POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Light Curve Inversion for Attitude Reconstruction of Tumbling Space Debris

Laurea Magistrale in Space Engineering - Ingegneria Spaziale

Author: Matteo Gallucci

Advisor: Prof. Mauro Massari

Co-advisors: PhD Candidate Riccardo Cipollone, PhD Candidate Andrea De Vittori, Research Fellow Giovanni Purpura

Academic year: 2020-2021

1. Introduction

The number of space debris, its combined mass and area has been steadily increasing since the beginning of the space age [1]. A worsening of this situation could lead to an overcrowding of the Low Earth Orbit (LEO) regime, triggering a collision chain that would hinder the safety of space missions. The monitoring and tracking of resident space objects, performed by the space agencies SST (Space Surveillance and Tracking) segment, is of fundamental importance. Particularly, direct observation of a debris through the collection of the sunlight reflected from its surface, allows to record in a short video or in a long-exposure image the variably-illuminated trail left by its motion (streak or tracklet). From the extraction of light intensity over time (light curve), angular and astrometric information can be retrieved and used to determine the orbit and the attitude state of the object. Information about the latter would hugely benefit the predictions on how the debris will reenter the atmosphere and the different active debris removal (ADR) techniques.
The main methods used to determine the space debris attitude state inherit most of their features from studies on light curves of asteroids and variable stars. These techniques are primarily able to determine the rotational period of the object (period finding algorithms), while only in few particular cases they are able to fully characterise the overall attitude state (epoch method). Conversely, a full state reconstruction is granted by light curve inversion methods [2]. In these, a fit of the real light curve with a simulated one is performed taking as input the attitude state guess; then, the difference between real and simulated signal is minimized over attitude states.
In this thesis, a variation of light curve inversion methods is employed to determine the space debris attitude state. In particular, since the overall work is intended to be applicable in the first stages of the study of the motion of an unknown object, the light curve simulation is performed according to simplified shape and reflection model. In addition, both real and simulated FITS (Flexible Image Transport System) images are considered.

2. Fundamentals

The space debris shape is simplified and approximated as a flat square plate. In this way, the

body frame (B) rotation with respect to an inertial reference frame (Earth-Centered Inertial, ECI) can be represented by the torque-free Euler's rotation Equations (EE):

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \mathbf{I}\boldsymbol{\omega} \times \boldsymbol{\omega} \tag{1}$$

Where $\boldsymbol{\omega}$ is the vector of angular velocities (ω_x, ω_y and ω_z) and $\mathbf{I}$ is the diagonal inertia matrix.
In addition, the orientation of B with respect to ECI can be represented through the yaw (ϕ), pitch (θ), and roll (ψ) angles. They identify three consecutive rotations with a 321 sequence that is condensed in a rotation matrix ($\mathbf{A_{B/N}}$). The change in time of $\mathbf{A_{B/N}}$, in relation to the rotational dynamics in B (kinematics), is expressed by:

$$\dot{\mathbf{A}}_{\mathbf{B/N}} = -\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \mathbf{A_{B/N}} \tag{2}$$

In order to retrieve the kinematics and dynamics time evolution, Eq. 2 can be integrated alongside EE with specified initial conditions.
The target object reflects specularly the sunlight. The reflection is influenced by the relative positions between the Sun, the observer and the debris (see Figure 1), that can be identified with two versors:

- Observer-Debris versor ($\hat{\mathbf{O}}$): identified by the debris celestial coordinates (Right-Ascension (RA) and Declination (DEC))
- Debris-Sun versor ($\hat{\mathbf{S}}$): Sun position in ECI identified by its ephemeris

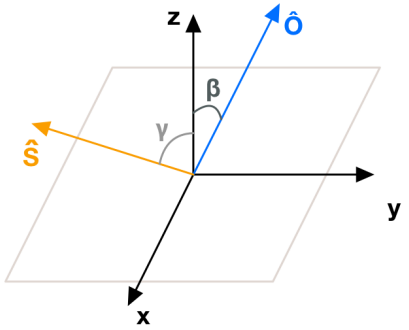


Figure 1: Reflecting Model

The plate area seen by the observer is given by the product of the simply projected (A_{proj}) and the simply illuminated (A_{ill}) areas, and it is influenced by the dynamics of the object. Therefore, the area variation expression is given by:

$$A_{var} = (\hat{\mathbf{O}} \cdot \hat{\mathbf{N}})(\hat{\mathbf{S}} \cdot \hat{\mathbf{N}}) = \cos\beta \cos\gamma \tag{3}$$

Where $\hat{\mathbf{N}}$ is the normal to the face of the flat plate considered (either $+z$ or $-z$). The area variation is different from zero only if it is illuminated ($\gamma < 90°$) and visible ($\beta < 90°$) at the same time. Equation 3 identifies a light curve, since it is a scaled representation of how the light intensity reflected by the object changes in time (due to its dynamics). In fact, it is possible to link A_{var} to a pixel intensity - L_{var} - by introducing a shift (n) and a multiplicative factor (k), so that:

$$L_{var} = A_{var} \cdot k + n \tag{4}$$

If n is chosen as the mean pixel value, whenever $A_{var} = 0$, the streak is blended with the background and correctly non-visible.
In this way, the light curve is considered as a pixel intensity variation in time. A more classical approach is to consider the streak apparent magnitude (M) variation in time, that can be computed as:

$$M = m_{ref} + I_m - I_{m_{ref}} \tag{5}$$

To evaluate it, the reference magnitude (m_{ref}) of a control star in the field of view (FOV) and the streak and control star instrumental magnitudes (I_m and $I_{m_{ref}}$, respectively) are needed. For the former, star features can be queried from astronomical catalogues and matched with the stars in the image. The knowledge of the latter instead implies the study through aperture photometry of the flux received from the sources in the field [3]. This second technique is employed whenever the considered streak is saturating the detector pixels, thus losing the link between incoming radiation and pixel counts.
Once the real light curve is extracted - whether in terms of light intensity or magnitude -, it is fitted with the simulated one by minimizing the difference between the two through genetic algorithm (GA). This algorithm mimics the Darwinian theory of survival of the fittest in nature, and is an iterative process in which the solutions are represented by individuals collected in a population. GA performs different operations on this population (selection, crossover,

and mutation), and evaluates the objective function value (fitness score) for the solutions at each generation. The worse fit solutions are discarded and the process stops whenever the fitness scores rate of change is below a certain threshold ($FunTol$) or a maximum number of generations ($MaxGen$) is reached.

3. Angular Velocity Extraction Process

The proposed approach to determine the attitude state of the space debris requires three main computation steps:

1. Light curve extraction and post-processing
2. Light curve simulation through the dynamics and kinematics of a square flat plate
3. Optimization of the objective function, minimizing the difference between the two light curves

In order to validate the developed code, synthetic images are generated and analyzed.
The first step for the generation of such images is the plate dynamics simulation, following the main physical concepts exposed in Sec. 2. The simulation, developed in MATLAB programming environment, is carried out by a function (`dynamics.m`) that takes as input an array ($\mathbf{var_{input}}$) containing:

- Three initial yaw-pitch-roll angles: $\mathbf{angles_0}$
- Three initial angular velocities: $\boldsymbol{\omega_0}$

The output is the illuminated-projected area variation array ($\mathbf{c_{tot}}$) that is fed as a *.txt* file to TIG (Tracklet Image Generator [4]). TIG is a Python application that takes as input night sky acquisitions and outputs a faithful representation of the object tracklet, according to its dynamics. For the purpose of this thesis, some modifications are done to the program in order to integrate it with the other portions of the code, while its main structure is left untouched. In this new version, the inputs to TIG are:

- One or more night sky FITS images
- The *.txt* file of the area variation
- Another *.txt* file containing the Azimuth (Az) and Elevation (El) evolution for a given tracked object

According to this last file, TIG identifies the number of pixels needed to represent the given passage of the object, and places the streak selecting the pixel coordinates in the image. The colouring of the tracklet is performed by substituting these central pixel values with $\mathbf{c_{tot}}$ scaled values, according to Eq. 4, with the introduction of some noise. A certain thickness is assigned to the streak and the colouring is performed also for the pixels around the central line, with decreasing intensity. In this new version, TIG can represent saturated or partially saturated streaks. Saturation entails an increase in the smearing of the streak itself due to the bleeding effect. Furthermore, the pixel values must be clipped to the maximum value (saturation threshold, sat) according to the input sky image.
Now, either real or synthetic FITS images are fed to SAP (Streak Analysis Pipeline), a novel Python application developed for the extraction and analysis of streaks. SAP, exploiting several Python astronomical packages, performs the following main tasks:

1. Detection and extraction of the streak(s) from the input FITS file
2. Extraction of central line luminosity variation (light curve in terms of light intensity)
3. Evaluation of the streak thickness
4. Source extraction with DAOPHOT algorithm, control star (lowest magnitude) identification, and circular aperture photometry
5. Catalogue query and catalogue matching with a cone search within the image FOV
6. Rectangular aperture photometry on the streak; each aperture is one pixel wide and with length equal to the maximum identified streak thickness. The light curve is extracted in terms of apparent magnitude variation in time

Tasks from 4 to 6 are performed in the case of saturated streaks. For each run on an image, SAP produces three HTML diagnostic pages to track how the application worked. Moreover, it outputs:

- Two tables containing useful information about the streak RA and DEC evolution in time and the observer location and time
- A *.txt* file featuring the real light curve evolution (either central line luminosity or apparent magnitude).

These outputs are used by the last part of the code, i.e. the optimization solver. The GA MATLAB implementation (`ga.m`) is used to fit the extracted real light curve with the simulated one, that is generated by scaling with Eq. 4 the dynamics simulator output $\mathbf{c_{tot}}$. In particu-

lar, both light curves are scaled with respect to *sat* (range between 0 and 1) in order to have a faster optimization; then, an objective function (`obj_fun.m`) sums all the elements of the absolute value of the difference between the two. In this way, a scalar *cost* measure is obtained, and its value is minimized by GA over the selected design variables, that are:

- Attitude initial conditions: three initial angular velocities (ω_{0x}, ω_{0y}, and ω_{0z}) and three initial yaw-pitch-roll angles (ϕ_0, θ_0, and ψ_0, cf. Sec. 2), so that only six parameters are needed to simulate the dynamics and input in `dynamics.m`, as $\mathbf{var_{input}}$.
- Scaling parameters: only multiplicative factor k, since n is taken as the mean pixel value of the image coming from SAP.

The lower and upper boundaries for the design variables range are defined between 0° and 360° for the angles, between 0 and 1 for k, and between $-\omega_{lim}$ and ω_{lim} for the angular velocities. The value of ω_{lim} is determined by performing a Fourier series expansion of the real light curve up to the second order and evaluating its frequency. For the fitting of saturated streak light curves a different scaling is carried out; no shift is considered and the simulated light curve is built as $\mathbf{mag_{sim}} = 1 - \mathbf{c_{tot}} \cdot k$. As a consequence, the scaling of the real one is performed with respect to $(m_{ref} - I_{m_{ref}})$.

4. Results

The overall procedure of angular velocity extraction is applied on both synthetic and real images. In the realization of synthetic images, different rotational regimes of space debris are considered, in order to account for different real case scenarios [5]. In particular, three different regimes are arbitrarily defined:

- *Slow Rotation*: maximum angular velocity below 10 $^\circ/s$
- *Medium Rotation*: maximum angular velocity below 90 $^\circ/s$
- *Fast Rotation*: maximum angular velocity below 180 $^\circ/s$

Moreover, both non-saturated and saturated streaks are analyzed and the overall goodness of the different methods is compared.
The real images analyzed are taken from the Pratica di Mare ground station through a 350mm-optical telescope (PdM-MITE, Pratica di Mare - MIlitary TElescope) operated by personnel of Aero-Space System Engineering Group of Flight Test Wing [6].
Concerning the machine on which the tests were performed, it is a PC with an Intel i7-6500U CPU@2.5 GHz with a 16 GB RAM.

Synthetic Images

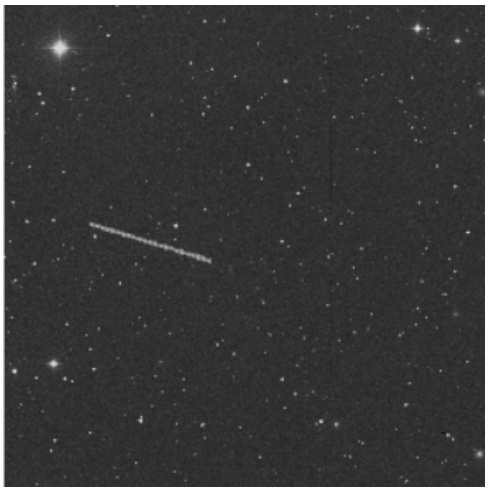


Figure 2: Example of a medium-rotating debris synthetic image.

The main distinction between the generated synthetic images is given by the attitude initial conditions and the multiplicative factor k.
For each rotational regime, among the various synthetic images that were generated, a representative one is selected (see Figure 2). The results show the effectiveness of the method and the variability between different runs on the same image. Moreover, the differences between rotational regimes are outlined. In general, a key role is also played by the simulation time, since the solver speed is penalized when it increases.
For each rotational regime, the initial conditions for the dynamics simulation and the multiplicative factor k are summarized in Table 1, along with the results obtained through the optimization for a given run. In addition, the same values for a slow-rotating saturated streak are presented.
The results show that:

1. The angular velocity norm ($\|\boldsymbol{\omega_0}\|$) is accurately retrieved in all the rotational regimes. The accuracy decreases with the increasing of the rotations and in general for saturated streaks.

2. The angular velocity z-component is the best fitted one. The in-plane components are sometimes inverted. Thanks to the symmetry of the plate, this is not considered an issue.
3. The attitude angles are usually not properly fitted. This is mainly due to the presence of multiple solutions related to the light curve fitting, and of the noise introduced in TIG.
4. The k value is optimally fitted in most cases.

	Slow		Medium		Fast	
	Sim	Opt	Sim	Opt	Sim	Opt
ω_{0x} [°/s]	0.1 5	0.15 2.75	25	9.5	81	76.47
ω_{0y} [°/s]	1.5 2	1.4 -2.3	11	25.6	50	57.44
ω_{0z} [°/s]	4 8	4.03 -8.3	50	50.1	120	120.1
$\|\omega_0\|$ [°/s]	4.27 9.64	4.26 9.03	56.9	57.1	153.2	153.6
ϕ_0 [°]	240 0	106 281	0	124	200	191
θ_0 [°]	245 80	351 272	0	43	15	20
ψ_0 [°]	165 200	341 89	0	8	45	42
k [-]	0.153	0.120	0.229	0.226	0.183	0.182

Table 1: Synthetic Images - Results: for the slow-rotating regime the results are presented for both non saturated (top) and saturated (bottom) streaks. For the latter no optimization of k is considered.

The results can be characterized in terms of the relative error for the angular velocity norm for each run (err):

$$err = \left| \frac{\|\omega_0^{sim}\| - \|\omega_0^{opt}\|}{\|\omega_0^{sim}\|} \right| \tag{6}$$

In order to obtain a statistical relevance of this parameter, the optimization with `ga.m` is carried out multiple times for the selected image. Then, the cumulative distribution function (CDF) of err values for each run is computed to evaluate the changes between different rotational regime cases.

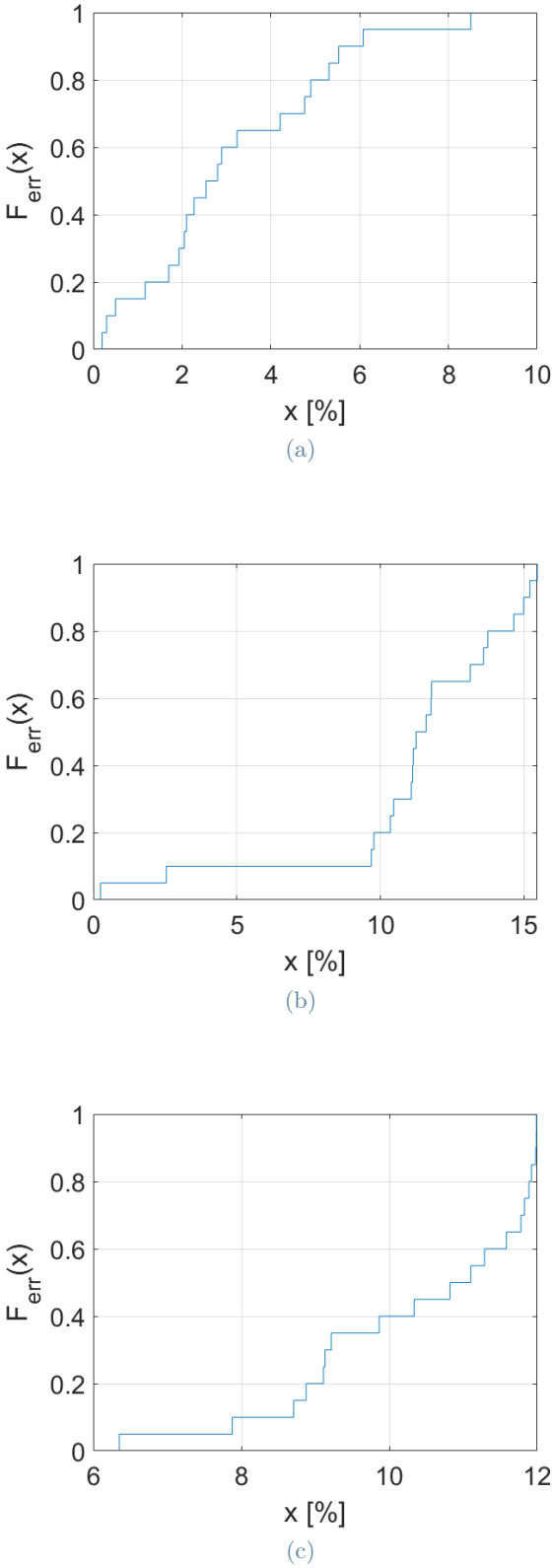


Figure 3: CDF of the relative errors. **(a)** CDF for slow rotation case. **(b)** CDF for fast rotation case. **(c)** CDF for saturated case.

As it can be seen in Figure 3, the steepness of the CDF decreases with faster rotations; this implies

a worsening of the performances, since the probability of having higher *err* values is increased from slow to fast rotational regimes. Furthermore, it can be noted a similar accuracy penalization for saturated streaks in terms of CDF.

Real Images

Figure 4: Real image from the PdM-MITE 2017 observation campaign. The rendering of the image is in logarithmic scale in order to better visualize the faint streak.

In the real images case, the results of the optimization can be directly shown and the overall process is lighter in terms of tasks to be performed. The analyzed images are coming from a series of acquisitions performed at the Pratica di Mare ground station during an observation campaign in September 2017.
As it can be seen in Figure 4, the image acquired by PdM-MITE contains a faint streak left by a debris identified as the COSMO SkyMed 2, an ASI (Agenzia Spaziale Italiana) artifical LEO satellite launched in 2007.
The curve extracted by SAP is quite noisy. Therefore, a filtering of the signal is performed, as well as a re-sampling in order to reduce the dimensions of the luminosity array.

ω_{0x} [°/s]	ω_{0y} [°/s]	ω_{0z} [°/s]	$\|\omega_0\|$ [°/s]
-11.5	2.8	149	149.47

Table 2: Real Image - Results

The results in terms of angular velocity (see Table 2) are rather consistent with a fast spinning object in LEO and, as seen before for fast rotating debris, the solver performance decreases with the increased value of angular velocity both in terms of prediction accuracy and of optimization speed. Nevertheless, for the identified object, the value of ω_{0z} seems quite high. The short-evolution dynamics does not allow a more precise estimate.
The optimization is performed for 100 times on the same image to evaluate the variability of the results. To achieve so, a new relative error is defined as:

$$err = \left| \frac{\|\boldsymbol{\omega_0}\| - \mu}{\mu} \right| \tag{7}$$

Where $\mu = 140.01°/s$ is the statistical mean of the angular velocities norm. Then, it is possible to evaluate the CDF of *err* and consider the probability of having a $\|\boldsymbol{\omega_0}\|$ that differs more than 1σ (standard deviation, $\sigma = 31.58°/s$) from μ.

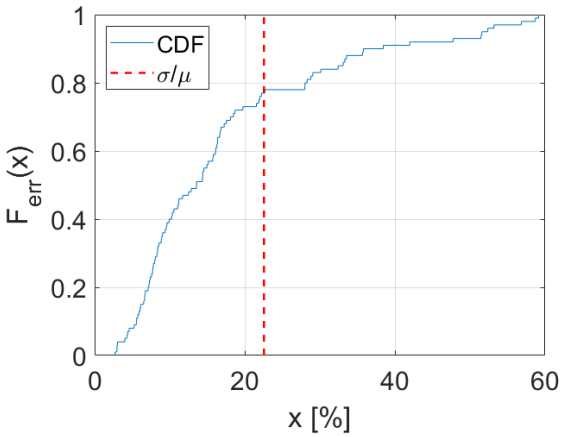


Figure 5: CDF of the relative error for real streaks. The CDF is quite steep, even if *err* is not really contained. Nevertheless almost 80% of the results are within 1σ of μ.

The results for 100 runs, as it can be seen in Figure 5, are quite variable. This is mainly due to the short evolution of the extracted light curve that may have different optimal fittings and to the high rotational speeds, leading to worse results

5. Conclusions

The attitude determination of a tumbling space debris is a quite difficult challenge. In this thesis, a light curve inversion for attitude reconstruction is carried out on both real and syn-

thetic images. The generation of synthetic images, performed using TIG and the dynamics simulator, outputs FITS files that resemble the characteristics of a real streak left by a tumbling debris. Moreover, SAP successfully extracts, analyzes, and post-processes the streak present in any input FITS image. The optimization part of the code is able to identify different rotational regimes of the tumbling object and conduct the analysis accordingly. The performance of the optimization generally decreases with the increase of the rotation rate (fast rotating debris) and if the streak is saturated. The analysis proved to be quite successful in the determination of the norm of the angular velocity vector and, in some cases, of its components. The results for real images containing tumbling space debris are quite variable and their accuracy should be validated. More complex models could be used for the reflection and shaping of the debris when a refined analysis is required.

