...n_c, \tag{3.2.2}$$

where n_c is the number of facets of the RSO. Assuming flat faces, the specular part of the distribution results in:

$$\rho_{\text{spec}}(i) = C_{\text{spec}} \frac{(\mathbf{u^I_{obs}} \cdot \mathbf{u^I_{spec}})}{(\mathbf{u^I_{sun}} \cdot \mathbf{u^I_n})}, \tag{3.2.3}$$

where the vector $\mathbf{u^I_{spec}}$ is the preferred direction of specular reflection, defined as $\mathbf{u^I_{spec}} = 2(\mathbf{u^I_n} \cdot \mathbf{u^I_{sun}})\mathbf{u^I_n} - \mathbf{u^I_{sun}}$. On the other hand, the diffuse term results in:

$$\rho_{diff}(i) = \frac{C_{diff}}{\pi}. \tag{3.2.4}$$

where C_{spec} and C_{diff} are reflection coefficients dependent on the RSO's surface material.

As the apparent magnitude is the measure reaching the observer's sensors of the light reflected by the RSO, the first step is to calculate the fraction of (visible) light reaching the RSO:

$$F_{\text{sun}}(i) = \Phi_{\text{sun,vis}} \rho_{\text{total}}(i)(\mathbf{u}^I_n(i) \cdot \mathbf{u}^I_{\text{sun}}), \tag{3.2.5}$$

where $\Phi_{\text{sun,vis}}$ is the power per unit area received by an object illuminated by sunlight. Only a fraction of the light reflected by the RSO is visible to the observer:

$$F_{\text{obs}}(i) = \frac{F_{\text{sun}}(i)\mathcal{A}(i)(\mathbf{u}^I_n(i) \cdot \mathbf{u}^I_{\text{obs}})}{\|\mathbf{d}^I\|^2}. \tag{3.2.6}$$

Finally, the apparent magnitude is given by a sum of the fraction of light the observer receives from each of the facets the RSO :

$$m_{\text{app}} = -26.7 - 2.5 \log_{10} \left| \sum_{i=1}^{N_F} \frac{F_{\text{obs}}(i)}{\Phi_{\text{sun,vis}}} \right|. \tag{3.2.7}$$

Note that, for the RSO to reflect light, it must be illuminated by sunlight, meaning that the periods of eclipse need to be restricted. Furthermore, if either the angle between the surface normal and the observer or the angle between the surface normal and the Sun direction is greater than $\pi/2$ the observer will not receive any light from this surface.

These calculations have been implemented in the simulator through the *magnitude_apparent.m* function, presented below:

From this function, a significant portion of the results presented in Section 4 has been derived. The *magnitude_apparent.m* function, described earlier, has played an important role in computing various outcomes that contribute to the findings detailed in the subsequent section.

4 Analysis and results

Based on the enhancements made to the simulator and the new implementations in the photometry module, results have been extracted that must be carefully analyzed for the ongoing improvement of the simulator in terms of precision.

4.1 Visibility sphere radius analysis

Primarily, the *magnitude_apparent.m* function has been employed to conduct a comprehensive sweep considering the size of the debris (RSO) and the distance **d** between the observer and the RSO. The objective is to determine the radius of the observer's visibility sphere, a critical parameter in understanding the simulation outcomes and optimizing precision.

To achieve this, multiple simulations have been conducted by placing the observer in a fixed circular orbit with orbital parameters detailed in Table 4.1, which have been chosen to manage a relative position between the observer, the sun and the RSO that allows the visibility of the RSO. Additionally, various RSOs of different sizes have been generated, each placed in circular orbits within the same orbital plane while varying the semi-major axis of their orbits. However, as the attitude of the RSOs is generated randomly, several simulations were conducted for each combination of distance and RSO size, to get the average visibility in each case.

Table 4.1: Orbital parameters of the observer's orbit.

$\boldsymbol{a}$ [km]	$\boldsymbol{e}$ [-]	$\boldsymbol{i}$ [°]
6878	0	0.6

In Figures 4.1 and 4.2, the dependencies among the size, distance, and apparent magnitude of the RSO are illustrated. These figures depict how the apparent magnitude varies concerning the reduction in the RSO size and/or an increase in the distance between the RSO and the observer. It is observed that the magnitude increases (indicating reduced visibility[1]) with a decrease in the RSO size and/or an increase in the separation between the RSO and the observer.

[1]It is noteworthy that as defined in Equation 3.2.7, the more negative the magnitude, the more visible the RSO is to the observer.