6. Acknowledgements

First of all, I wish to show my deepest gratitude to my advisor professor Mauro Massari, who followed me patiently during the development of this work. I would also like to thank the Italian Air Force for the precious images from the Pratica di Mare ground station used in the thesis. Moreover, I would like to pay my special regards to Andrea De Vittori, Riccardo Cipollone and Giovanni Purpura, who followed and helped me in the writing of the code, and for their precious advice about the drafting of the thesis and, generally, for their extreme availability.

References

[1] ESA Space Debris Office. ESA's Annual Space Environment Report, 27 May 2021. `https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf`.

[2] F. Piergentili, G. Zarcone, L. Parisi, L. Mariani, S. H. Hossein, and F. Santoni. LEO Object's Light-Curve Acquisition System and Their Inversion for Attitude Reconstruction. *Aerospace*, 8(1), 2021.

[3] W. Romanishin. An Introduction to Astronomical Photometry Using CCDs. `https://www1.phys.vt.edu/~jhs/phys3154/CCDPhotometryBook.pdf`, March 2002.

[4] R. Cipollone and A. De Vittori. Machine Learning Techniques for Optical and Multibeam Radar Track Reconstruction of LEO Objects. Master's thesis, Politecinco di Milano, School of Industrial and Information Engineering, Department of Aerospace Science and Technology, Milan, Italy, 2020.

[5] J. Šilha, J-N. Pittet, M. Hamara, and T. Schildknecht. Apparent rotation properties of space debris extracted from photometric measurements. *Advances in Space Research*, 61(3):844–861, 2018.

[6] G. M. D. Genio, J. Paoli, E. D. Grande, F. Dolce. Italian Air Force Radar and Optical Sensor Experiments for the Detection of Space Objects in LEO Orbit.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Light Curve Inversion for Attitude Reconstruction of Tumbling Space Debris

Tesi di Laurea Magistrale in
Space Engineering - Ingegneria Spaziale

Author: **Matteo Gallucci**

Student ID: 944399
Advisor: Prof. Mauro Massari
Co-advisors: PhD Candidate Riccardo Cipollone, PhD Candidate Andrea De Vittori, Research Fellow Giovanni Purpura
Academic Year: 2020-21

Abstract

In recent years, the problem of space pollution has become more and more important in the space field. Monitoring and tracking of the space debris population is fundamental in order to keep space accessible for human operations. In particular, observations through optical telescopes allow space agencies to directly observe the orbiting object and study the trail (streak or tracklet) it leaves on the detector. From these streaks, it is possible to extract in post-processing the variation of the light reflected by the object in time (light curve). From light curves, the key features of the space debris such as its orbital parameters and attitude state can be retrieved. Particularly, the information about the latter is of fundamental importance when it comes to actively remove the debris or to predict how it will behave when/if it will reenter the atmosphere.

Most of the techniques used for determining the attitude motion of a space object rely on methods developed for the determination of the rotational period of asteroids and variable stars (period finding algorithms). More complex methods, such as light curve inversion, allow to reconstruct the attitude state by simulating the dynamics of a geometry-defined object and the way it reflects the light coming from the Sun.

In this thesis, a variation of the latter is used to extract the angular velocity of a generic tumbling object, that is approximated by a square flat plate. Its dynamics evolution is employed to build synthetic light curves through a simple reflection model. They are used to fit the real ones extracted from FITS images, through an optimization process that outputs the attitude state of the object itself. Moreover, synthetic images containing streaks are generated to validate the code using the same simulation process.

Considering the results, the code correctly extracts and analyzes streaks in real and synthetic images. The target object angular velocity, both in norm and components, is accurately determined for moderate rotational periods. With faster rotations, the results validity slightly decreases, but it is still acceptable for the preliminary analysis of the space debris tumbling motion.

Keywords: space debris, optical telescope, light curve inversion, attitude motion, FITS.

Abstract in lingua italiana

Negli ultimi anni il problema dell'inquinamento dello spazio è diventato sempre più importante nell'ambito spaziale. Il monitoraggio della popolazione di detriti spaziali è fondamentale affinché lo spazio rimanga accessibile per le operazioni umane. In particolare, le osservazioni tramite telescopi ottici permettono alle agenzie spaziali di osservare direttamente gli oggetti in orbita e di studiare le scie (streak) che lasciano sulle camere. A partire da queste streak è possibile estrarre la variazione di luce riflessa dall'oggetto nel tempo (curva di luce). Dalle curve di luce si può risalire alle caratteristiche principali dei detriti spaziali, come i loro parametri orbitali o lo stato attitudinale. Quest'ultimo risulta particolarmente d'aiuto nella rimozione attiva di detriti spaziali o quando è necessario predire il comportamento del detrito al rientro nell'atmosfera terrestre.

La maggior parte delle tecniche utilizzate per determinare il moto attitudinale di un oggetto spaziale si basa su metodi sviluppati per la determinazione del periodo rotazionale di asteroidi e stelle ad intensità variabile (algoritmi period finding). Metodi più complessi, come l'inversione della curva di luce, consentono di ricostruire lo stato attitudinale grazie alla simulazione della dinamica di un oggetto a geometria ben definita che riflette la luce proveniente dal sole.

In questa tesi, una variazione di quest'ultimo metodo viene usata per estrarre la velocità angolare di un generico oggetto rotante, approssimato da una piastra piana quadrata. La dinamica di quest'ultima viene impiegata per costruire curve di luce sintetiche attraverso un semplice modello di riflessione. Queste sono utilizzate per fittare curve di luce reali estratte da immagini FITS, attraverso un processo di ottimizzazione che fornisce in output lo stato attitudinale dell'oggetto stesso. Inoltre, vengono generate attraverso il medesimo processo di simulazione immagini sintetiche contenenti streak per validare il codice.

Per quanto riguarda i risultati, il codice estrae ed analizza in maniera corretta streak in immagini FITS reali e sintetiche. La velocità angolare del target, sia in norma, sia in componenti, è determinata accuratamente per periodi di rotazione moderati. Con rotazioni più veloci, la validità dei risultati dimunuisce leggermente, ma rimane comunque accettabile per l'analisi preliminare del moto attitudinale del detrito spaziale.

Parole chiave: detriti spaziali, telescopi ottici, inversione curve di luce, moto attitudinale, FITS.

Contents

1 | Introduction

1.1. Space Debris

As of September 2021, more than 331 million space debris objects, ranging from 1 mm up to sizes larger than 10 cm, are orbiting the Earth [1]. In general, the number of objects, their combined mass and area have been steadily increasing since the beginning of the space age, as it can be seen in Figure 1.1.

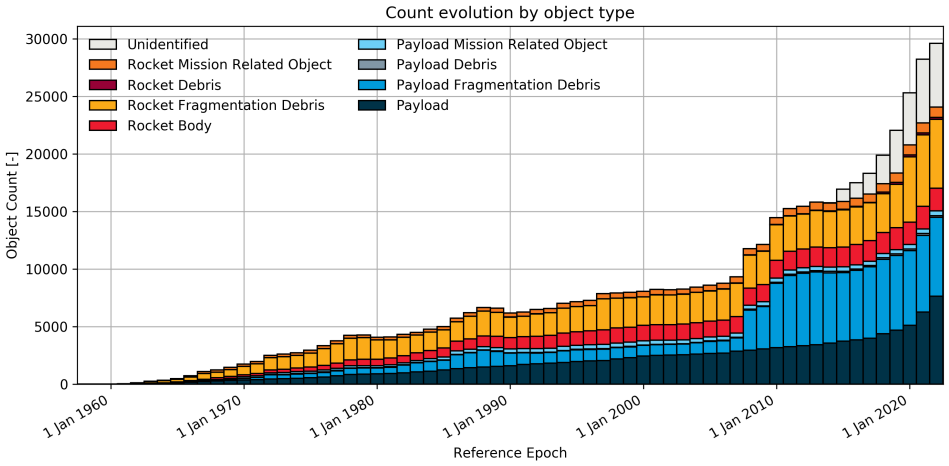


Figure 1.1: Evolution of number of objects [2]

Moreover, in recent years, commercial satellite launches, usually deploying big constellations, have become fairly common (see Figure 1.2), posing a great challenge to space sustainability.

It is also worth noting that a vast portion of space debris is composed by small fragments generated by on-orbit events (fragmentations), such as collisions, explosive break-ups, and tear and wear. The number of fragments continues to rise also due to deliberate actions such as anti-satellite tests[1].

[1]The Chinese FengYun-1C test in 2007 alone increased the space object population by 25% [3]

This situation could lead to an overcrowding of the Low Earth Orbit (LEO) regime, triggering a collision chain that would hinder the safety of space missions (Kessler Effect, [4]).

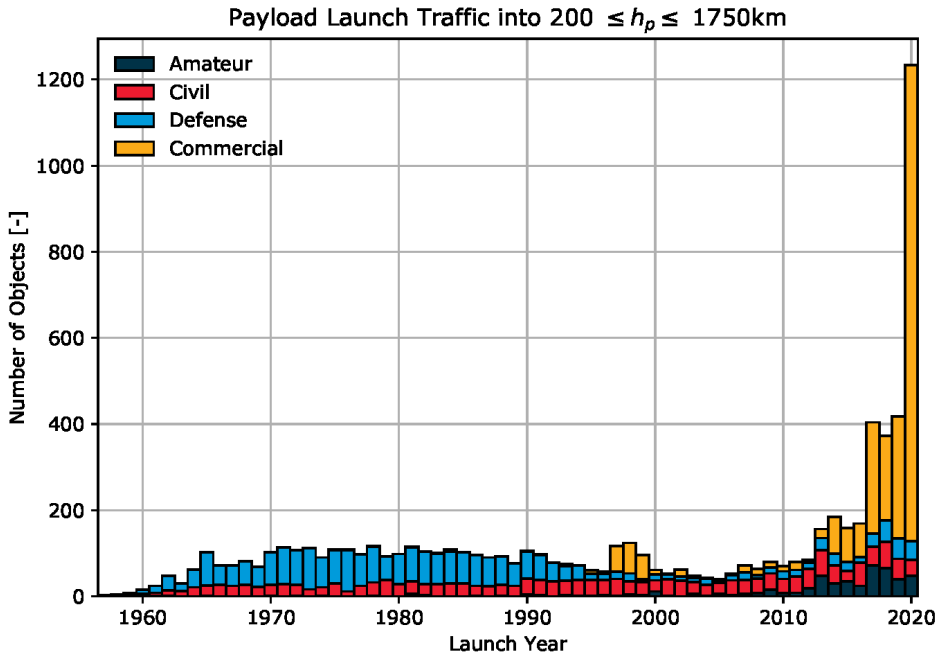


Figure 1.2: Evolution of launch traffic near LEO per mission funding [2]

A first broad classification of space debris can be done in terms of the possibility to trace back the object to a launch event (Identified vs. Unidentified (UI)) [2].
Identified objects can be further divided according to their purpose:

- Payload (PL[2]) and Payload related debris (PX): space objects designed to perform a specific function in space excluding launch that are now non-operative, all related debris such as Payload mission related objects (PM), Payload fragmentation debris (PF) and Payload debris (PD).
- Rocket body (RB) and Rocket body related debris (RX): space objects designed to perform launch that are now spent orbital stages, all related debris such as Rocket mission related object (RM), Rocket fragmentation debris (RFr) and Rocket debris (RD).

Furthermore, Identified objects can also be differentiated in catalogued and asserted ones.

[2]The acronyms refer to the nomenclature used in [2]

Usually, the former has its orbital elements maintained for a prolonged period of time. For the latter no report by a space surveillance system has been done yet, but the object is known to exist in the space environment by design (e.g. rocket bodies performing a reentry burn after inserting a payload into orbit).
In general, for catalogued objects, orbital information can be retrieved from different datasets, whereas for asserted ones these are usually taken from different databases such as DISCOS (Database and Information System Characterising Objects in Space) of ESA [2]. On the other hand, for attitude state information and any estimate related to UI objects, independent analyses have to be carried out.
Therefore, to keep space pollution under control and determine whether an object will safely reenter in the atmosphere or collide with other active or inactive objects, space agencies perform a set of operations that falls under the Space Surveillance and Tracking (SST) segment. SST has the main goal of ensuring safe space operations, by assessing the risk of collision for each of its catalogued objects [5].

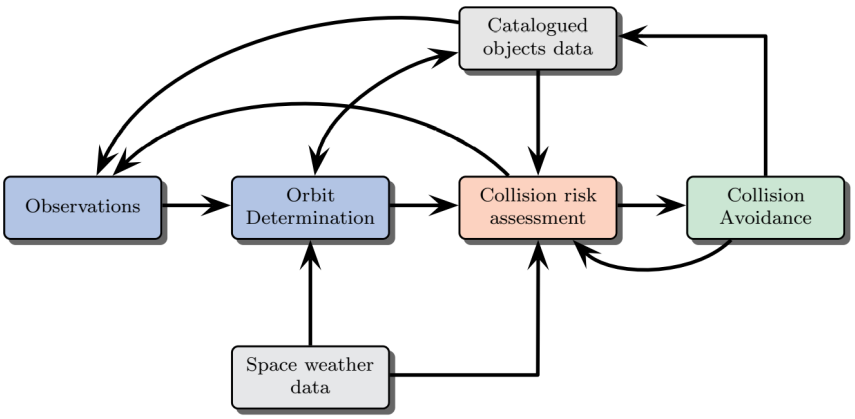


Figure 1.3: SST Operations [6]

In Figure 1.3, a scheme of some activities related to the monitoring and tracking of space debris is presented. As it can be seen, most of the activities are intertwined and/or consequential. Undoubtedly, one of the most important tasks is object observation starting from catalogue information, allowing accurate orbit determination and the estimation of other features such as the tumbling motion.
Overall, there are many different ways to observe and monitor the space debris population [7], with regard to its size, mass, area and distribution:

1. Passive optical: direct observation of the object through the collection of the sunlight reflected from its surface. The data is acquired thanks to optical telescopes (see Sec. 2.2), while several conditions need to be fulfilled: object illuminated by the Sun,

night observations, and clear sky conditions. The motion of the debris can be recorded in a video or in a long-exposure image (where the debris leaves a streak or tracklet of variably illuminated pixels according to the orbital and attitude motion, see Figure 1.4). From the extraction of light intensity over time (light curve, see Sec. 2.2), angular and astrometric information can be retrieved and used to determine the orbit and the attitude state of the object. In addition, its surface properties and shape can be studied through multi-band spectroscopy.

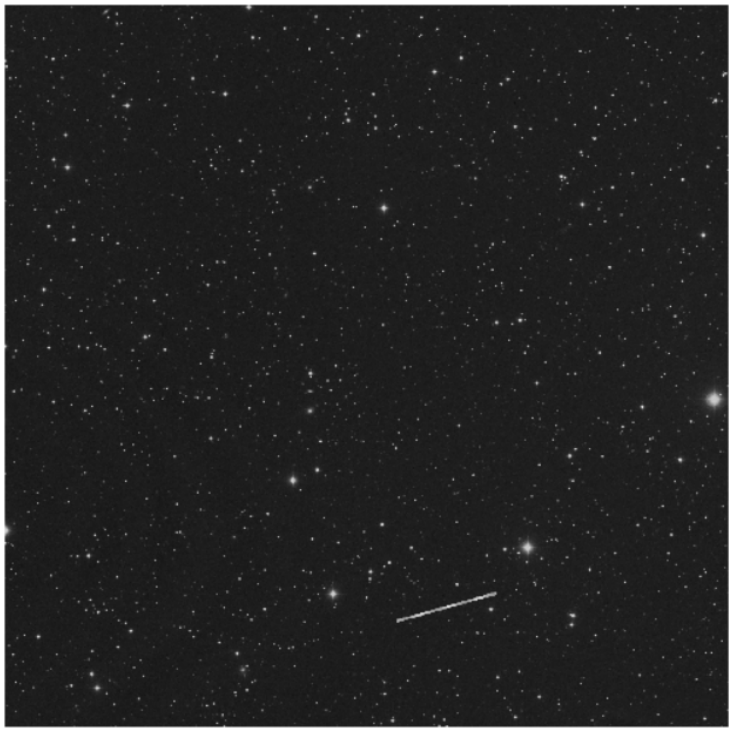


Figure 1.4: Example of a simulated streak left by a space debris

2. Active optical: acquisition of range and angular measurements through Satellite Laser Ranging (SLR) systems. Ultra-short light pulses are emitted by a ground station towards a resident space object and reflected back. Depending on the light time of flight, the target range and attitude can be inferred with a high degree of accuracy. This kind of systems is still influenced by weather conditions, while is partially affected by environmental illumination conditions. In general, high power lasers are required for the majority of the targets.

3. Radar: radar measurements are not influenced by weather conditions and can be performed during the whole day. Through the Radar Cross Section (RCS) parameter - i.e. target ability to reflect radar energy -, the target size, shape, orientation and material properties can be investigated

4. In-situ: exposure of a satellite to the space environment to test various materials and their suitability for space missions, and study the effects of radiation and other space hazard. These measurements are mainly used to model the smaller (less than 1 mm) debris population.

In the mid 1980s space agencies started to deal with space pollution, but it wasn't until the 1990s that mitigation measures were put in place. Nowadays, the main measures followed by most of international parties are focused on the limitation of space debris release, collision avoidance manoeuvres and post-mission disposal regulations [2].
Long-term evolution models have shown that mitigation measures can reduce the growth of space debris generation, however long-term proliferation is still expected, even with full mitigation compliance and even if all launch activities were to be halted [8]. Therefore, Active Debris Removal (ADR) techniques have been studied and employed in recent years in order to further alleviate the problem.
In particular, ADR focuses on large and massive objects at high altitudes that may reach critical densities in LEO and have the largest environmental impact in case of collision, since the resulting fragments would have a long lifetime. Moreover, many of the different ADR techniques would hugely benefit from prior knowledge of the attitude state of the object [9].
Hence, a combination of mitigation and active removal, applied as soon as possible, is thought to be the most effective strategy to keep the space environment sustainable and accessible.

1.2. State of the Art and Proposed Approach

As seen in Sec. 1.1, one of the most effective ways to retrieve information about a space debris is through observation campaigns at night and the analysis of the obtained light curve.
In the following section, the main methods used to determine the angular velocity of space debris will be presented. In general, the discipline inherits most of its features from studies on light curves of asteroids [10] and variable stars [11]. In particular, light curve inversion techniques (determination of attitude state from a light curve measurement) for asteroids allow to assess not only the angular velocity of the object, but also some of its physical features (shape and size) when working in different bands of the spectrum [12].
Nevertheless, most of the hereby presented techniques are able to determine mainly the rotational period of the object, while only in few particular cases they are able to fully characterise the overall attitude state.

In addition, the adopted method to extract the attitude state of the tumbling space object is outlined and the structure of the document is briefly described.

Period Finding Algorithms

There are numerous types of period finding algorithms that can be applied to light curves of celestial objects [13]. Since the scope of this thesis is to develop an alternative method to determine not only the period, but also the entire attitude state, the following techniques will be briefly presented in order to have a smattering of the problem:

i. Fourier-based methods: methods that exploit signal frequency decomposition in order to retrieve the period of the base sinusoidal function. In its most simple form, Fourier decomposition rewrites the periodic signal as a sum of sines and cosines characterized by a given period. This is the concept behind FFT (Fast Fourier Transform) method and Welch's Method. The former operates a Discrete Fourier Transform (DFT) of the signal converting it into the frequency domain. In the latter instead, the DFT is performed to different segments of the data called bins and the final result is averaged over the segments. These methods are mainly influenced by the light curve data quality rather than the real object physical properties. Moreover, more reliable results can be obtained if the acquisition time is at least six times the expected period.

ii. Epoch Folding method: similarly to Welch's method, this technique divides the overall observation time in bins; in this case though, the bin length is equal to a guessed period of the main peak identified in the signal. Then, each part of the light curve is summed and a new light curve, with length equal to one bin, is obtained (folding of the curve, see also Sec. 2.2): if no recognizable pattern is present, the hypothesized period is wrong, whereas if an exaggerated peak is present the period is correct. Evidently, this technique needs a first period guess that can be either set manually after visual observation of the light curve or derived with other methods beforehand (e.g. Fourier analysis). This means that epoch folding can be used for refinement of results found with other methods. This method requires an acquisition time longer than one and a half times the guessed period.

iii. Principal Component Analysis: this approach turns the original signal into a new set of uncorrelated variables, called principal components (PC), by means of an orthogonal transformation. The dimension of these PC is usually much lower than the original signal one, and the period is usually retrieved through interpolation of the curve *PC vs phase*. Inevitably, some information may be lost due to the

performed trasnformation.

iv. Lomb-Scargle method: this technique falls under the least-squares spectral analysis methods, where the frequency is obtained through a least-square fit of the data. Generally, the signal is represented by a weighted sum of sinusoids; in this case, also a time delay is introduced in order to have mutually orthogonal sinusoids. This method could lead to inconclusive results that should be cross-checked with other methods.

In Table 1.1, a summary of the advantages and disadvantages of each method is presented.

Method	Advantages	Disadvantages
Fourier-based	Readily available results	Limited resolution and need of equally spaced-data
Epoch Folding	No need of equally spaced data	Initial value required
PC	Lower dimensional problem	Loss of information
Lomb-Scargle	Reliable results, ok with limited resolution data	Not always conclusive results, need of cross-checking

Table 1.1: Period Finding Algorithms Comparison

Epoch Method

The epoch method is vastly described and used for the attitude determination of asteroids and tumbling satellites [14]. In particular, it allows to determine, under certain conditions, the object angular velocities and the spin axis orientation. The key quantity that needs to be accounted for is the phase angle bisector (PAB) (see Figure 1.5), i.e the vector that bisects the Sun-object-observer angle.
For the orbiting spinning object, two spin periods can be identified: the sidereal one (time it takes for the object to complete one rotation with respect to the stars) and the synodic one (interval of time to complete one rotation with respect to the observer). The link between these two different quantities is the PAB time variation; in particular, the synodic frequency is written as a function of the sidereal one and the PAB projection on the plane orthogonal to the spin-axis of the object. Then, by estimating the synodic frequency through Fourier analysis, the unknowns of the problem become the sidereal frequency and the PAB variation that is written as a function of the spin axis orientation. Consequently, a least-squares fitting of the real synodic frequency is performed and the unknowns become the outputs of the process.