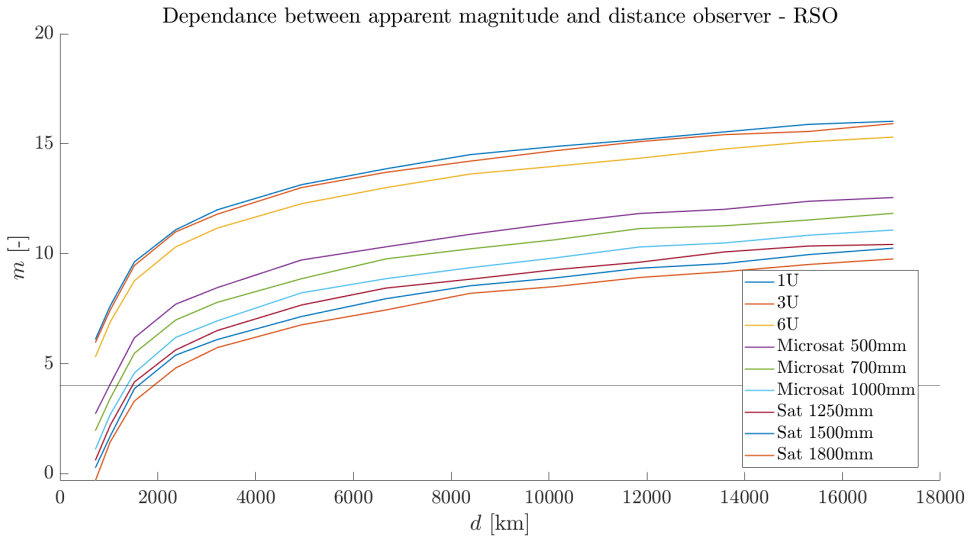

Figure 4.1

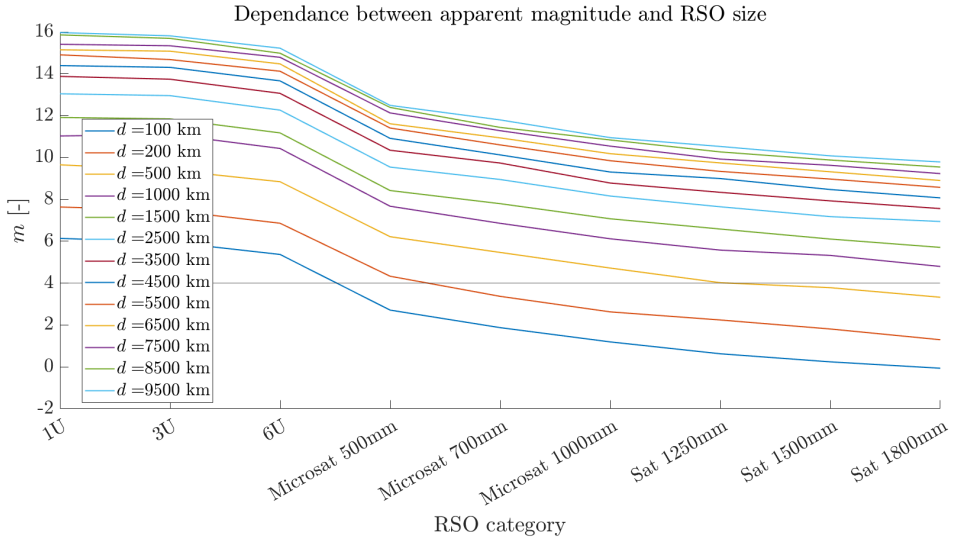

Figure 4.2

The apparent magnitude that the observer is able to perceive is constrained by the optical sensors employed, which will depend on factors such as the mission budget, the number of observers, and the desired constellation type. Initially, the plan is to utilize a constellation of cubesats equipped with basic tracking cameras. Following the guidelines from [**limmapp**], a visibility threshold has been set at $m = 4$.

Returning to the examination of the previous figures, this constraint on the apparent magnitude that the observer can perceive limits a range of distances and sizes of RSOs that the satellite will be able to detect. Thus, it has been established that the observer-RSO limit distance for the sensors to detect the RSO is around $d = 1000\,\text{km}$.

4.2 Observation windows

On the other hand, to comprehend the influence of the Sun's position on the fraction of reflected light reaching the observer, another scan has been conducted. In this case, the observer (again with the orbital parameters outlined in Table 4.1) and the RSO are kept fixed, while the sun's location (in Earth inertial axes) is varied over the course of a year. The outcome of this procedure is depicted in Figure 4.3.