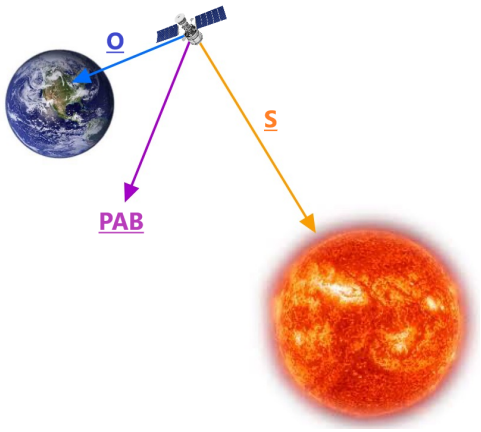


Figure 1.5: Phase-Angle-Bisector

As it will be seen in the next section, this kind of process does not differ much from a standard light curve inversion. Nevertheless, the epoch method works properly only under certain conditions, such as: constant rotation axis direction and rotational period, cylindrical bodies rotating about the major inertia axis, only specular glints detection.

Light Curve Inversion for Attitude Reconstruction

As detailed in previous sections, it appears evident that attitude determination from the object light curve is hard to deal with. In fact, many peculiar quantities related to the object motion have to be retrieved from the only periodic variation of light intensity.
In order to overcome such difficulties, light curve inversion (see Figure 1.6) can be used to reconstruct the attitude state. This is done through a fit of the real light curve with a simulated one taking as input the attitude state guess; then, the difference between real and simulated signal is minimized over attitude states [15].
Undoubtedly, one of the critical points is the generation of the simulated light curve: this is usually achieved using rendering programs ([15], [16]) that model the object under analysis together with the different illumination and shadow conditions. Another important part is played by the light curve extraction routine that needs to be properly designed according to the image to be processed. Finally, once the real light curve is extracted and the simulated one is retrieved by entering the object attitude state, a global optimization algorithm minimizes the difference between the two, providing the value of the unknown variable. In terms of non-linear optimization lots of algorithms have been developed and customized for different problems [17].

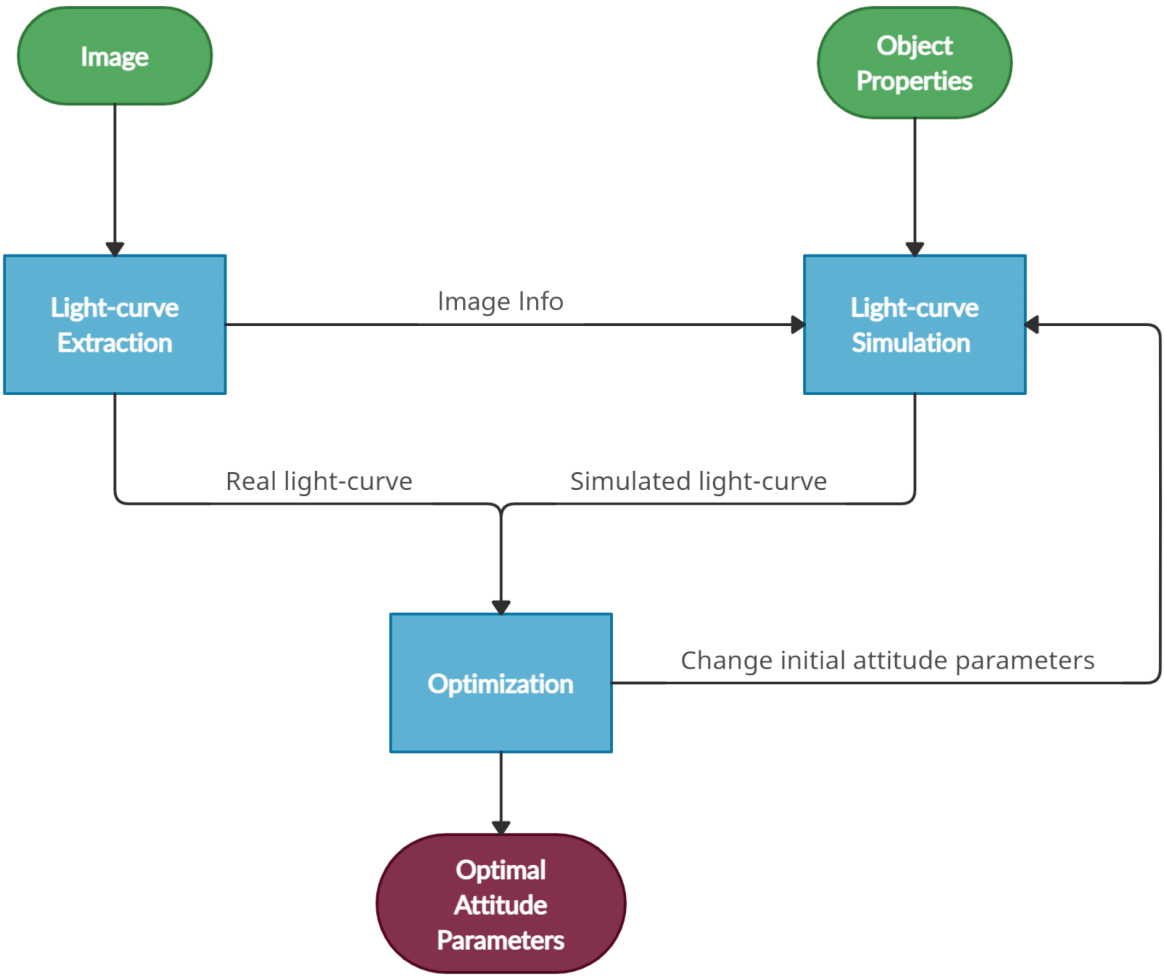


Figure 1.6: Light Curve Inversion Workflow

This kind of approach is adopted in this thesis, since it is considered to be the most suitable and tunable among the surveyed methods.

Thesis Objective and Workflow

The main objective of the following work is to determine the attitude state of a space debris from optical observations through the extraction and fitting of its light curve.
As stated in Sec. 1.2, the main methods used to extract the rotational period are rather limited in terms of the quality of the results and object orientation prediction. Conversely, the reference methods for light curve inversion are usually applied to specific targets, whose shape, size and other physical properties are known. Since the overall work is intended to be applicable in the first stages of the study of the motion of an unknown space object, i.e. when no information about the object shape, orbit or attitude

is available, a proper customization of the workflow has to be done (see Figure 1.7).

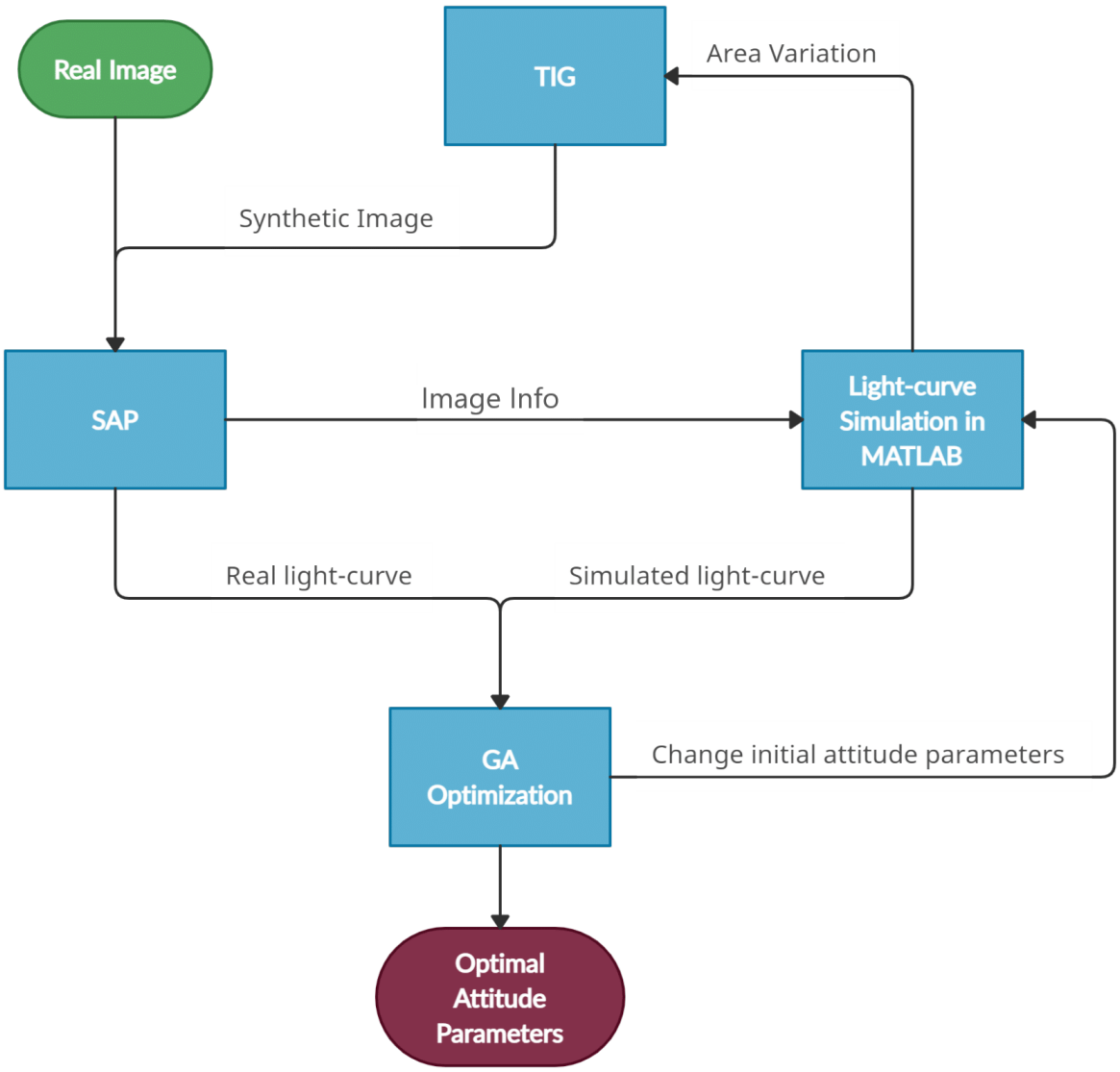


Figure 1.7: Thesis Workflow

First of all, given the lack of information on the object, it is not possible to use a rendering program, and the debris is approximated as a square flat plate that specularly reflects the sunlight (cf. Sec. 2.1); its dynamics is simulated in MATLAB (cf. Sec. 3.1).
Moreover, all the image processing and light curve extraction is performed by an original analysis pipeline (Streak Analysis Pipeline - SAP) written in Python (cf. Sec. 3.2), employing ad hoc packages for astronomical image manipulation.
Both real and simulated images are considered; in particular, the creation of synthetic images is done through a modified version of an image generator (Tracklet Image Generator

- TIG). This software tool, developed by R. Cipollone and A. De Vittori in [5], takes as input night sky acquisitions and the object tumbling motion information obtained with the developed MATLAB dynamics simulator; the output is a faithful representation of the object tracklet (cf. Sec. 3.1).
Furthermore, the choice of the optimization routine falls on the genetic algorithm (GA) in MATLAB environment (cf. Sec. 3.3), thanks to its extreme flexibility and to its unconventional way of reaching adequate solutions even for complex problems.
Finally, the optimization results both for real and synthetic images are presented (cf. Chap. 4), as well as a conclusion chapter (cf. Chap. 5), where the overall code effectiveness is summarized.

2 | Fundamentals

Before describing the detailed structure of the code (see Chap. 3), an extensive review of some key physical concepts has to be presented, in order to justify some of the choices that characterized the work.
Particularly, the different equations that describe 3D attitude dynamics, kinematics, and the selected reflection model are presented, as well as the transformations between different reference frames. Then, the working principle of optical telescopes and detectors is briefly explained. Moreover, an introduction to photometry and astrometry is carried out in order to understand the fundamentals behind image calibration and magnitude determination. Finally, the use, scope, and structure of FITS (Flexible Image Transport System) images are quickly reviewed, since this image file format has a broad use in astronomical applications, including this work.

2.1. Dynamics, Kinematics and Reflection Model

Reference Frames

A reference frame (RF) is a coordinate system with a specified origin and orientation and it is usually composed by a set of three mutually orthogonal axes.
A first conceptual distinction between RFs can be made whether the RF is undergoing an acceleration or a rotation; in the negative case, the RF is inertial, while if the RF is accelerating or rotating with respect to an inertial one it is called non-inertial.
In order to describe an attitude problem, a fixed inertial RF and a rotating RF can be identified. The former is the Earth-Centered Inertial (ECI) frame, while the latter is the Body (B) frame (Figure 2.1):

– ECI: origin in Earth's center, fixed with respect to the stars (inertial). It has the xy-plane that coincides with the Equator, with the x-axis in the same direction of the Vernal Equinox Line (intersection between equatorial and ecliptic planes). The z-axis is aligned with the Earth rotation axis. It is mainly used to describe the rotation of non-inertial frames, and to identify the inertial position of objects

through celestial coordinates: Right-Ascension (RA) and Declination (DEC).

– B: origin in the center of the considered object. It rotates with the body and thus it is non-inertial; the three orthogonal axes may or may not coincide with the principal axes, for which the inertia matrix is diagonal. It is the most used RF to describe the attitude motion of celestial objects with respect to an inertial frame.

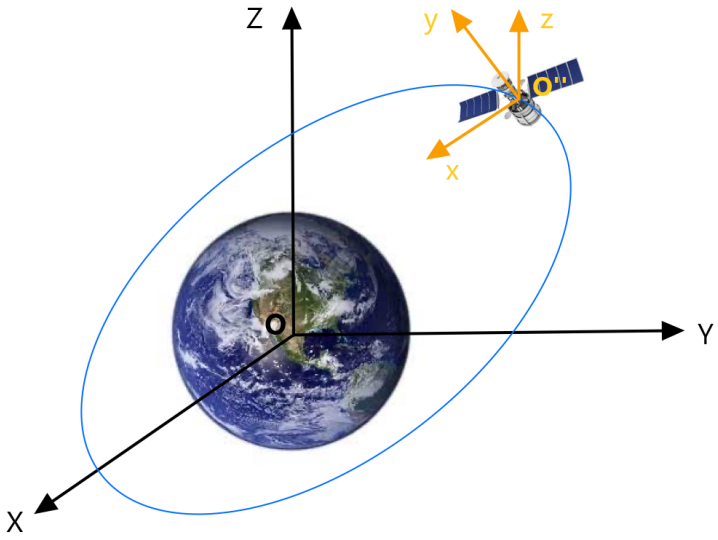


Figure 2.1: ECI and B RF: ECI in black (XYZ) is fixed in inertial space and originates from the Earth centre (O); B in orange (xyz) is rotating with the object

Another useful RF is the so-called Earth-Centered Earth-Fixed (ECEF). As its name suggests, its origin is in the Earth center and the xy-plane, corresponding to the Equator, is rotating with the x-axis that is following the location of the Greenwich meridian in inertial space tracked by the Greenwich Sidereal Time (GST - Greenwich position with respect to the Vernal Equinox Line). This RF can be used to identify the location of an observer on the Earth surface, through the geographical coordinates: latitude (vertical displacement with respect to the Equator) and longitude (horizontal displacement with respect to the Greenwich meridian).
Therefore, by setting the latitude and longitude coordinates of the observer, it is possible to identify the origin of another commonly used RF, that is the Topocentric Horizon (TH) one (Figure 2.2). In particular, the xy-plane of TH is parallel to the local horizon of the observer and, if the Earth is considered as a perfect sphere, tangent to the observer location. The z-axis follows the position vector of the observer starting from the center of the Earth towards the celestial sphere (infinite expansion of the Earth's sphere) called zenith.

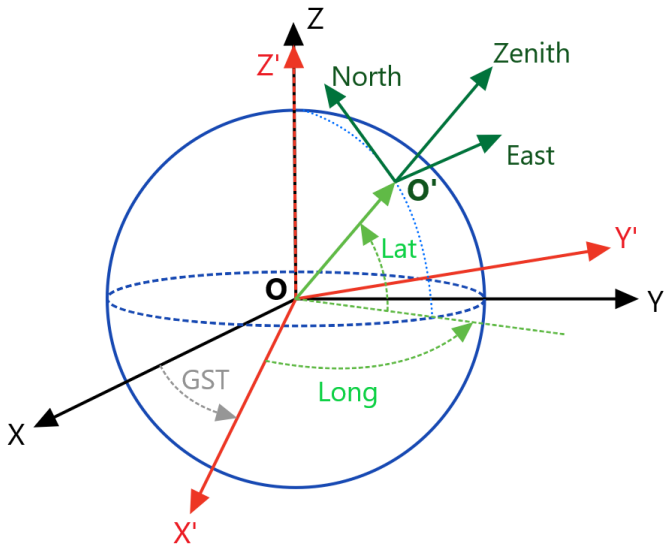


Figure 2.2: TH and ECI RF: TH in green (East, North, Zenith) is identified by the observer position (O') on the Earth surface. O' is defined by longitude and latitude with respect to ECEF in red (X'Y'Z'), which is rotated with respect to ECI in black (XYZ) of GST (function of time).

In this frame, it is possible to locate a celestial object with respect to the local observer horizon through the local coordinates: Azimuth (Az) and Elevation (El).

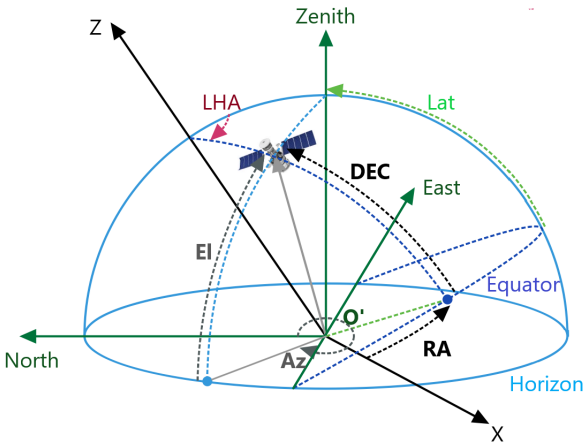


Figure 2.3: RA-DEC and Az-El: for a spherical Earth, TH and ECI can share the same origin; Az is computed clockwise starting from North, while RA is considered anti-clockwise starting from the Vernal Equinox Line. On the other hand, both DEC and El start from their respective xy-plane and cover an arc of great circle to reach the object onto the celestial sphere.

Therefore, a celestial object can be represented in two different RFs through two sets of coordinates, respectively RA and DEC in ECI and Az and El in TH (Figure 2.3).

In general, the transformation between RFs can be performed with rotation matrices. Given a vector $\mathbf{a_{123}}$ written in a generic inertial frame (123), it is possible to rewrite it as a new vector $\mathbf{a_{uvw}}$ in another RF (uvw) with the following equation:

$$\mathbf{a_{uvw}} = \mathbf{A_{rot}}\mathbf{a_{123}} \tag{2.1}$$

Where $\mathbf{A_{rot}}$ is a rotation matrix, i.e. it is square, orthogonal and with determinant equal to 1.
It is useful to determine the transformation between TH and ECI, since in certain situations either RA and DEC or Az and El of the target may be available. In order to retrieve the local coordinates from the celestial ones, some additional parameters need to be known:

- UTC: Coordinated Universal Time, it is defined by the passage of the Sun across the Greenwich meridian, and it is adopted globally. The local time is related to the UTC through the different time-zones.
- Longitude: longitude of the observer location started measuring from the Greenwich meridian (positive eastwards, negative westwards) in the range $[-180°, +180°]$.
- Latitude: latitude of the observer location starting from the Equator (positive northwards, negative southwards) in the range $[-90°, +90°]$.

Once the (Az-El) couple and the GST are determined from the celestial coordinates (cf. Appendix A), it is possible to build up the rotation matrix from TH to ECI [18]:

$$\mathbf{R_{N/T}} = \begin{bmatrix} -\sin\vartheta & -\sin\varphi\cos\vartheta & \cos\varphi\cos\vartheta \\ \cos\vartheta & -\sin\varphi\sin\vartheta & \cos\varphi\sin\vartheta \\ 0 & \cos\varphi & \sin\varphi \end{bmatrix} \tag{2.2}$$

Where ϑ is equal to the sum of the observer East longitude (Λ) and GST, while φ is the observer latitude.
Furthermore, rotation matrices can also be used to describe the attitude evolution of an object (i.e. time variation of the orientation of one frame with respect to another).
The different RFs that will be used throughout the work are summed up in Table 2.1.

RF	Inertial	x-axis	y-axis	z-axis
ECI	Y	Vernal Equinox Line	⊥ xz-plane	Earth Rotation Axis
B	N	X-Body	Y-Body	Z-Body
ECEF	N	Greenwich Meridian	⊥ xz-plane	Earth Rotation Axis
TH	N	Local East	Local North	Zenith

Table 2.1: Main Reference Frames

Euler's rotation Equations

Euler's rotation Equations (EE) describe the rotational dynamics of a body with respect to an inertial reference frame. In principle, EE are a system of non-linear totally coupled differential equations, where the angular velocities are the unknowns to be integrated.
For any 3D rigid body, the mass distribution can be expressed by means of the moments of inertia, that tell how the object will rotate starting from an initial condition or responding to applied torques. If the considered body has two symmetry planes relative to a given RF (e.g. B), then the latter coincides with the principal axes (PA) RF and the moments of inertia can be stored in a diagonal (3x3) matrix, called inertia matrix (**I**):

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2.3}$$

Where I_x, I_y and I_z are the principal moments of inertia (all positive) and the (xyz) axes are the body principal axes (PA) of inertia [19].
In this context, EE can be written as:

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \mathbf{I}\boldsymbol{\omega} \times \boldsymbol{\omega} + \mathbf{M} \tag{2.4}$$

Where $\boldsymbol{\omega}$ is the vector of angular velocities (ω_x, ω_y and ω_z) in the body/principal axes RF, while **M** is the vector of possible torques applied to the body. In order to obtain the three angular velocity components, numerical integration of three differential equations must be performed, and initial conditions are needed.
As stated in Sec. 1.2, in this work the generic space debris is considered to be a flat square plate of unitary area and mass; by definition, such a solid has three planes of symmetry and considering the z-axis as the out-of-plane one (see Figure 2.4), it is possible to write:

$$I_x = I_y = I = 1/12 \tag{2.5}$$

Then, thanks to the perpendicular axis theorem [20]:

$$I_z = I_x + I_y = 1/6 \tag{2.6}$$

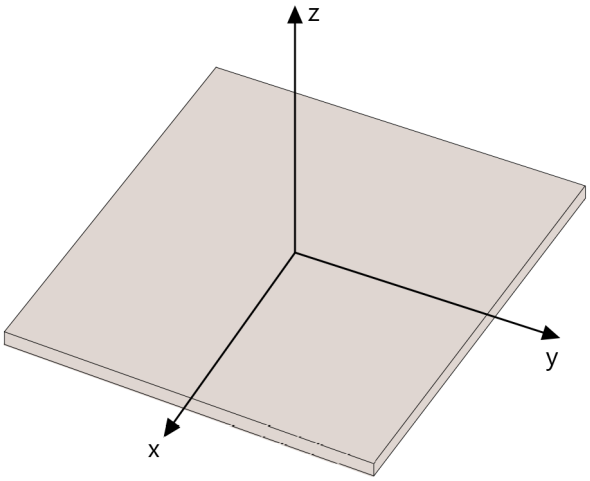


Figure 2.4: Flat Square Plate: B and PA frames coincide; the plate has an infinitesimal thickness and z is the major inertia axis [21].

Moreover, if the observation of the debris is sufficiently short and the dimensions are compact enough, all external torques can be brought to zero. Therefore, Eq. 2.4 can be simplified and written in components as:

$$\begin{cases} I\dot{\omega}_x = (I - I_z)\omega_y\omega_z \\ I\dot{\omega}_y = (I_z - I)\omega_z\omega_x \\ \dot{\omega}_z = 0 \end{cases} \tag{2.7}$$

In this way, the x and y equations are decoupled from the third one, whose solution is constant and equal to the initial condition ($\omega_z = \omega_{0z}$). Thus, an analytical solution in time for x and y can be easily found [21], given the initial conditions (ω_{0x} and ω_{0y}):

$$\begin{cases} \omega_x = \omega_{0x}\cos\hat{\lambda}t - \omega_{0y}\sin\hat{\lambda}t \\ \omega_y = \omega_{0x}\sin\hat{\lambda}t + \omega_{0y}\cos\hat{\lambda}t \end{cases} \quad \text{where:} \quad \hat{\lambda} = \frac{I_z - I}{I}\omega_{0z} \tag{2.8}$$

The angular velocities evolution can be seen in Figure 2.5.

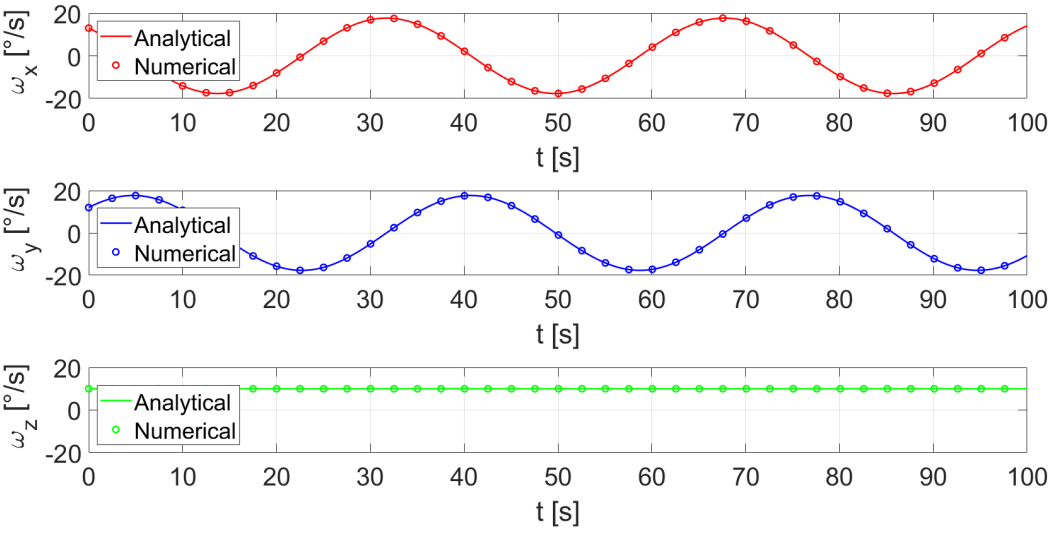


Figure 2.5: Dynamics of a flat square plate: evolution of angular velocities in a 100 second interval. Analytical and numerical solution give the same exact behaviour: constant angular velocity about the z axis (plot at the bottom), sinusoidal behaviour around the x and y axes (top two plots).

Attitude Parameters

The attitude parameters define the orientation of one RF with respect to another, i.e. rotation of B with respect to ECI. In general, there are several ways to represent the change in the three degrees of freedom that describe the attitude motion in space. Here, two main sets of parameters will be described.

As said before, it is possible to use rotation matrices, that are usually referred to as direction cosine matrix (DCM), to switch from one RF to another one. In particular, a DCM has nine components (3x3) and is formed by three direction cosines (cosines of the angles between each axis of the two RFs) stored column by column.
Considering a generic vector in ECI, its rotated version in the body frame through the DCM ($\mathbf{A_{B/N}}$) is:

$$\mathbf{a_B} = \mathbf{A_{B/N}}\mathbf{a_{ECI}} \tag{2.9}$$

Where the notation B/N means that the application to a vector of the DCM will rotate it from an inertial frame (N) to a body (B) one.
Moreover, DCMs allow to perform consecutive transformations between different RFs. For instance, the rotation of PA (if it is not already coincident with B) with respect to

an inertial RF (e.g. ECI) can be written as:

$$\mathbf{a_{PA}} = \mathbf{A_{PA/B}A_{B/N}a_{ECI}} = \mathbf{A_{PA/N}a_{ECI}} \tag{2.10}$$

Therefore, it is possible to represent multiple rotations just by pre-multiplying the starting vector by the DCMs that represent the successive rotations starting from the last one ($\mathbf{A_{PA/B}}$ first, and $\mathbf{A_{B/N}}$ second).

The orientation of a rigid body relative to an inertial frame can also be specified by only three parameters/angles [19].
The yaw-pitch-roll angles (ϕ, θ, and ψ) shown in Figure 2.6 define a sequence of three rotations that relates B and ECI.

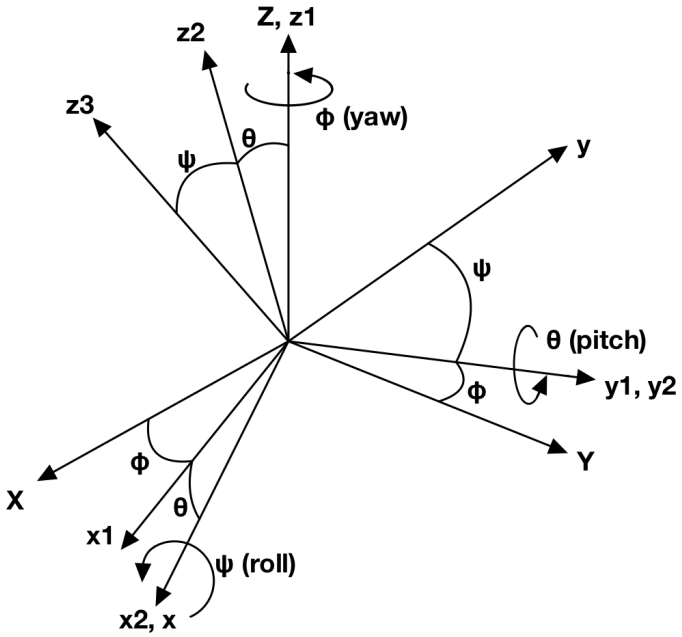


Figure 2.6: Yaw Pitch Roll angles: 321 sequence from XYZ (ECI) to xyz (B)

Their sequence is the 321 one:

1. Rotation about Z-axis through the yaw angle ϕ:

$$\mathbf{R_3}(\phi) = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

2. Rotation about new y-axis ($y_2 = y_1$) through pitch angle θ:

$$\mathbf{R_2}(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.12}$$

3. Rotation about x-axis through roll angle ψ:

$$\mathbf{R_1}(\psi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix} \tag{2.13}$$

As seen in Eq. 2.10, the overall rotation is given by multiplication of the three matrices in the opposite order:

$$\mathbf{A_{B/N}} = \mathbf{R_1}(\psi)\mathbf{R_2}(\theta)\mathbf{R_3}(\phi) \tag{2.14}$$

This equation represents the direct mapping from yaw-pitch-roll angles to DCM.

Once the set of attitude parameters to work with is chosen, it is possible to evaluate how these change in time in relation to the rotational dynamics in the body frame (kinematics). The kinematic evolution of a RF can be visualized as a set of consecutive rotations in small instants of time. Since no rule for consecutive rotations is present in the yaw-pitch-roll angles formulation, only DCM kinematics can be stated as [21]:

$$\dot{\mathbf{A}}_{\mathbf{B/N}} = -[\boldsymbol{\omega}\wedge]\mathbf{A_{B/N}} \quad \text{where :} \quad [\boldsymbol{\omega}\wedge] = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.15}$$

Therefore, in order to retrieve the time evolution of the DCM, Eq. 2.15 needs to be integrated alongside the EE with specified initial conditions.

Reflection Model

The final step is to express how the target object reflects the sunlight towards the observer and how the dynamics affects this phenomenon.
The main features to be considered are:

- Relative positions between the Sun, the observer and the debris
- Projected area towards the observer

- Properties of the sunlight reflecting surfaces

All the relative positions can be evaluated in the inertial frame (ECI) by using two versors ($\hat{\mathbf{O}}$ and $\hat{\mathbf{S}}$, see Figure 2.7). In particular, the debris location can be easily represented through celestial coordinates (RA and DEC), while the Sun position in ECI can be estimated through ephemeris (see Sec. 3.1):

$$\hat{\mathbf{O}} = \begin{bmatrix} \cos(RA)\cos(DEC) \\ \sin(RA)\cos(DEC) \\ \sin(DEC) \end{bmatrix} \quad \text{and} \quad \hat{\mathbf{S}} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} \tag{2.16}$$

Once the two versors are available, it is necessary to represent how the rotating flat plate is seen by the observer.

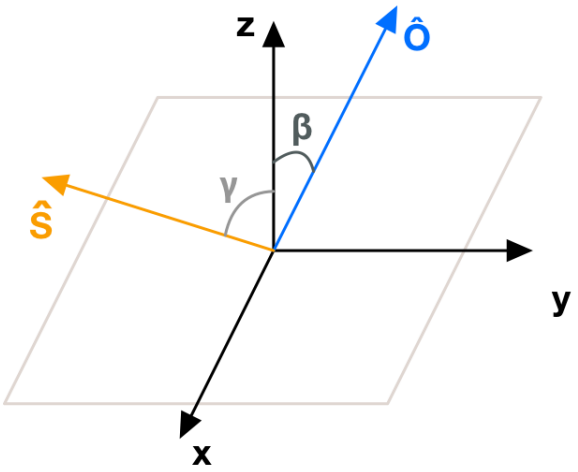


Figure 2.7: Reflecting Model

In general, the projected area (A_{proj}) of the plate is proportional to the cosine of the angle between the normal to the surface (z-axis in this case) and the direction of observation ($\hat{\mathbf{O}}$). Nevertheless, the illumination conditions that are given by $\hat{\mathbf{S}}$ need to be considered: the portion of area illuminated (A_{ill}) by the Sun is in fact proportional to the cosine of the angle between the normal to the surface (z-axis again) and the Sun direction. Therefore, the overall illuminated-projected varying area (A_{var}) is just the product of the simply projected and simply illuminated areas. Moreover, it should be noted that a face will give a contribution ($A_{var} \neq 0$) only if it is actually reflecting the sunlight and the illuminated

portion is visible by the observer at the same time [22]:

$$\begin{cases} A_{var} = A_{proj} A_{ill} = (\hat{\mathbf{O}} \cdot \hat{\mathbf{N}})(\hat{\mathbf{S}} \cdot \hat{\mathbf{N}}) = \cos\beta \cos\gamma, & \text{if } \gamma < 90° \text{ and } \beta < 90° \\ A_{var} = 0, & \text{otherwise} \end{cases} \tag{2.17}$$

Where $\hat{\mathbf{N}}$ is the normal to the face of the flat plate considered, while β and γ are the angles between observer-normal and Sun-normal, respectively (see Figure 2.7). Considering a zero-thickness plate, the only faces that can actually reflect light have as normal $+z$ and $-z$; thus, only one of the two will be illuminated and visible at the same time.
Equation 2.17 identifies a light curve, since it is a scaled representation of how the light intensity reflected by the object changes in time (due to its dynamics). In order to express a proper light variation, there are two possible strategies: in the first one, the area variation is linked to a physical brightness through complex reflectance functions that rely on the physical properties of the object [23]; the second and chosen one, given the generality of the approach, links A_{var} to a pixel intensity (see Sec. 2.2) - L_{var} - by introducing a shift (n) and a multiplicative factor (k), so that:

$$L_{var} = A_{var} \cdot k + n \tag{2.18}$$

As detailed in Sec. 3.3, the shift n is computed as the mean of the pixel values in the image. Therefore, if $A_{var} = 0$, the variation in pixel intensity L_{var} is actually uniformed to the background and the streak is correctly not visible.
With this simplified approach, the plate specularly reflects the rays hitting the surface with an angle equal to the incident one with respect to the normal. In general, the portion of light reflected is regulated by the specular reflection coefficient (ρ_s); in this case, ρ_s is embedded inside the values of k and n, which will be further discussed in Chap. 3.

2.2. Optical Telescopes

The main operations performed by an optical telescope are [24]:

- Collection of light using static or dynamic optics and collimation (rays brought parallel) to fit subsequent optics
- Filtering of specific wavelengths
- Changing the direction of photons through optics and focusing them on the detectors
- Transformation of photons in voltage (analog) performed by detectors

- Conversion from analog to digital quantity
- Processing and storing in memory of digital info into a digital file

The light-collecting surfaces and the optics that focus the rays onto the detectors (or the eye-piece for human observation) are either mirrors or lenses; thus, optical telescopes are divided in refractors, if lenses are used to focalize rays, and reflectors, if mirrors are employed to reflect the impinging rays onto the detector (Figure 2.8).

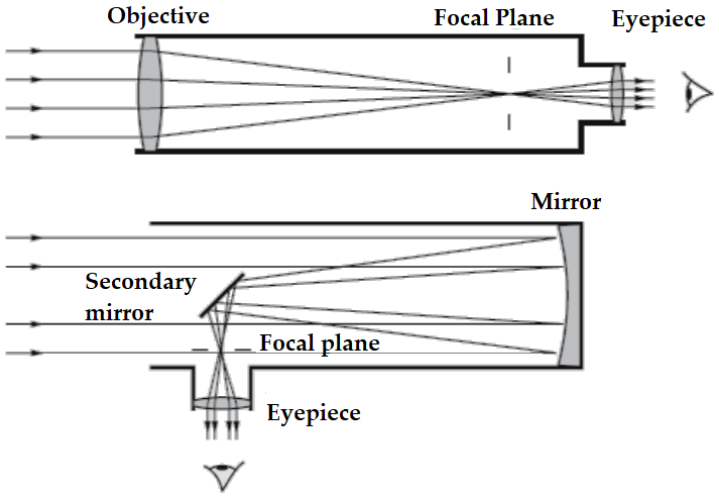


Figure 2.8: Refractor and reflector telescope [25]

Ground observations are inevitably influenced by the presence of the atmosphere [25]. First of all, when the light passes through the unsteady turbulent air (with gradient of density and temperature), its rays undergo random changes in refraction and are no longer parallel to each other. Thus, the amount of light that reaches the detector varies constantly and the source appears to scintillate. In order to alleviate such a problem, telescopes collect light over a large area, diminishing the rapid changes of the scintillation. Nevertheless, point sources still may experience differences in refraction along different paths of light through the atmosphere that smear the image. This phenomenon is called seeing and it is embedded in the physics of optical telescopes for which the image of a point source will never be a point, but it will be spread onto the detector (in combination with the intrinsic diffracting nature of light). Finally, all the light that passes through the atmosphere may suffer scattering and absorption by molecules and could be attenuated by dust particles.
Therefore, transparency and darkness are fundamental to observe faint objects in the night sky. Moreover, some wavelength regions in the electromagnetic spectrum are strongly absorbed by the atmosphere (see Figure 2.9); luckily, the visible window (from 300 to 800

nm) is practically undisturbed.

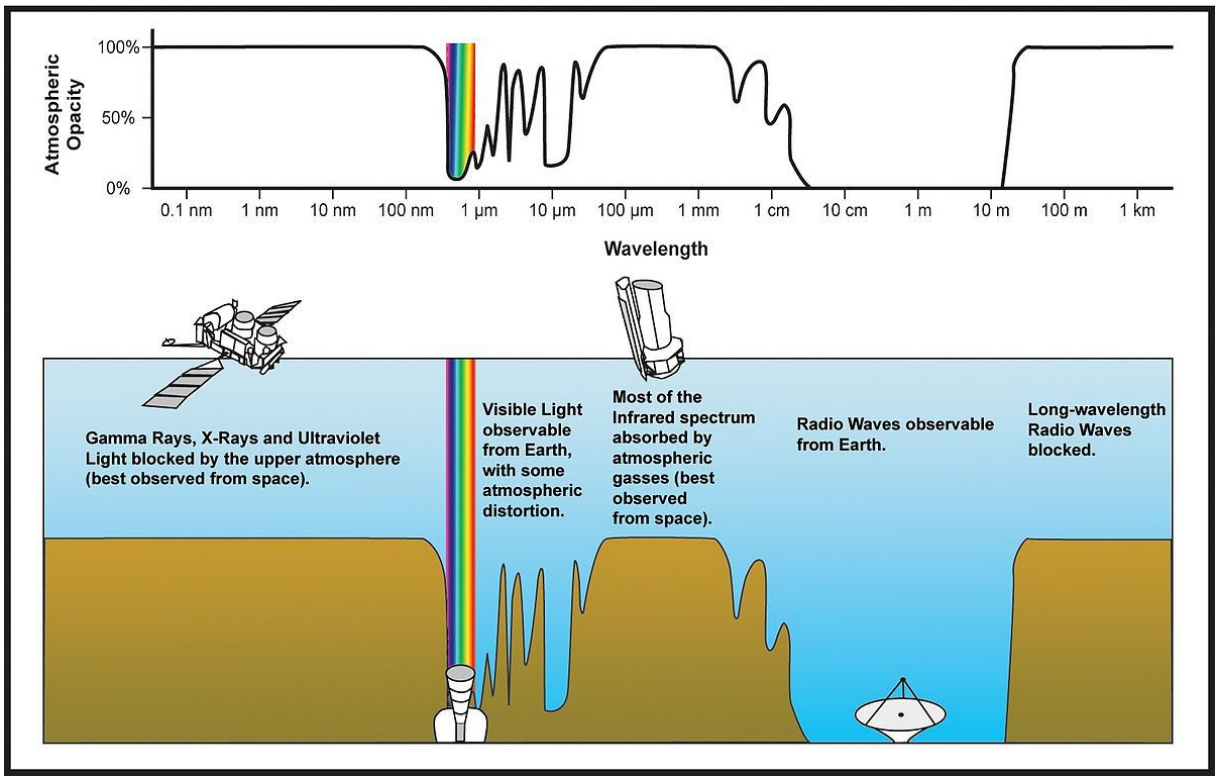


Figure 2.9: Atmospheric Electromagnetic Opacity: the most energetic part of the spectrum is completely absorbed by the atmosphere, while the opacity decreases for the visible part and is basically null for radio-waves [26].

Consequently, the three major tasks [25] that an optical telescope must fulfill are:

1. Collect light from a large area, in order to diminish scintillation phenomena and study faint objects
2. Increase the apparent angular diameter of the object, improving the resolution
3. Measure the positions of the objects

Once the light is correctly focused onto the detector, with the minimization of external disturbances, it is converted into a digital format and stored in an image file that can have different extensions (mainly FITS files).

Refractor Telescope

In refractor telescopes the light is collected by a lens that forms an image in the focal plane. The light gathering power of the telescope is characterized by the ratio of the aperture to

the focal length, that is the distance between the lens and its focus (point on optical axis where the infinitely-distant parallel rays are brought). Moreover, the telescope can be characterized by the magnification and the resolving power. The former is obtained with systems of lenses that enlarge the image projected onto the detector/eyepiece, while the latter determines the minimum angular separation between two point sources in the sky that the instrument is able to discern. The theoretical limit for the resolution is set by the diffraction of light, for which the image of a point is formed as a small disk. The spreading of a point source onto the detector has to be carefully evaluated when performing studies on the celestial object that generated that disk (see Sec. 3.2).
The main issue for refractors, which are usually quite heavy instruments, is the chromatic aberration: the light refracted by the lens is dispersed, since glass refracts different colours (wavelengths) by different amounts. In order to remove such aberration, achromatic lenses (combination of two or more lenses) collimate all the different rays back to one point.

Reflector Telescope

Reflector telescopes use mirrors to focalize the light onto the detectors, and are the most commonly used telescopes in space research. This kind of instrument is particularly suitable for large systems, since mirrors are easier to manufacture and lighter than lenses. Mirrors are usually spherical or parabolic and coated with a thin layer of aluminium; in this way, the rays hitting their surface are reflected and brought to a point (focus of the parabola or centre of the circle) independently from their wavelength (reflection does not suffer from chromatic aberration). Nonetheless, different aberrations may still be present such as coma and spherical aberrations. There are different configurations of reflectors, according to the way they reflect and bring the light towards the detector. One of the most common architectures is the Newtonian one, with a two mirrors configuration: the primary reflects the light onto a secondary put in front of the primary focus; in this way, the secondary inclined mirror reflects the light onto the detector located on the side of the telescope tube (see Figure 2.8).