This figure shows the relation between the apparent magnitude of the RSO and θ (defined as the angle formed by the position vector of the RSO $\boldsymbol{r_{RSO}}$ and the Sun $\boldsymbol{r_{Sun}}$projeted into the ecuatorial plane).

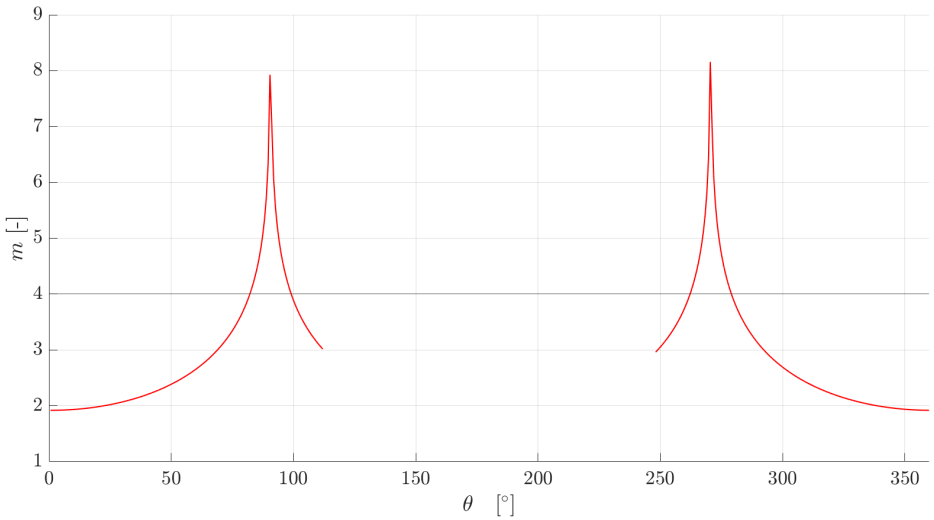

Figure 4.3: Dependance of the apparent magnitude m with the position of the Sun relative to the observer.

Within the figure, it is evident that the optimal solar position occurs when θ is in close proximity to zero. However, it is fundamental to note a potential intricacy in the case of an Above-the-Horizon (ATH) coverage. Specifically, if the target aligns precisely with the sun, the resulting light curve from the target may be entirely eclipsed by the incoming sunlight. Nonetheless, for Below-the-Horizon (BTH) orbits and/or distinct inclinations, the presented outcome within this graph remains accurate. As this angle is not an input to the program, it must be translated to the date it corresponds, which can be seen in Figure 4.4, where the date-defined observation window can be seen.

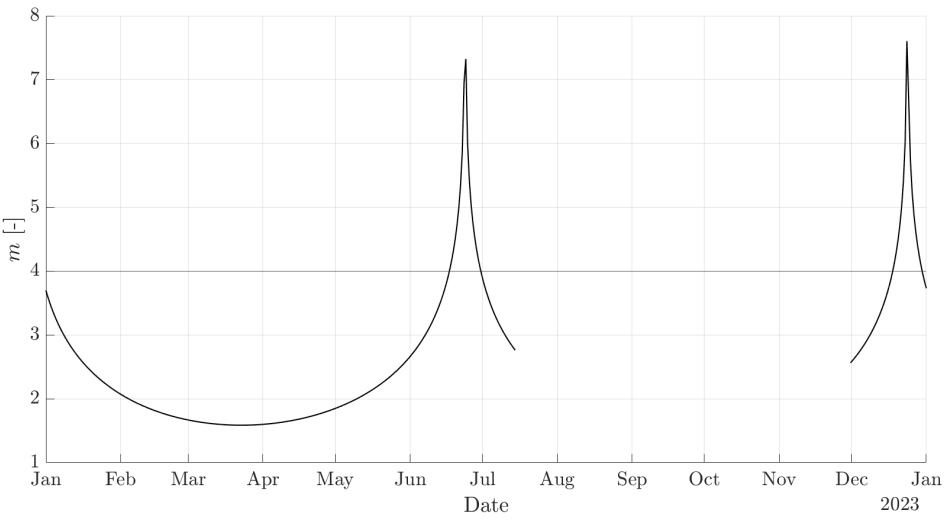

Figure 4.4: Dependance of the apparent magnitude m with the position of the Sun in terms of date.

Table 4.2: Observation window

	Start	**End**
$\boldsymbol{\theta}$ [°]	-60	60
Date	1 January 2023	16 June 2023

Based on these outcomes, observation windows have been delineated to subsequently conduct simulations in which the RSO falls within the field of view of the observer.