Detectors

Detectors are usually placed after collimation, filtration and isolation of the light entering the telescope. Their task is to convert the incoming photons into an electric current proportional to the radiation amount. Once this photon-electron conversion is performed, the information has to be digitized (using analog-to-digital (A/D) converters), processed and stored.
There are two main devices employed to perform the aforementioned tasks: Charged-

Coupled-Device (CCD) and Complementary Metal-Oxide Semiconductor (CMOS) cameras (see Figure 2.10). In both, each pixel (smallest portion of sensitive material inside the device) could be associated with a "bucket" collecting electrons from the incident light [5]. The key difference is that CCDs have only one reading circuit to collect the electrons row by row, while CMOS perform this process in parallel thanks to the presence of one reading element for each pixel. Usually, CCDs provide high quality and low noise measurements and are quite simple if compared to CMOS. Generally though, CCDs are limited in reading speed due to the single acquisition system. This means that for high resolution images, mechanical shutters are needed to blind the detector during the read-out. On the other hand, CMOS does not suffer from reading speed issues thanks to its architecture with a single A/D converter for each pixel (or at least for each row of pixels). Nevertheless, CMOS cameras provide low quality, high noise measurements and are rather complex and expensive devices.

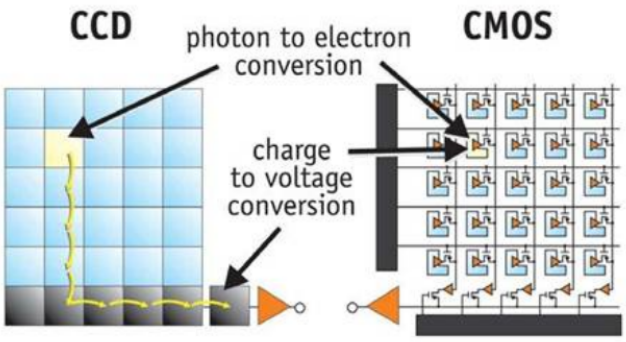


Figure 2.10: CCDs vs CMOS sensors [24].

For both technologies, one of the main issues is pixels saturation, that occurs when photons hit the sensitive material and generate a current too high to be properly handled by the read-out circuit. Therefore, a two-fold problem arises: firstly, the corresponding digital value will be at its maximum and not necessarily proportional to the actual radiation; secondly, the electrons in the pixel-bucket may overflow and spread over the adjacent pixels (bleeding effect), that in turn may saturate. Therefore, when the radiation reaching the detector is excessive, either the device is covered with mechanical or electronic shutters (protection from high energy radiation [27]) or saturation is accepted and studied. This second situation is what usually happens with space debris in LEO with high reflecting area [28] (e.g. International Space Station, ISS) that leave a streak on the image of almost completely saturated pixels. As detailed in Sec. 2.3, it is still possible to study and extract light curves of such saturated pixels by exploiting the spreading of the streak itself across the image.

FITS Images

Keyword		Keyword	
SIMPLE	Logical constant with the value T if the file conforms to FITS Standard	BSCALE	Floating-point number representing the coefficient of the linear term in the scaling equation, the ratio of physical value to array value at zero offset
BITPIX	Integer, its absolute value is used in computing the sizes of data structures	BZERO	Floating-point number representing the physical value corresponding to an array value of zero (default 0.0)
NAXIS	Non-negative integer no greater than 999 representing the number of axes in the associated data array	OBJCTRA	Character string giving the RA of the centre of the image in hh:mm:ss
NAXISn	Non-negative integer representing the number of elements along Axis n of a data array	OBJCTDEC	Character string giving the DEC of the centre of the image in ±dd:mm:ss
DATE-OBS	Character string giving the date on which the HDU was created, in the form YYYY-MM-DDThh:mm:ss	SITELAT	Character string giving the latitude of the observer in ±dd:mm:ss
TELESCOP	Character string identifying the telescope used	SITELON	Character string giving the longitude of the observer in ±dd:mm:ss
EXPOSURE	Floating-point number giving the exposure time of the image	FOCALLEN	Floating-point number giving the focal length of the instrument used (may not always be present)

Table 2.2: FITS Keywords

According to NASA's Definition of FITS [29], the Flexible Image Transport System (FITS) is the standard archival data format for astronomical data sets. FITS files are in fact commonly used by astronomers and space researchers as a data interchange and archiving format for astronomical images. These files allow the user to store the image and the fundamental metadata that describes it. Due to the peculiarity of the files, specialized software is needed to read them [30].
A FITS file is mainly composed of the primary Header and Data Unit (HDU) FITS structure. The HDU contains the primary header and may be followed by the primary data array. The header consists of different header blocks, describing important features of the image itself with relevant keywords as seen in Table 2.2. On the other hand, the primary data array comprises a single data array (1 to 999 dimensions, specified by `NAXIS`, see table). Its most simple form (`NAXIS` = 2) represents the value of the x and y pixels of the image. The pixel bit-depth in the image is specified in the header (`BITPIX`) and may be equal to 8, 16, 32 or 64 levels. 8-bit integers are unsigned and contained in one byte, corresponding to decimal values ranging from 0 to 255 (2^8 values). 16, 32, and 64-bit integers are two's complement signed binary integers, contained in two, four or eight bytes with decimal values ranging from $-2^{n_{bits}}$ to $2^{n_{bits}} - 1$.
Sometimes, a linear scaling is introduced in order to reduce the size of the data array; thus, the physical value of the pixels differs from the array one:

$$\text{physical\_value} = \texttt{BZERO} + \texttt{BSCALE} \cdot \text{array\_value} \tag{2.19}$$

Another important feature that can be represented within FITS HDUs is the mapping between image coordinates and physical (i.e., world) coordinate systems (WCSs). The transformation working principle can be seen in Figure 2.11.
If the FITS header does not present any WCS information, it can be computed through the `Astrometry.net` web service operating at `nova.astrometry.net` [31]. In particular, FITS images can be submitted for astrometric calibration and the computed WCS info are directly inserted into the image header.

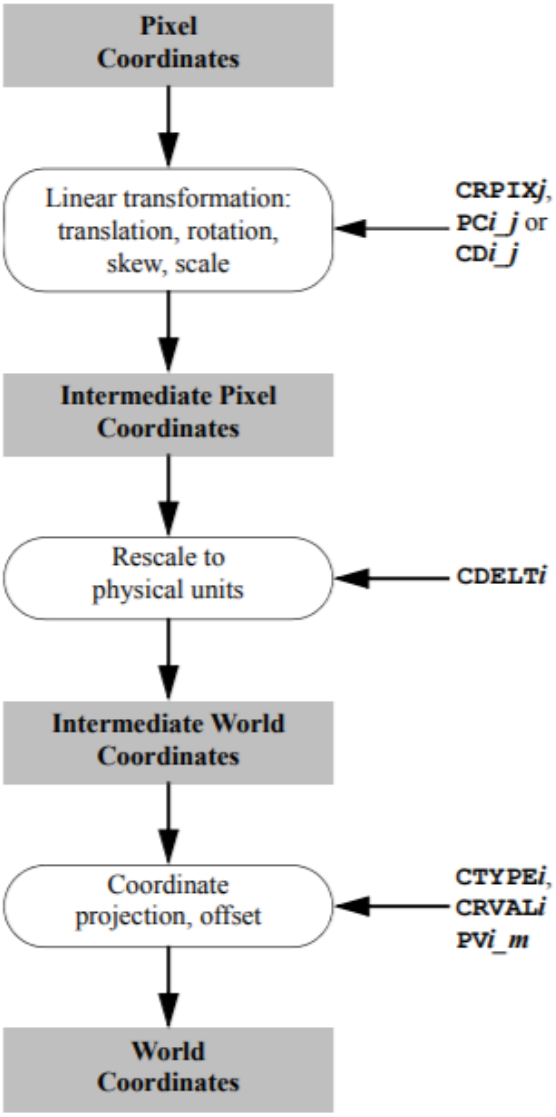


Figure 2.11: A schematic view of converting pixel coordinates to world coordinates (e.g. celestial coordinates) [29].

2.3. Light curves and Photometry

In Sec. 1.2, different methods to study the attitude motion of a celestial object are reviewed. In all of these techniques, the object light curve has to be known. In this section, a more thorough review of how to extract and characterize light curves is presented. Moreover, a brief overview of photometry and catalogue matching is essential in order to understand the principles behind the working of SAP (cf. Sec. 3.2).

Light curves

In the most generic case, a light curve is the variation of light intensity emitted or reflected by an object in a certain interval of time (Figure 2.12).

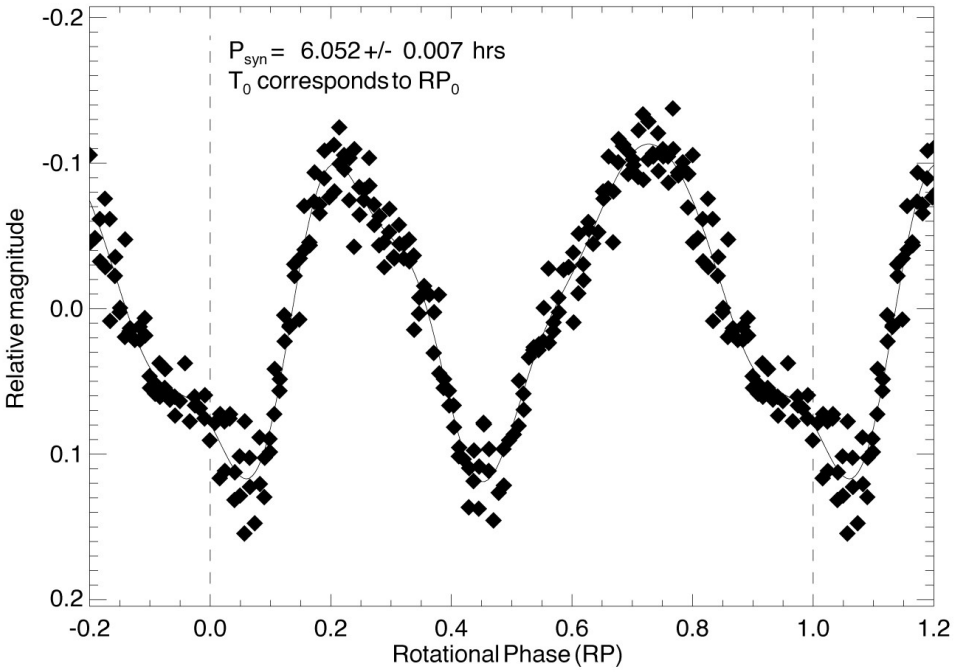


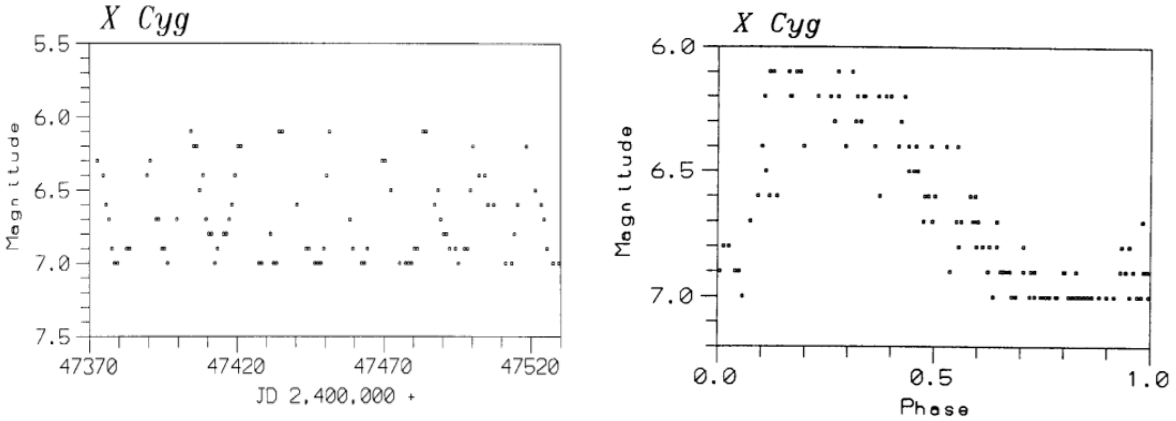
Figure 2.12: Example of a light curve of asteroid Steins: data from ESA Rosetta Mission [32]

The light intensity can be measured in different ways; in the most simple case, it can just be represented by the pixel values that are directly connected to the photons collected by the detector (see Sec. 2.2). That said, in many applications it is preferable to work in terms of magnitude. The magnitude is a unitless quantification of the light intensity in logarithmic scale. In particular, the transformation from intensity to magnitude must be performed with respect to a reference measurement:

$$M - m_{ref} = -2.5 \cdot \log_{10}\left(\frac{I_L}{I_{L_{ref}}}\right) \tag{2.20}$$

Where m_{ref} and $I_{L_{ref}}$ are the magnitude and light intensity of a reference celestial object, while M and I_L are the ones of the target object. According to this formula, the brighter the object (the higher the intensity), the lower the value of the magnitude; thus, the brightest objects in the sky will have negative values of magnitude.
In particular, Eq. 2.20 is used to define the apparent magnitude, i.e. an estimate of the brightness of a celestial body as seen from Earth. Therefore, it is influenced by the

object distance from the observer and interstellar absorption. Alternatively, it is possible to define the absolute magnitude, that is the apparent magnitude that an object would have if it were placed at a distance of 10 parsec (approximately 32.6 light-years). In this way, the intrinsic luminosity of the object, i.e. what it actually emits or reflects, is taken into consideration.
Finally, if the light curve shows a visible periodicity, it is sometimes useful to work with the so-called folded light curves, that is light curves represented with respect to full cycles (Figure 2.13).



(a) (b)

Figure 2.13: Light curves of variable star X Cyg [33]: **(a)** is the unfolded light curve, while **(b)** is the folded one.

In this way, instead of considering a time evolution, the light intensity or magnitude can be plotted against a rotational phase representing how far into the cycle the evolution is. Therefore, consecutive sections of the light curve are overlapped and a compressed representation is obtained in a range 0 to 1 (full period). It is worth noting that this different representation needs the period to be available (which is actually the unknown in the current problem); nevertheless, besides being employed in post-processing of the light curve to better visualize it, it can also be used in some period finding methods (cf. Sec. 1.2).

Photometry

Photometry is the science of measuring the amount of energy (flux), in the form of electromagnetic radiation that is received from celestial objects. The measurement of the flux is of fundamental importance to retrieve information about the total energy output of the

object (luminosity), the object size and other physical properties [34].
As seen in Sec. 2.2, the images coming from the detectors (either CCDs or CMOS) are naturally in a digital form with a linear relationship between photons and pixel values. Whenever this linearity is lost (pixel saturation), another way to retrieve the information about the photon count is needed.
Moreover, there are several complications that must be dealt with: atmospheric seeing determines a spreading of the objects in a number of pixels and causes the shape of the object image to vary with time; the sky may show undesired external light sources that need to be filtered out; in some scenarios, images of different objects may overlap (crowded fields), and a way to separate them must be envisaged.
Considering the image of an isolated star, a fundamental concept to proceed is the point spread function (PSF). The PSF describes the response of a detector to a point source. In general, the shape of a PSF should be circularly symmetric. Nonetheless, seeing forces to approximate a PSF as a central Gaussian core and a large halo approximated by a power law. A common measure of the angular size (estimate of the image spreading pixel-wise) of a PSF is the full width at half maximum (FWHM) that is the diameter where the flux falls to half its central value (see Figure 2.14).

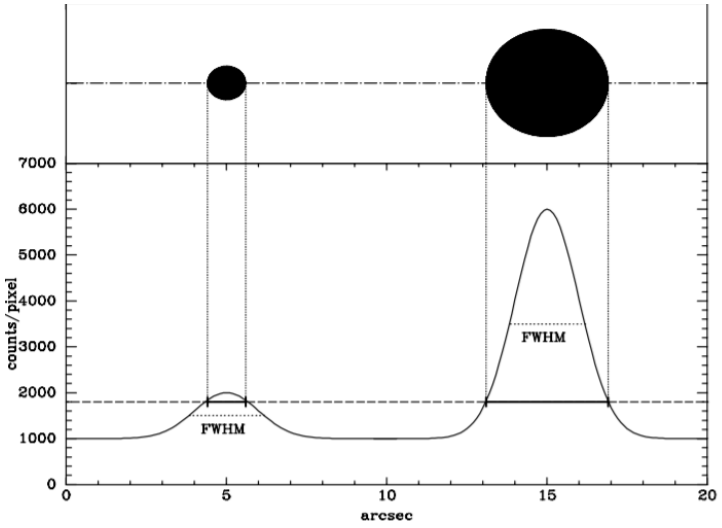


Figure 2.14: FWHM of a faint and bright star: brighter stars look bigger than faint ones, even if their image shape and size is exactly the same, due to the higher intensity of the latter [34]

Once the shape of the PSF is known, the relation between the pixel value and star brightness has to be retrieved. First of all, a window inside which all the pixel values are summed up (aperture) is defined. The aperture can have different shapes (circular, ellipsoidal, rectangular) and its dimensions is usually defined from the FWHM estimate of

the PSF. In particular, optimal aperture measurements have a diameter about 1.4 times the FWHM [34].
Then, the aperture is centred on the star to be measured and the flux is computed as the sum of all the pixel values contained inside. As mentioned before, the sky background needs somehow to be subtracted; there are two main ways to achieve so:

- Subtraction of the statistic mean of all the pixel values to the image itself. In this way, the background pixels are approximately zero and give no contribution to the flux measure.
- Definition of a central circular aperture for the flux and annulus measurements. With the annulus, an estimate of the background next to the source itself is provided, leading to more accurate results.

The second method is in general preferred, but needs a sufficiently isolated star, so that no other contribution but the sky background is falling inside the annulus.
Finally, once the flux coming from a given source is determined, its instrumental magnitude (I_m) can be defined as:

$$I_m = -2.5 \cdot \log\left(\frac{N_{ap}}{t_{exp}}\right) \tag{2.21}$$

Where N_{ap} is the sum of the counts in the sky-subtracted aperture, while t_{exp} is the exposure time of the image.
Therefore, aperture photometry gives the flux and instrumental magnitude of a star in an image. Nevertheless, it can be applied to any source of light, even to streaks left by space debris. In this case, there are different ways to define the shape of the aperture: the most simple approach is to define rectangular apertures [35] that, pixel-by-pixel, slide along the streak itself (cf. Sec. 3.2).

Now, if a reference star is identified in the field of view (FOV), the apparent magnitude of the object of interest (M) only demands I_m of both sources (reference ($I_{m_{ref}}$) and object of interest (I_m)) [36]. Then, similarly to Eq. 2.20:

$$M = m_{ref} + I_m - I_{m_{ref}} \tag{2.22}$$

Clearly, for M to be determined, m_{ref} needs somehow to be known.
In this work, aperture photometry will be mainly applied to the apparent magnitude variation of saturated streaks, while for non-saturated ones a simpler approach evaluating the central light intensity variation will be used (cf. Sec. 3.2).

Catalogue Matching

To compute the apparent magnitude of an unknown object - whether a star or a space debris reflecting light -, as seen in Eq. 2.22, the magnitude of a reference star must be known.

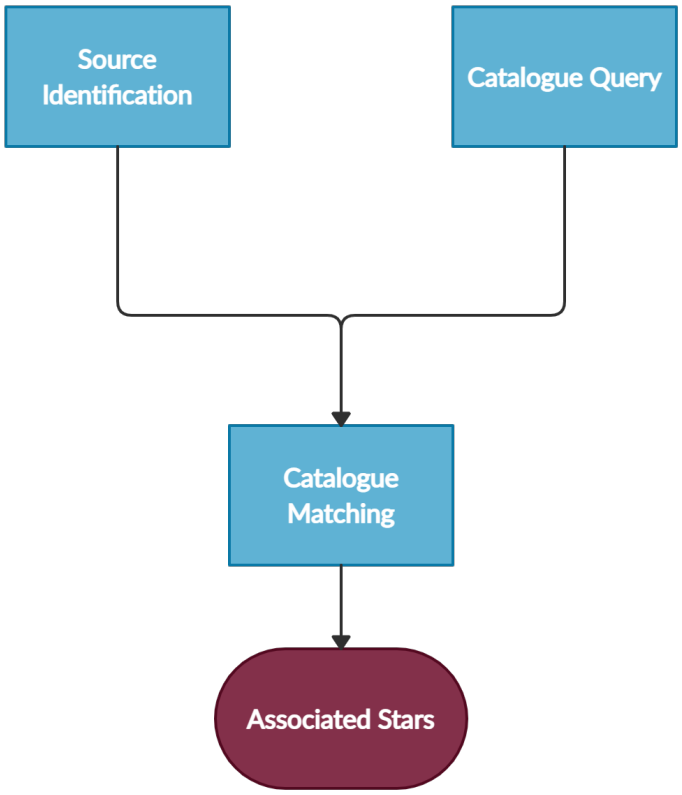


Figure 2.15: Catalogue Matching Workflow: 1. Source Detection; 2. Catalogue Query through different servers; 3. Catalogue Matching to get star association.

The identification of a star in a certain portion of the sky requires few steps (see also Figure 2.15):

1. Identification of the sources in the image: the stars present in the image must be pinpointed in terms of their celestial coordinates (RA and DEC). The identification is carried out through source finding algorithms [37].
2. Catalogue Query: given the central celestial coordinates of the image and its angular extension, star catalogues can be queried in order to retrieve the actual stars inside the search radius set by the image width. In particular, star catalogues usually present information about positions in the sky and magnitudes in different spectra of the stars [38].

3. Catalogue Matching: comparison between the positions of the identified stars in the image and the queried ones. A proper matching allows to associate each star in the catalogue to the ones in the image.

If the matching is successful, a control star can be chosen (usually the brightest one in the field, i.e. lowest magnitude) and its apparent magnitude (m_{ref}) used as a reference.

2.4. Genetic Algorithm

The genetic algorithm (GA) is one of the most used metaheuristic algorithms for optimization related problems [39]. These algorithms can solve complex real-life problems, and are inspired by biological evolution process, swarm behavior, and physics law. It is possible to broadly divide such algorithms in single-solution based and population based ones. The first uses single candidate solution and improves it by means of local search that may cause the solution to get stuck in local optimum. On the other hand, population-based metaheuristics use multiple candidate solutions during the search process, maintaining sufficient diversity in the population to expand the search region and reduce the value of the cost function.
Among the population-based metaheuristic algorithms, GA is inspired from evolutionary biological processes. In fact, it mimics the Darwinian theory of survival of the fittest in nature [40]. In GA, usually five phases are considered: Initial Population Generation; Fitness Function Evaluation; Selection; Crossover; Mutation.

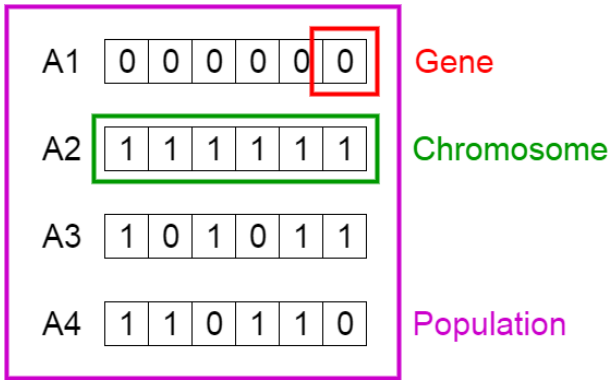


Figure 2.16: Population, Genes and Chromosomes [41]

The process begins with the random generation of a set of N individuals called population. Each individual is a possible solution of the considered problem, i.e. parameters that may guarantee a global optimum (Y_b) of the objective function. An individual is represented by a set of variables (genes) that, in the most simple implementation of the

algorithm, are joined in a binary string (either 0 or 1 value), called chromosome Y_i (see Figure 2.16).
The fitness function determines how fit each individual is according to the objective to minimize in the problem. Each chromosome of the population is evaluated by the fitness function and a fitness score is associated to it. Depending on this value, each individual will have a certain probability to be reproduced into the successive generations. In fact, in the selection process two pairs of individuals (parents) are selected based on their fitness score. The higher the fit of the individuals, the higher the probability to be selected.
Crossover is the most significant phase of the GA: for each obtained pair of parents, a crossover point (CP) is randomly chosen within the genes. Genes inside each chromosome of the two parents are swapped up to the crossover point, and two new chromosomes are created (offspring, see Figure 2.17). The new offspring are then added to the population. In some of them, certain genes may be subject to mutation with a low random probability, i.e. certain genes in the string are flipped from one value to another. Mutation is introduced in order to guarantee a correct diversification in the metaheuristic algorithm. The selection, crossover, and mutation operations are repeated on the current population until a new population is obtained (new generation). Since the number of chromosomes in the population is constant, in new generations individuals with the lowest fitness score are discarded, giving way to new (better fit) offspring. The sequence of phases is repeated to produce new individuals in each generation characterized by progressively higher fitness scores. Convergence is obtained as soon as the produced offspring are not significantly different from the previous generation (according to a certain function tolerance, $FunTol$) or a maximum number of generations (N_{MAX}) is reached.

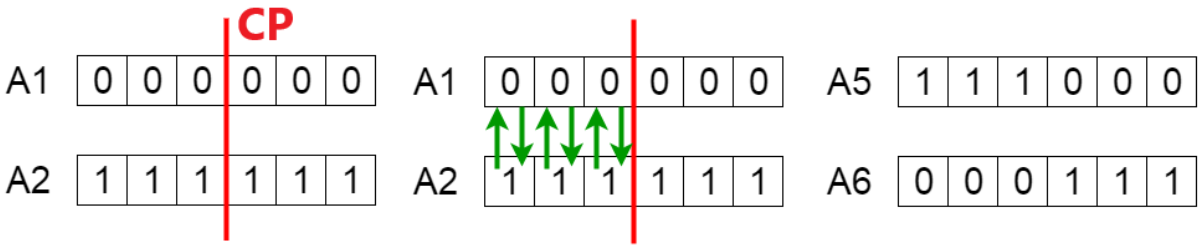


Figure 2.17: Crossover Scheme [41]

GAs use a variety of operators to perform the phases listed above. In particular, there are several different encoding schemes employed to translate each possible solution into a chromosome:

- Binary encoding: the gene or chromosome is represented as a string of 1 or 0 (as seen before)

- Octal encoding: the gene or chromosome is represented in the form of octal numbers, from 1 to 7
- Hexadecimal encoding: the gene or chromosome is represented in the form of hexadecimal numbers, from 0 to 9 or from A to F
- Value encoding: the chromosome is represented using a string with different values that can be real (floating-point) or integer numbers, or characters

As it can be seen in Sec. 3.3, the implementation of the GA in MATLAB (`ga.m`, [42]) is employed in the code. In particular, `ga.m` uses a floating-point value encoding scheme, since it is able to solve more complex problems with respect to simple binary encoding. Furthermore, there are various selection, crossover and mutation techniques that can be employed in order to obtain a better convergence of the GA [39].
The working principle of GA is summed up in Alg. 2.1.

Algorithm 2.1 Genetic Algorithm [39]

- Generate initial population of N chromosomes Y_i $(i = 1, ..., N)$
- Set iteration counter $k = 0$
- Compute fitness function of each chromosome

while $k < N_{MAX}$ **do**
- Select chromosomes with best fitness score
- Apply crossover operation on the pair of parents with crossover probability
- Apply mutation operation on the produced offspring with mutation probability
- Replace old population with newly generated population
- Evaluate the fitness function on the new individuals
- Evaluate the variation (var) of the fitness score with respect to the previous generation

if $var > FunTol$ **then**
- Increment iteration counter: $k = k + 1$

else
break

end if

end while

- Return the global optimum Y_b

3 | Angular Velocity Extraction Process

As seen in Sec. 1.2, the proposed approach to determine the attitude state of the space debris requires three main computation steps:

1. Light curve extraction and post-processing
2. Light curve simulation through the dynamics and kinematics of a square flat plate
3. Optimization of the objective function, minimizing the difference between the two light curves

In order to validate the developed code, synthetic images are generated and analyzed. For the creation of such images, the same code used to generate a target dynamics is leveraged to reconstruct the area variation fed to TIG. As mentioned before, TIG is an image tracklet generator developed by A. De Vittori and R. Cipollone in their master's thesis work [5]. In particular, TIG is customized to draw space object tracklets, considering the tumbling motion, over a night sky image.
Images (whether real or synthetic) are analyzed through SAP (Streak Analysis Pipeline) (cf. Sec. 3.2), a novel Python application specifically developed for this thesis. SAP extracts the main features of the target FITS image and streak (i.e. the real light curve). FITS image metadata (observer location, time of observation) and some streak properties (RA and DEC evolution in time) are exploited by the dynamics simulator (developed in MATLAB) to generate the corresponding overall area variation (cf. Sec. 3.1). By scaling this value according to Eq. 2.18, the simulated light curve is generated. The real light curve is instead fitted by the optimization process carried out in MATLAB with `ga.m` by changing the inputs of the dynamics simulator (cf. Sec. 3.3).

3.1. Synthetic Image Generation

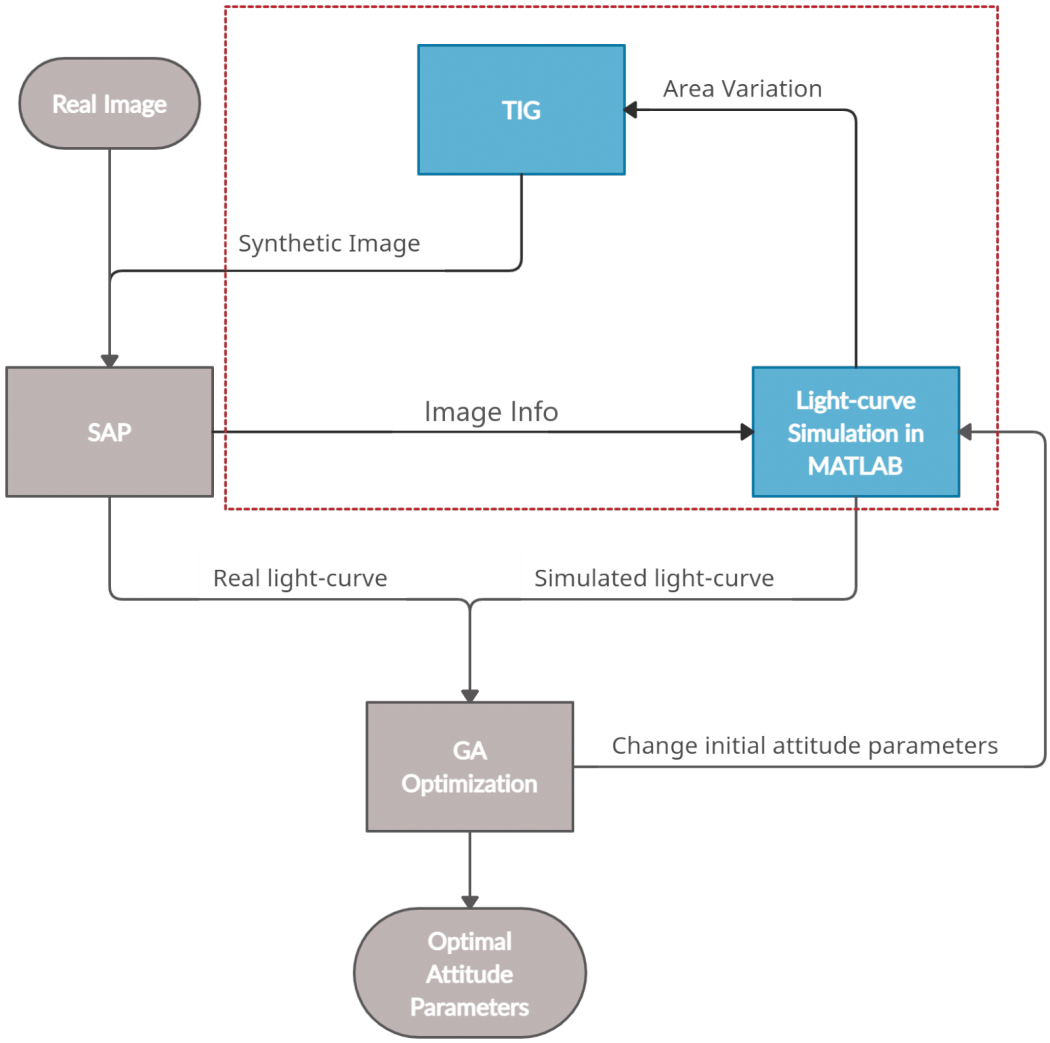


Figure 3.1: Thesis Workflow: Synthetic Image Generation

The first step performed when generating synthetic images (Figure 3.1) is the simulation of the plate dynamics. In order to do so, the main physical concepts exposed in Sec. 2.1 are followed. Then, once the unitary illuminated-projected area variation is determined, TIG is used to generate the image that will be fed to SAP (see Figure 3.2).
Since the dynamics simulation (used also in the case of real images in the optimization step (cf. Sec. 3.3)) needs some information stored in the input FITS image header, as well as some important features extracted by SAP (see Sec. 3.2), a two-run approach is followed in the generation of the image. A first run (trial run (TR)) sets up a trial image, used in a second run (real run (RR)) that finally produces a synthetic image.

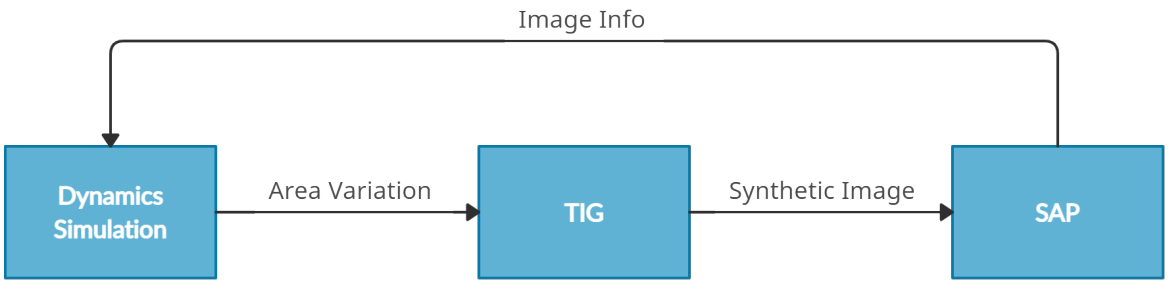


Figure 3.2: Image Generation Block: SAP is only used to provide information to set up trial images when generating synthetic images.

Before giving more details about this two-run approach, a brief explanation of the dynamics simulator and TIG working principle must be done, in order to better understand the rationale behind the two-run structure.

Dynamics and Kinematics Simulation

The dynamics simulation of the square flat plate is performed in MATLAB programming environment. The code consists of two parts: a script and a function.
The script (`main_dyn.m`) reads from the tables coming from SAP (cf. Sec. 3.2) the evolution of RA and DEC of the considered object and the associated time vector (L points), the UTC, latitude, and longitude of the observer. Moreover, it sets the value of the inertia matrix of the flat plate and its normal (z-axis):

Quantity	Dimension	Unit	Description
Ra	`double` [Lx1]	[°]	RA evolution of the streak
Dec	`double` [Lx1]	[°]	DEC evolution of the streak
t	`double` [Lx1]	[s]	Time vector
UTC	`datetime` [1x1]	[-]	UTC of the observation
φ	`double` [1x1]	[°]	Latitude of the observer
λ	`double` [1x1]	[°]	Longitude of the observer
I	`double` [3x3]	$[kgm^2]$	Inertia matrix of the plate
N	`double` [2x3]	[-]	Matrix collecting the direction of the normal ($+z$ and $-z$ in B)

The function (`dynamics.m`, see Alg. 3.1) simulates the dynamics and kinematics of the

flat plate given a set of initial conditions (initial yaw-pitch-roll angles[1] and initial angular velocities) and the data provided by `main_dyn.m`.

Algorithm 3.1 Dynamics Simulation

% PROTOTYPE: `[c_tot] = dynamics(var_input, param)`

% ——— %

- Extract the initial conditions $\boldsymbol{\omega_0}$ and $\mathbf{angles_0}$ from $\mathbf{var_{input}}$

- Get the initial DCM ($\mathbf{A_0}$) through `angle2dcm.m`

- Determine the direction of the Sun in ECI ($\hat{\mathbf{S}}_\mathbf{N}$) through ephemeris using `sun.m` [43]

- Integrate Euler equations using `Euler.m` to get dynamics ($\boldsymbol{\omega}$) and kinematics ($\mathbf{A}$) evolution

% Loop over the time:

for i =`1:length(t)` **do**

- Determine the Observer-Object versor in ECI ($\hat{\mathbf{O}}_\mathbf{N}$) (cf. Eq. 2.16)

% Orthonormalize the DCM to correct integration errors with the following scheme [21]:

$\mathbf{A_{ortho}}[i] = 3/2 \cdot \mathbf{A}[i] - \mathbf{A}[i](\mathbf{A}[i])'\mathbf{A}[i] \cdot 1/2$

% Switch to Object-Observer versor in B:

$\hat{\mathbf{O}}_\mathbf{B} = -\mathbf{A_{ortho}}[i]\hat{\mathbf{O}}_\mathbf{N}$

% Define the Object-Sun versor in B:

$\hat{\mathbf{S}}_\mathbf{B} = \mathbf{A_{ortho}}[i]\hat{\mathbf{S}}_\mathbf{N}$

% Loop over the surface normals (two for a flat plate):

for j = `1:size(N,1)` **do**

- Extract the normal as a row ($\mathbf{n_B} = \mathbf{N}[j,:]'$)

% In order to be seen, the area must be illuminated by the Sun and at the same time visible to the observer:

if $\hat{\mathbf{O}}_\mathbf{B} \cdot \mathbf{n_B} > 0$ **and** $\hat{\mathbf{S}}_\mathbf{B} \cdot \mathbf{n_B} > 0$ **then**

- Compute the projected area variation according to Eq. 2.17

else

- The projected area variation is equal to zero

end if

end for

end for

Return the projected area variation: $\mathbf{c_{tot}}$

[1] See Sec. 3.3 for details about the selection of the attitude parameters

This function considers the orbital motion of the debris (RA, DEC evolution), as well as the direction of the Sun for the given observation time. It outputs the area variation of the flat plate as a function of time. The inputs to `dynamics.m` are:

- `param`: structure collecting all the quantities defined in the table above
- $\mathbf{var_{input}}$: [1x6] row array collecting the initial conditions for the three angles ($\mathbf{angles_0}$) and three angular velocities ($\boldsymbol{\omega_0}$)

The reasoning behind `dynamics.m` follows the procedure exposed in Sec. 2.1; its output is the overall area variation determined in Eq. 2.17, which is saved as a *.txt* file in the same folder where TIG operates.

TIG - Tracklet Image Generator

As previously stated, TIG is a Python application for the generation of synthetic images containing streaks left by space debris in a real or simulated sky image [5]. For the purpose of this thesis, some modifications are done in order to integrate the program with the other portions of the code, while the main structure of the code is left untouched.
In general, TIG is composed of three main Python files:

1. *TIG_main.py*: the main file to run the program
2. *TIG_Classes.py*: the core of the code, where most of the computations and analyses are performed
3. *TIG_External_Functions.py*: file containing some useful functions used both by *TIG_Classes.py* and by *TIG_main.py*

For the modified version of TIG to work properly, three different files need to be fed to the program: one or more night sky FITS images, a *.txt* file of the area variation with time ($\mathbf{c_{tot}}$) and another *.txt* file containing the output from SCOOP [44] (see Figure 3.3). SCOOP (SpaceCraft and Objects Observation Planner) is a software, internally developed in the Department of Aerospace Science and Technology of Politecnico di Milano, that provides the relative Azimuth and Elevation for a given tracked object. In particular, the input to SCOOP is a Two-Line orbital Element set (TLE), an ASCII string featuring a list of orbital elements of the object in question, that can be propagated using the SGP4 algorithm [45].

```
---------------------- OBSERVATION WINDOWS ------------------------

Observation windows request by the user
Starting date:   06 Dec 2019 01:00:00 UTC
Ending date:     08 Dec 2019 01:00:00 UTC

-----------------------------------------------------------------

OBJECT ID:      #58     TLE epoch: 19335.18457772
-----------------------------------------------------------------
EPOCH (UTC)                    MIL_AZ [deg]    MIL_EL [deg]
-----------------------------------------------------------------
06 DEC 2019 02:32:33.000         151.597530        6.444600
06 DEC 2019 02:32:34.000         151.511420        6.401620
06 DEC 2019 02:32:35.000         151.425520        6.358640

...

06 DEC 2019 02:35:03.000         140.750360        0.034180
*****************************************************************

OBJECT ID:      #117     TLE epoch: 19335.20705967
-----------------------------------------------------------------
EPOCH (UTC)                    MIL_AZ [deg]    MIL_EL [deg]
-----------------------------------------------------------------
06 DEC 2019 02:49:52.000          14.765110        7.202210
06 DEC 2019 02:49:53.000          14.885980        7.161560
06 DEC 2019 02:49:54.000          15.006560        7.120870
...
```

Figure 3.3: An example of a SCOOP text output file, displaying the relevant information of the selected passages [5].

In this work, the input images are taken from the Online Digitized Sky Surveys server at the European Southern Observatory (ESO) archive that gives access to the Digitized Sky Survey 1 and 2 (DSS1 and DSS2) produced at the Space Telescope Science Institute through its Guide Star Survey group [46]. The images (see Figure 3.4) are based on photographic data obtained using the Oschin Schmidt Telescope on Palomar Mountain (California) and the UK Schmidt Telescope in Australia.

Figure 3.4: Example of a night sky image surveyed from DSS1. The server needs the central coordinates (RA, DEC) and the image size to output the FITS image.

TIG original workflow [5] can be summarized as follows:

1. Definition of three variables: Img, Mask and Passage

   -> Img: information on the input image

   -> Mask: features related to the output mask

   -> Passage: information on the passages contained in SCOOP output

2. Passage extraction: Az and El at each time instant of a random passage are read from SCOOP output; initial and final conditions of the extraction are randomly selected

3. Noise addition: random noise is added for simulated images

4. Trail placement and colouring: the number of pixels that form the tracklet for the considered random passage are determined; the tracklet in the image is randomly positioned and coloured

5. Other image processing operations: equalization, star fitter, vignetting, and mask creation

In this work, the main modifications regarded the randomness of many operations, as well as the pixel selection and colouring.
Firstly, the passage selection is fixed so that the (Az, El) evolution of the streak is always the same for the current run.
Then, the selection of pixels is modified, in order to properly account for the area variation. The idea is that, at least in the central part of the streak, the variation has to be faithfully reproduced. In order to characterize and justify this modification, the method followed to associate the pixels to a given passage must be explained.
Once the arrays of successive Az and El coordinates evolving in time are extracted from the SCOOP output text file, they are interpolated with piecewise Lagrange second-order polynomials (f_i) (see Figure 3.5) and a start and end point are selected (red dots in the image).
In order to determine the number of pixels associated to the angular evolution, the ratio between the resolution of the image and the FOV is computed; then this pixel per degree ratio ($r_{p/d}$) is used in the following equation:

$$n_{px} = \sqrt{(Az_{end} - Az_{start})^2 + (El_{end} - El_{start})^2} \cdot r_{p/d} \tag{3.1}$$

Where n_{px} is the number of pixels required for the streak in question.

Therefore, the arrays of Az or El evolution (**Az** and **El**, respectively) can be scaled with the new number of components given by n_{px}. If **Az** is built in this way, then **El** can be determined by evaluating the interpolating function in **Az**.

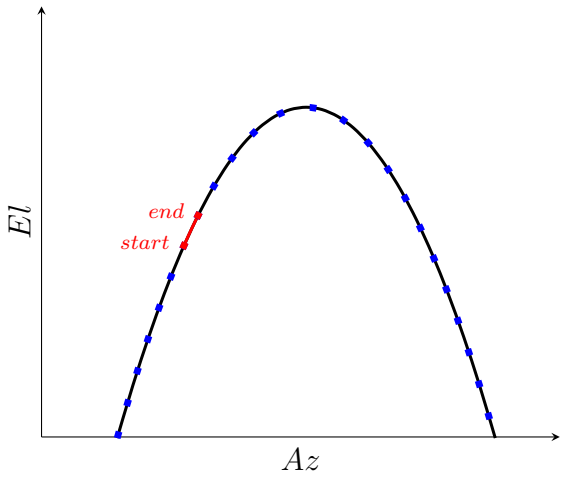


Figure 3.5: The quadratic interpolating function (in black) with Lagrangian basis on the passage points (in blue). The red portion accounts for the randomly chosen trail to be placed in the image [5].