With the information of these findings, observation windows have been meticulously defined to conduct subsequent simulations where the RSO resides within the visibility sphere of the observer. It is noteworthy that, for this parameter sweep, a distance of 300 km between the RSO and the observer has been considered, incorporating BTH coverage, and using a square-shaped satellite with a side length of 1 m.

4.3 Visibility times

Subsequently, multiple simulations have been executed with varying numbers of observers to assess visibility durations. For this purpose, the RSO has been fixed in an orbit with parameters outlined in Table 4.3, within the launch windows specified in Table 4.2. It is essential to note that while the observation window scan was conducted for a satellite with a side length of 1 m, in this scenario, to ensure robust visibility regardless of the RSO's attitude, simulations have been conducted with a satellite measuring 1.5 m in side length.

Table 4.3: Orbital parameters of the RSO's orbit.

a **[km]**	e **[-]**	i [°]
6878	0	0.6

The obtained observation times have been illustrated in Figure 4.5 and dimensionless representations, relative to the total simulation time, are presented in Figure 4.6. This approach provides a clear visualization of the fraction of the orbit covered by the observer.

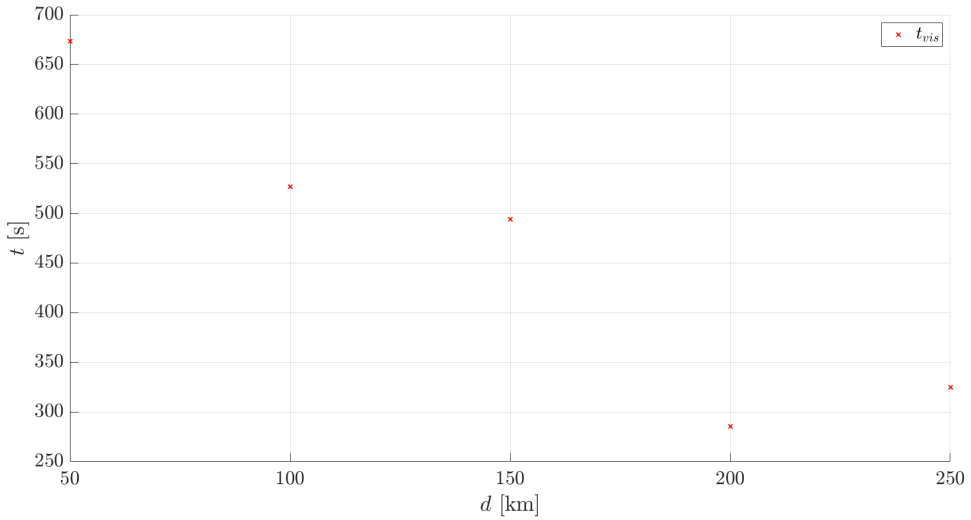

Figure 4.5: Dependance of the observation time with the distance between the observer and the RSO

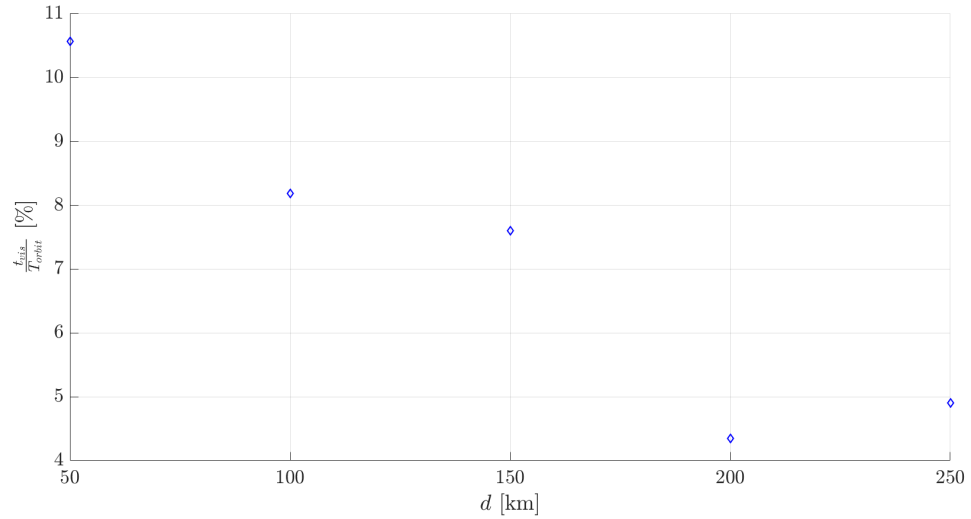

Figure 4.6: Relation between the percentage of orbit visible and the distance between the observer and the RSO