Once these two arrays are defined, $r_{p/d}$ is used to determine the corresponding pixel coordinates with respect to the image center in horizontal ($\mathbf{pix_h}$) and vertical ($\mathbf{pix_v}$) direction. In order not to have all the streaks in the center of the image, a vertical (k_v) and horizontal (k_h) random shifts are introduced. For the same reason as the random passage extraction, in this work, the shifts are set as fixed values by the user.
With this procedure, the streak pixels are identified as couples of (x,y) coordinates in the image that, when rounded, can sometimes repeat. Clearly, this could be an important issue when considering that each value of the input area variation will be associated to a pixel coordinates couple. In this way, there would be the risk to write on the same pixel (same (x,y) coordinate) two consecutive values, thus losing the correct variation evolution. Therefore, all the repeating couples in the [$\mathbf{pix_h}$, $\mathbf{pix_v}$] array are deleted according to Alg. 3.2.
As for the colouring, the area variation array ($\mathbf{c_{tot}}$) coming from the dynamics simulation has to be scaled according to Eq. 2.18. Therefore, a multiplicative factor k and a shift n are defined as follows:

$$\begin{cases} k : \text{set by the user, boundaries defined by } \texttt{BITPIX} \text{ and } \texttt{BZERO/BSCALE} \\ n : \text{pixel mean values in the considered image} \end{cases} \tag{3.2}$$

In particular, the choice of n allows to reduce the number of variables of the optimization problem (cf. Sec. 3.3).
Moreover, some noise needs to be introduced in the tracklet colouring process. In particular, the noise is summed to the luminosity expression and determined as a Gaussian noise with zero mean and a user-defined standard deviation (std). Therefore, the expression of the luminosity, i.e. synthetic light curve ($\mathbf{L_{syn}}$), is:

$$\mathbf{L_{syn}} = \mathbf{c_{tot}} \cdot k + n + noise(std) \tag{3.3}$$

Algorithm 3.2 Delete Repeating Couples

- Stack vertically rounded $\mathbf{pix_h}$ and rounded $\mathbf{pix_v}$ to get a [n_{px}x2] matrix (**points**)
% Loop over the rows (n_{px}):
for `i=1:range(`$n_{px} - 1$`)` **do**
  % Check if current row is equal to successive one:
  if **points**`[i, :] == points[i + 1, :]` **then**
    - Collect the row index in a vector ($\mathbf{del_{row}}$)
  end if
end for
- Delete the rows inside **points** according to $\mathbf{del_{row}}$ and obtain $\mathbf{points_{new}}$

Now, for each point identified in $\mathbf{points_{new}}$, the old pixel value is replaced by the corresponding value of $\mathbf{L_{syn}}$. In this way, a thin streak of varying luminous pixels is obtained. In order to account for atmospheric seeing and resolution issues (cf. Sec. 2.2), a randomly selected thickness variation ($\mathbf{thick_{var_i}}$) between a minimum ($thick_{min}$) and a maximum ($thick_{max}$) value is given to the streak (see Figure 3.6).

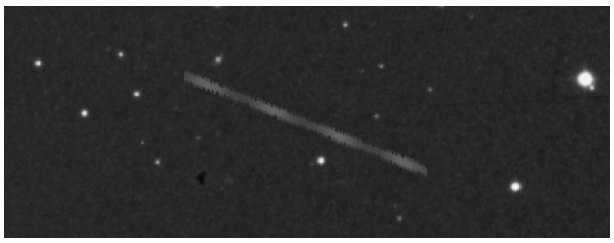
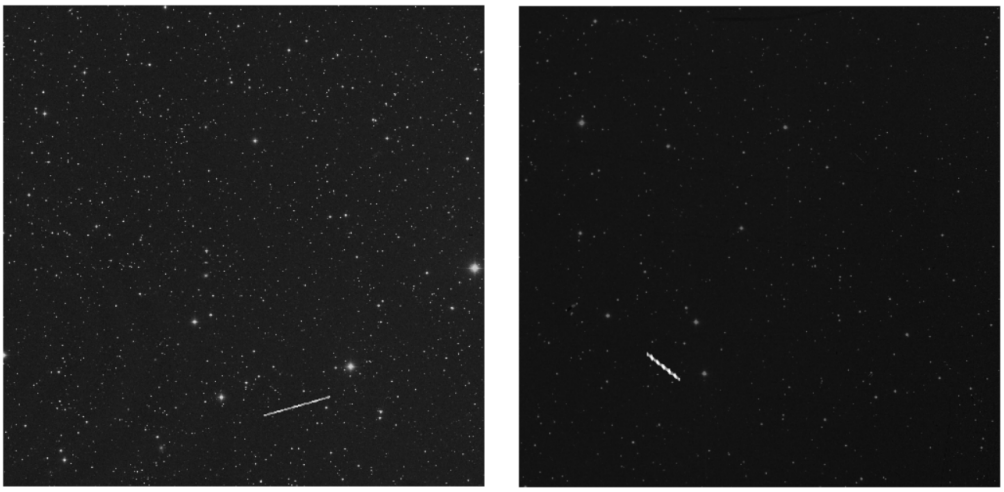


Figure 3.6: Detail on streak's thickness variation. In case of non-saturated one, the variation in thickness is assigned randomly.

Then, four arrays are defined:

- **vect**: array ranging from minus to plus $\mathbf{thick_{var_i}}$
- $\mathbf{vect_2}$: array ranging from 2·min(**vect**) to 2·max(**vect**), excluding the pixels already present in **vect**
- $\mathbf{vect_3}$: array ranging from $1.5 \cdot \min(\mathbf{vect_2})$ to $1.5 \cdot \max(\mathbf{vect_2})$, excluding the pixels already present in $\mathbf{vect_2}$
- $\mathbf{vect_4}$: array ranging from $1.5 \cdot \min(\mathbf{vect_3})$ to $1.5 \cdot \max(\mathbf{vect_3})$, excluding the pixels already present in $\mathbf{vect_3}$

These arrays specify pixels color around $[\mathbf{pix_h}, \mathbf{pix_v}]$. In particular, it is necessary to consider two distinct conditions, whether the streak develops more horizontally or vertically. The corresponding colouring is performed as it is outlined in Alg. 3.3 and the outcome can be seen in Figure 3.7.
In this new version, TIG can represent saturated or partially saturated streaks (Figure 3.7). As seen in Sec. 2.3, saturation entails an increase in the smearing of the streak itself due to the bleeding effect. Moreover, the pixel values must be clipped to the maximum value (saturation threshold, *sat*) according to the input sky image (following `BZERO` and `BSCALE` indications, cf. Sec. 2.2). In order to render these effects, the central part of the tracklet is left almost all saturated (depending on the values coming from $\mathbf{L_{syn}}$), while also $\mathbf{vect_3}$ and $\mathbf{vect_4}$ are used to guarantee a correct enlargement corresponding to saturated central pixels and an overall thickening of the streak (see Alg. 3.4).



(a) (b)

Figure 3.7: Synthetic Images from TIG: **(a)** is non saturated, while **(b)** is the saturated one.

Algorithm 3.3 Streak Colouring

% PROCEDURE: colour pixels of matrix of the image ($\mathbf{img_{array}}$) according to the pixels identified by $\mathbf{pix_v}$ and $\mathbf{pix_h}$
% ——— %
% Check if the streak is developed more horizontally or vertically:
$hor = max(\mathbf{pix_v}) - min(\mathbf{pix_v}) < max(\mathbf{pix_h}) - min(\mathbf{pix_h})$
if *hor* **then**
  - Delete repeating horizontal coordinates (see Alg. 3.2): avoid overwriting
else
  - Delete repeating vertical coordinates (see Alg. 3.2): avoid overwriting
end if
% Define the arrays for the thickness:
for $i =$ `1:size(`$\mathbf{pix_v}$`)` **do**
  % Check if $\mathbf{thick_{var_i}}[i]$ is even or odd:
  if $\mathbf{thick_{var_i}}[i]\%2 ==$ `0` **then**
    $\mathbf{vect} =$ `range`$(-\mathbf{thick_{var_i}}[i]/2, \mathbf{thick_{var_i}}[i]/2)$
  else
    $\mathbf{vect} =$ `range`$((-\mathbf{thick_{var_i}}[i]/2 + 1), (\mathbf{thick_{var_i}}[i]/2 + 1))$
  end if
  - Build $\mathbf{vect_2}$, $\mathbf{vect_3}$ and $\mathbf{vect_4}$
  % Differentiate saturated and non-saturated condition:
  if $\mathbf{L_{syn}}[i] < sat$ **then**
    % Consider vertical development:
    if *hor* **then**
      % Colour the central part of the streak:
      for n `in` $\mathbf{vect}$ **do**
        $\mathbf{img_{array}}[\mathbf{pix_v}[i] + n][\mathbf{pix_h}[i]] = \mathbf{L_{syn}}[i]$
      end for
      % Colour pixels above and below the central pixel with decreasing intensity:
      for r `in` $\mathbf{vect_2}$ **do**
        $\mathbf{img_{array}}[\mathbf{pix_v}[i] + r][\mathbf{pix_h}[i]] = \mathbf{L_{syn}}[\mathbf{i}] * (1 - 0.15 * |r|/2)$
      end for
    else
      % Consider horizontal development: same procedure, loop over horizontal pixels
    end if
  else
    - Saturated streak: check Alg. 3.4
  end if
end for

Algorithm 3.4 Streak Colouring - Saturated

% PROCEDURE: colour pixels of matrix of the image ($\mathbf{img_{array}}$) according to the pixels identified by $\mathbf{pix_v}$ and $\mathbf{pix_h}$ with saturated values in $\mathbf{L_{syn}}$

% ——————————————————————————————— %

if $\mathbf{L_{syn}}[i] >= sat$ **then**

  % Distinguish between vertical or horizontal development:

  if *hor* **then**

    % Saturate the central part of the streak:

    for n `in` **vect do**

      $\mathbf{img_{array}}[\mathbf{pix_v}[i] + n][\mathbf{pix_h}[i]] = sat$

    end for

    % Colour pixels above and below the central ones with decreasing intensity (clip to *sat* if the values are still above *sat*):

    for r `in` $\mathbf{vect_2}$ **do**

      $\mathbf{img_{array}}[\mathbf{pix_v}[i] + r][\mathbf{pix_h}[i]] = \mathbf{L_{syn}}[\mathbf{i}] * (1 - 0.03 * |r|)$

    end for

    % Keep colouring pixels above and below with decreasing intensity (clip to *sat* if the values are still above *sat*):

    for s `in` $\mathbf{vect_3}$ **do**

      $\mathbf{img_{array}}[\mathbf{pix_v}[i] + s][\mathbf{pix_h}[i]] = \mathbf{L_{syn}}[\mathbf{i}] * (1 - 0.06 * |s|)$

    end for

    % Check if farther pixel coloured is still saturated:

    if $\mathbf{img_{array}}[\mathbf{pix_v}[i] + s][\mathbf{pix_h}[i]] == sat$ **then**

      % Keep on decreasing the intensity:

      for p `in` $\mathbf{vect_4}$ **do**

        $\mathbf{img_{array}}[\mathbf{pix_v}[i] + p][\mathbf{pix_h}[i]] = \mathbf{L_{syn}}[\mathbf{i}] * (1 - 0.08 * |p|)$

      end for

    end if

  else

    - Consider horizontal development: same procedure, loop over horizontal pixels

  end if

end if

Trial Run and Real Run

Now, it is possible to better explain the logic behind TR and RR. As seen before, `dynamics.m` works with some information that is either stored in FITS image header

or computed by SAP. Therefore, when simulating the dynamics from scratch, this information is not available from the beginning.
In TR (see Figure 3.8), a trial dynamics is simulated with fake information regarding all the inputs to `dynamics.m` and with random initial conditions. In particular, the time vector of the simulation is set to be compliant with the SCOOP output file. The Sun direction can either be computed through ephemeris with a fake UTC or directly set with artificial RA and DEC of the Sun. In this way, a sufficiently long area variation array ($\mathbf{c_{tot}^{TR}}$) is obtained and ready for TIG. Hence, a fixed passage selection and extraction of fixed length and a predetermined positioning of the streak is set, so that in both TR and RR the streak will be exactly the same in terms of length and position. Then, TIG is run with the fake colouring based on $\mathbf{c_{tot}^{TR}}$, with a background sky image containing some information needed by `dynamics.m` for the RR (UTC, φ, and ϑ).

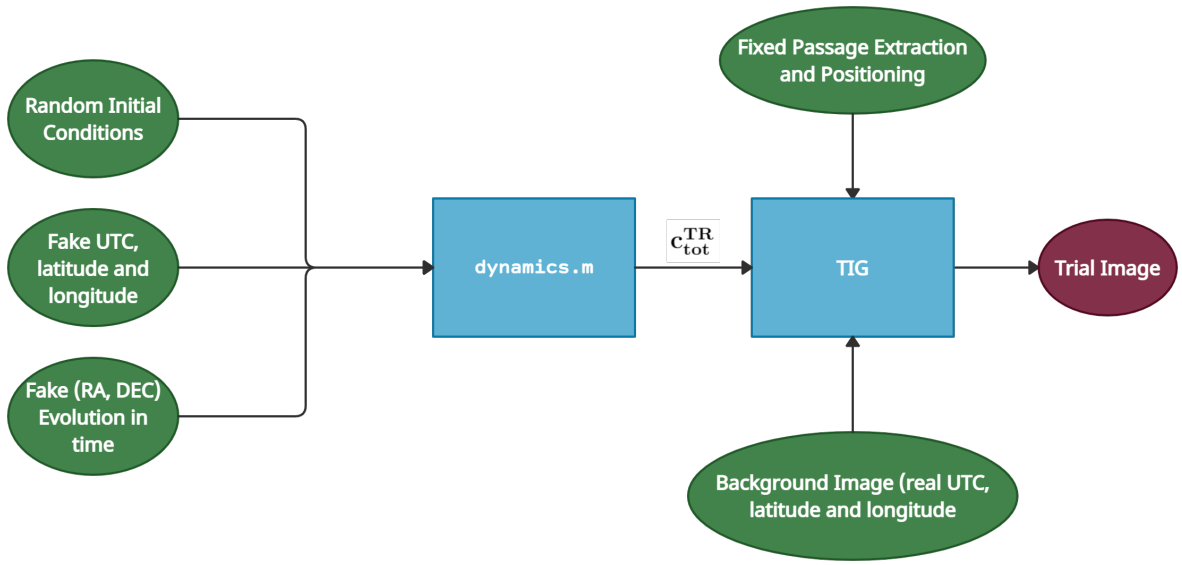


Figure 3.8: Block Scheme of the Trial Run. The output (Trial Image) is given to SAP.

The last bit of missing information is the **Ra** and **Dec** evolution of the tracklet. Since the SCOOP output refers to an observer location that is different from the one stored in the FITS file header, TIG cannot be used to retrieve **Az** and **El** and then to transform them into **Ra** and **Dec** (cf. Appendix A). On the other hand, SAP is developed to analyze any kind of image (both real and synthetic) and is able to identify and extract the streak from the image itself. Therefore, in TR, SAP is used on the trial image exclusively to extract **Ra** and **Dec** and the vector of time **t** (cf. Sec. 3.2 for more details).
In this way, all the inputs to `dynamics.m` and `main_dyn.m` are now available for RR to be performed. In RR (see Figure 3.9), a finer evaluation of the initial conditions is done

in order to obtain a streak that is correctly visible in the image generated by TIG.

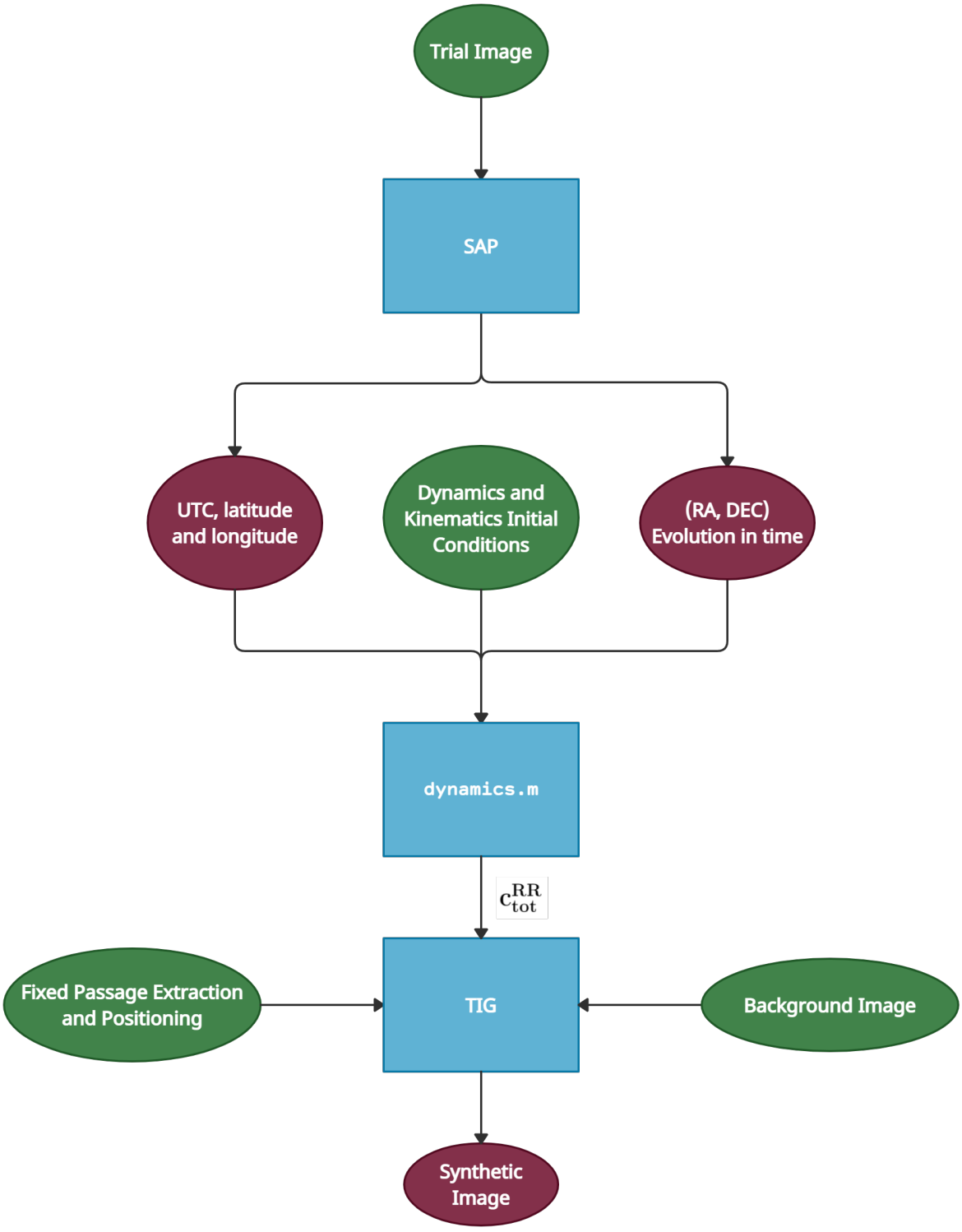


Figure 3.9: Block Scheme of the Real Run. The output is the synthetic image that will be given to SAP to perform a complete analysis.

Therefore, the initial direction of the Observer-Object versor and Observer-Sun versor is evaluated in TH, to see if the extracted information is actually representing a feasible situation (see Figure 3.10). If the Sun versor is in a somehow incompatible illumination condition (i.e. above the horizon or with a too low angle with respect to the object) an artificial direction is forced in order to have the object lit and visible on ground.

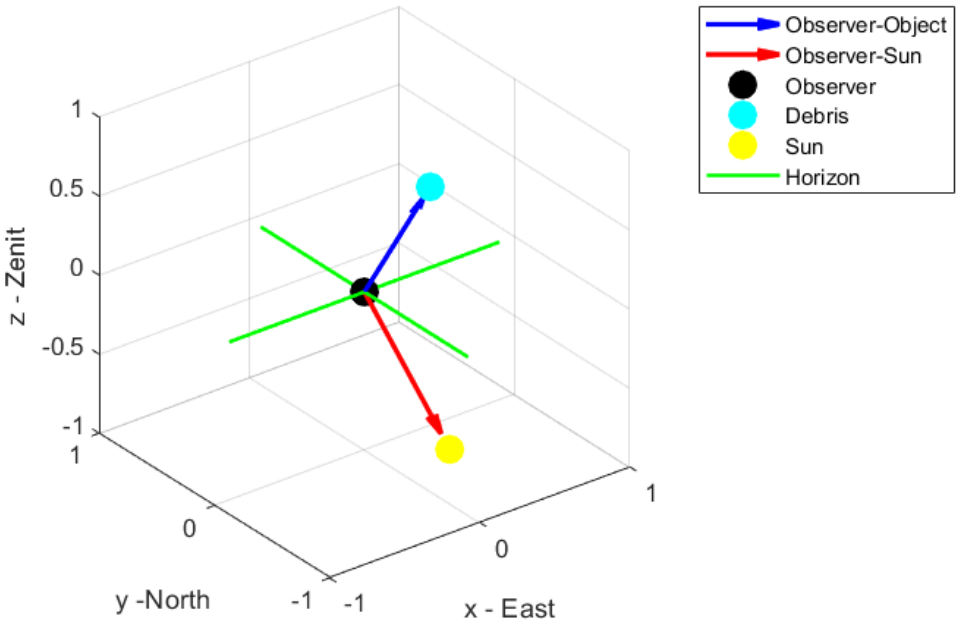


Figure 3.10: Initial directions between Observer, Object and Sun in TH frame. Feasible situation with Sun below the horizon illuminating the debris that is visible by the observer, for a given UTC, latitude, and longitude.

Clearly, this expedient is used only in the case of synthetic images, where the presence of a light-reflecting debris is forced into the image and the illumination coming from the Sun for the given observation time may not be the most suitable one; with real images, the direction of the Sun can be directly computed through ephemeris.
Once the initial directions are identified, the user needs to set the initial attitude parameters and angular velocities fed into `dynamics.m`. This choice is done in order to have an overall $\mathbf{c}_{\mathbf{tot}}^{\mathbf{RR}}$ variation always different from zero. This implies that both γ and β are always lower than 90° for a given normal (see Sec. 2.1) or that the rotation is sufficiently fast that the non-reflecting zero-thickness sides are not seen by the observer.
Finally, $\mathbf{c}_{\mathbf{tot}}^{\mathbf{RR}}$ is simulated with the correct **Ra** and **Dec** arrays. Then, this quantity is fed to TIG, which generates the final image according to the desired area variation trend using the same parameters as in TR.

The overall process is quite lengthy, but in this way all the physics behind the procedure is preserved, which is particularly important when evaluating streaks in real images.

3.2. SAP - Streak Analysis Pipeline

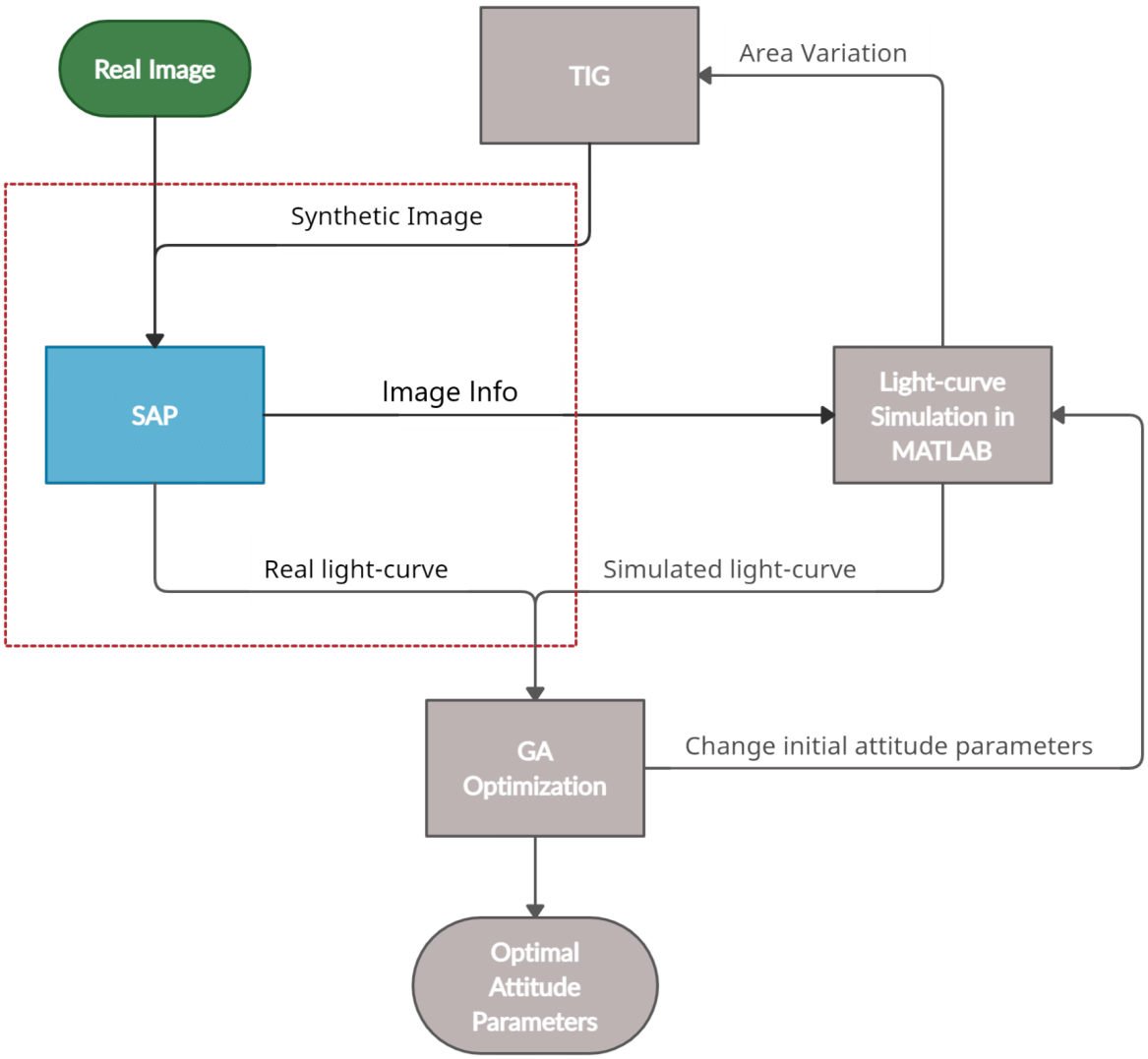


Figure 3.11: Thesis Workflow: SAP

The Streak Analysis Pipeline (SAP, Figure 3.11) is a Python application that performs the following main tasks (see Figure 3.12):

1. Detection and extraction of the streak(s) from the input FITS file
2. Extraction of central line luminosity variation (light curve in terms of light intensity)

3. Evaluation of the streak thickness
4. Detection of the sources in the image and control star (lowest magnitude) identification and aperture photometry
5. Catalogue query and catalogue matching with the field in the image
6. Aperture photometry on the streak and light curve extraction (in terms of apparent magnitude)

Tasks from 4 to 6 are performed in the case of saturated streaks (cf. Figure 3.12).

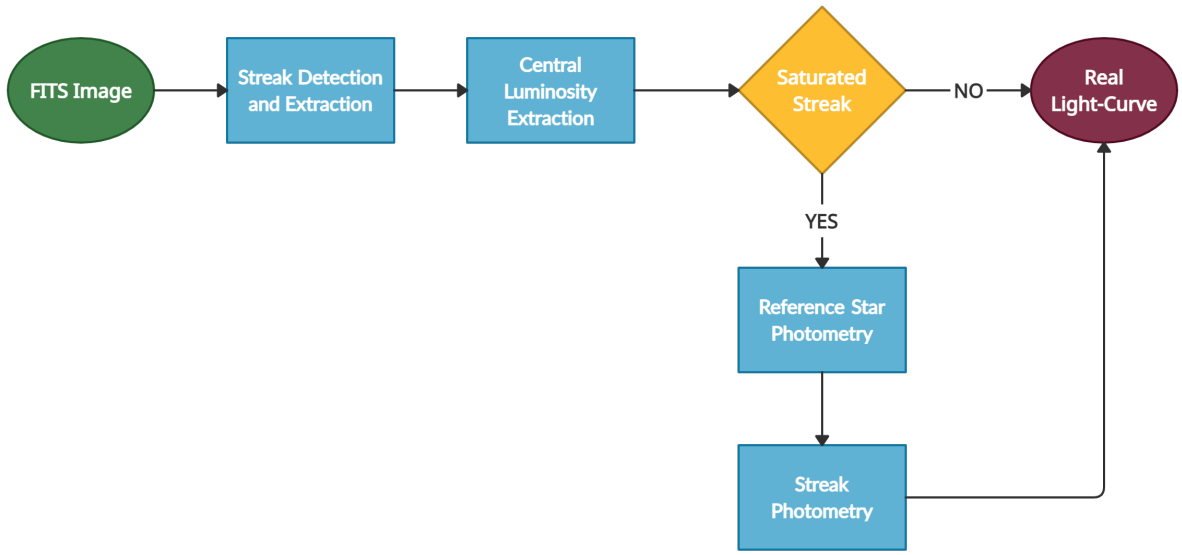


Figure 3.12: Block Scheme of SAP

The code consists of three Python files:

- *SAP_Main.py*: the main file to run the program
- *SAP_Classes.py*: the core of the code, where most of the computations and analyses are performed
- *SAP_Ext_functions.py*: file containing some useful functions used both by *SAP_Classes.py* and by *SAP_Main.py*

The overall code is conceived as an independent pipeline that can work with any FITS image that contains one or more streaks. For the purpose of this work, the output of SAP is saved in a folder where the MATLAB optimization operates. Moreover, for each run on the image different HTML pages for diagnostics are created.

Streak Detection and Extraction

Among the available Python packages for streak detection, `ASTRiDE` (Automated Streak Detection for Astronomical Images) [47] is selected, since it implements the functionality of streak detection that provides a border for each object (boundary-tracing or contour-tracing).
Moreover, `ASTRiDE` employs an algorithm for quantifying the shape of each border to assess whether or not it belongs to a streak. In order to do so, the steps followed are:

1. Background removal: `ASTRiDE` removes the background noise from the FITS image using the `photutils` package [48].
2. Contour map: border detection of the streak controlled by a threshold value
3. Morphology of the detected border: star-like sources are removed by considering different morphological parameters that `ASTRiDE` defines for each identified border

The different parameters defined for each streak are:

Quantity	Dimension	Description
SF	`double` [1x1]	Shape Factor. Circularity of the streak ($SF = 1$ for a circle)
RD	`double` [1x1]	Radius Deviation. Approximated deviation from roundness
$area$	`double` [1x1]	Area inside the border in px^2
x	`double` [Mx1]	X coordinate of the M points of the streak
y	`double` [Mx1]	Y coordinate of the M points of the streak
x_{min} and x_{max}	`doubles` [1x1]	Minimum and maximum X coordinates of the streak
y_{min} and y_{max}	`doubles` [1x1]	Minimum and maximum Y coordinates of the streak
x_c and y_c	`doubles` [1x1]	X and Y coordinates of the streak centroid
$slope$	`double` [1x1]	Angular coefficient of the streak with respect to the horizontal
$intercept$	`double` [1x1]	Intercept of the streak with the vertical axis

The output is organized as follows:

- *.txt* file where the main features of the detected streaks are organized. If more than one streak is detected, a connectivity parameter notifies the user whether the streaks are linked one over the other depending on their slopes
- Different figures: an image showing all the streaks identified (see Figure 3.13) and close-ups of each of them

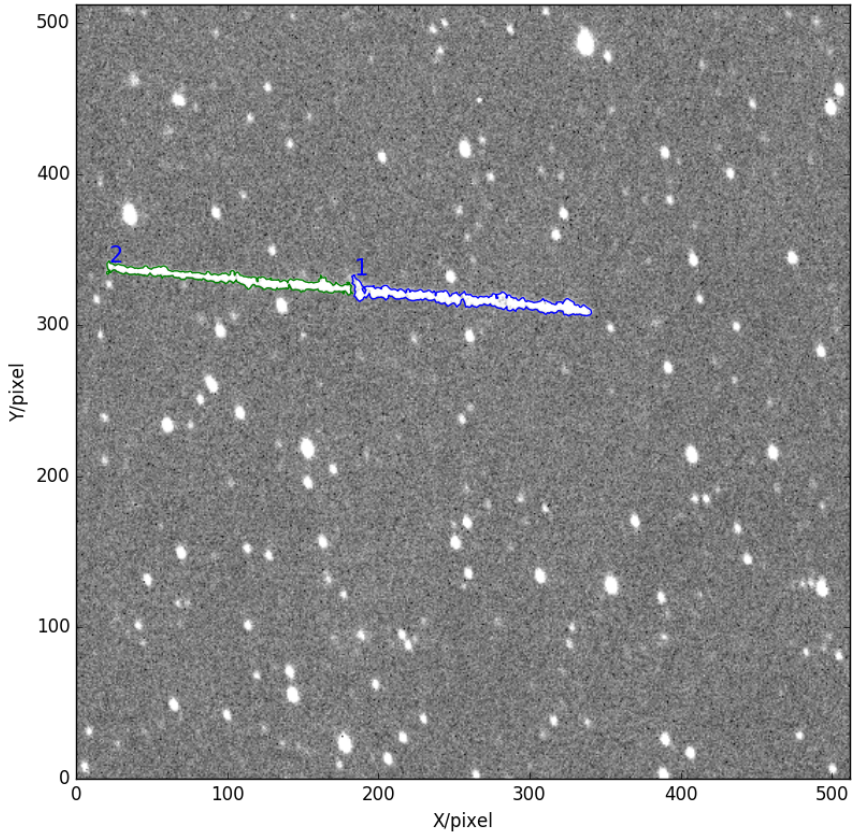


Figure 3.13: Example of `ASTRiDE` *all.png* output image. Each identified streak is marked with different numbers and colours [47].

Algorithm 3.5 Streak Extraction

- Extract the image array data from the FITS file ($\mathbf{img_{array}}$)
% Collect the points of the contour identified by `ASTRiDE`:
$\mathbf{points_2} = [\mathbf{x}, \mathbf{y}]$
- Create a `True/False` boolean matrix (**mask**) with the same dimension of $\mathbf{img_{array}}$ with `True` values in correspondence of $\mathbf{points_2}$
% Extract from $\mathbf{img_{array}}$ the pixels identified by **mask**:
$\mathbf{out}[\mathbf{mask}] = \mathbf{img_{array}}[\mathbf{mask}]$

All the information relative to the streak is also accessible in Python. If more than one streak is identified, SAP automatically selects the largest *area* streak. Once the streak is identified in the image, SAP proceeds to extract the interested pixels to be analyzed (cf. Alg. 3.5).
Therefore, the pixels of the streak become the only non-zero ones in **out** as it can be seen in the figure below:

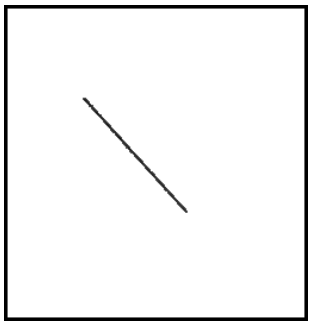


Figure 3.14: Detail of the streak extracted in **out**. All pixel values are null, but the streak ones. In the image, the rendering is in negative colours to better visualize the streak.

Central Intensity Extraction

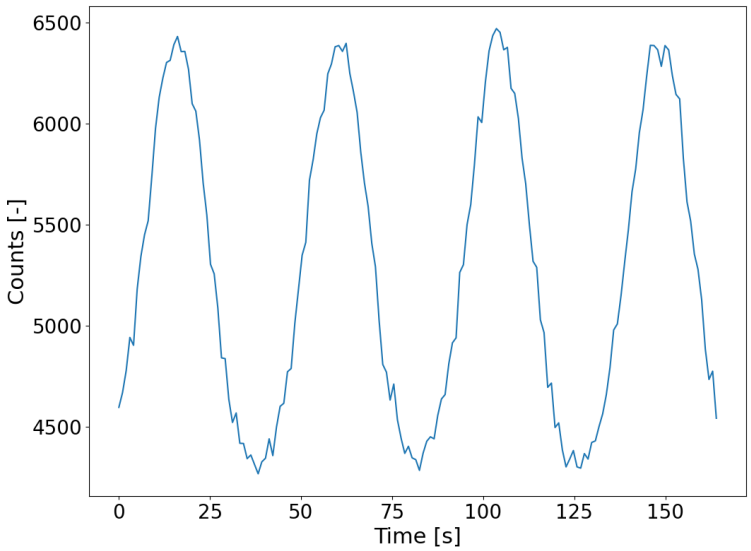


Figure 3.15: Typical light curve extracted by SAP of a synthetic streak as a variation of the central luminosity in time.

In the case of non-saturated streaks, the light curve extraction can be carried out by just considering the evolution of the pixel values along the streak central line (see Figure 3.15). In fact, this trend retains all the information contained in the supposed area variation, thanks to the scaling considered in Eq. 2.18.
First of all, the streak orientation must be defined in the same way as in Alg. 3.3, to check if the streak develops more horizontally or vertically. According to this distinction, the streak central line is analytically defined as a line with equation:

$$\begin{cases} y = m \cdot x + q & \text{if } hor \text{ is } \texttt{True} \\ x = y/m - q/m & \text{if } hor \text{ is } \texttt{False} \end{cases} \tag{3.4}$$

Therefore, depending on *hor*, x or y are chosen as dependent or independent variables, while the values of m and q are *slope* and *intercept*, respectively. In order to create two arrays ($\mathbf{x_{central}}$ and $\mathbf{y_{central}}$) that give the coordinates of the pixels along the central line, the boundary points (*start* and *end* for x or y coordinates) are defined according to Alg. 3.6.

Algorithm 3.6 Central Line Definition

% Consider distinct cases according to *hor*:
if *hor* **then**
% Set the horizontal boundaries:
$x_{start} = x_{min}$ and $x_{end} = x_{max}$
% Compute the dependent variable (y) with first case in Eq. 3.4:
$y_{start} = m \cdot x_{start} + q$
$y_{end} = m \cdot x_{end} + q$
- Check if $y_{start} < y_{end}$, otherwise switch them
- Compute n_x and n_y with Eq. 3.5 and the corresponding l_{streak}
% Build $\mathbf{x_{central}}$:
$\mathbf{x_{central}} = \texttt{linspace}(x_{start}, x_{end}, l_{streak})$
% Compute $\mathbf{y_{central}}$ accordingly:
$\mathbf{y_{central}} = m \cdot \mathbf{x_{central}} + q$
else
- Perform similar computation with y as the independent variable
end if

Then, the number of pixels in the horizontal and vertical direction needed to get from

one streak extreme to the other are computed:

$$\begin{cases} n_x = x_{start} - x_{end} \\ n_y = y_{start} - y_{end} \end{cases} \tag{3.5}$$

The overall length of the streak (l_{streak}) is obtained as:

$$l_{streak} = \sqrt{n_x^2 + n_y^2} \tag{3.6}$$

Then, according to *hor*, the independent variable is built as an array ranging from the previously identified extremes and with l_{streak} elements inside, while the dependent one is again built according to Eq. 3.4 (check again Alg. 3.6 for the complete procedure).
Up until this point, all the computations are performed at sub-pixel level: when rounding the pixel coordinates in a matrix that collects $\mathbf{x_{central}}$ and $\mathbf{y_{central}}$ ($\mathbf{points_3}$), some repetitions may occur. Therefore, Alg. 3.2 is applied in the same way as it was done in Sec. 3.1. Thereby, repeating rows in $\mathbf{points_3}$ are deleted and the extraction of the central line is performed pixel by pixel without overwriting.

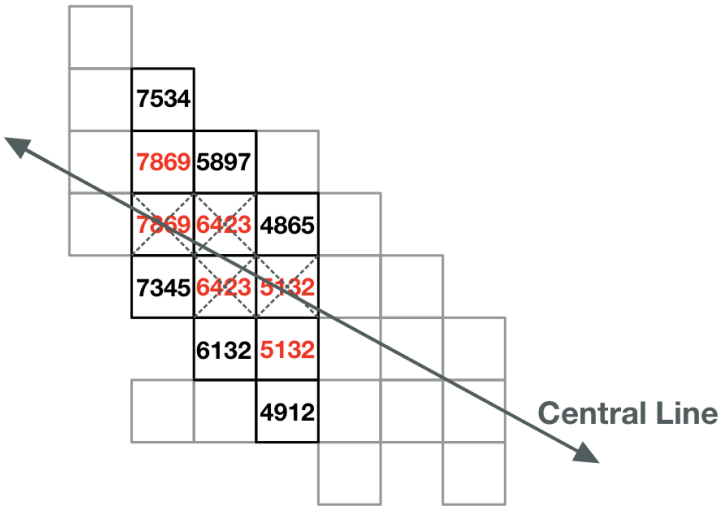


Figure 3.16: Visualization of repeating pixel values. The rounded coordinates given by the central line may extract the same value on a column (if horizontally developed) of a central pixel. The repeating value - at least for synthetic images, but also for real images due to smearing - is an issue when trying to extract the correct luminosity evolution.

Once the pixels of the central line are determined, their coordinates are used to directly extract the values of the pixels from the image. Two main issues may arise with this procedure:

- The extremes identified by `ASTRiDE` may not be precise, leading to the reading of pixels not really part of the streak.
- In the extraction along the central coordinates, the same pixel value may be read on different pixel coordinates due to the streak smearing. This is an important issue mainly for synthetic images, where, whether horizontally or vertically developed, the streak must be read without repetitions in order to correctly reproduce the input area variation evolution.

The solution of the first problem is simply checking if the extracted pixel values differ too much from each other or the associated brightness is too low. As regards the second issue, whenever a repeating value is part of the luminosity evolution, it is deleted and the associated coordinate is not considered anymore (see Figure 3.16).
In this way, the luminosity evolution ($\mathbf{lum_{centre}}$) along the streak is determined. Now, in order to retrieve a proper light curve, the time information is needed. In the case of real images, the `EXPOSURE` field from the FITS header must be read and a time vector is built ranging from zero to `EXPOSURE`, with the same number of elements as $\mathbf{lum_{centre}}$. For synthetic images instead, an artificial time vector is built, ranging from zero to the number of elements of $\mathbf{lum_{centre}}$, but with a certain time step (see Alg. 3.7).

Algorithm 3.7 Time Array Definition

```
% Consider distinct cases whether the image is real or synthetic:
if sim then
    % Directly build the time array with a user-defined time step (dt):
    t = linspace(0, dt · length(lum_centre), length(lum_centre))
else
    - Extract the EXPOSURE field from the header (t_exp)
    % Build the time array:
    t = linspace(0, t_exp, length(lum_centre))
end if
```

This approach succeeds only with non-saturated streaks; with saturated ones, only the streak central line coordinates are evaluated, while the light curve is determined in terms of apparent magnitude after all the photometry process.

Thickness Variation

The streak thickness variation is a useful information for many reasons. As for saturated streaks, it gives an indication on the different thickening due to the bleeding effect. For

non-saturated streaks instead, it can be useful to know if some link between the thickness variation and the smearing of the streak is present.
In order to evaluate the thickness, the following operations are performed:

1. Zero-padding of the mask with only streak illuminated pixels
2. Rotation of the zero-padded mask (check Alg. 3.8) so that the streak becomes horizontal
3. Evaluation of the thickness

Algorithm 3.8 Zero-Padded Image Rotation

% Pad the masked image data array with zeros on the sides:
$\mathbf{img_{padded}} = \texttt{pad}(\mathbf{out})$
- Rotate $\mathbf{img_{padded}}$ (avoid out-of-frame streak) according to *slope* to get the rotated padded image ($\mathbf{out_{rot}}$)
- Consider the origin (upper left-hand corner) in the original image ($\mathbf{c_{or}}$) and in the rotated one ($\mathbf{c_{rot}}$)
% Compute the streak centroid with respect to $\mathbf{c_{or}}$:
$\mathbf{xy_{or}} = [x_c, y_c] - \mathbf{c_{or}}$
- Rotate $\mathbf{xy_{or}}$ to get the streak's centroid in $\mathbf{out_{rot}}$ ($\mathbf{xy_r}$) through a 2D rotation matrix with an angle equal to *slope*
% Compute the streak centroid with respect to the centre of the image:
$\mathbf{xy_{rot}} = [x_c^{rot}, y_c^{rot}] = \mathbf{xy_r} + \mathbf{c_{rot}}$

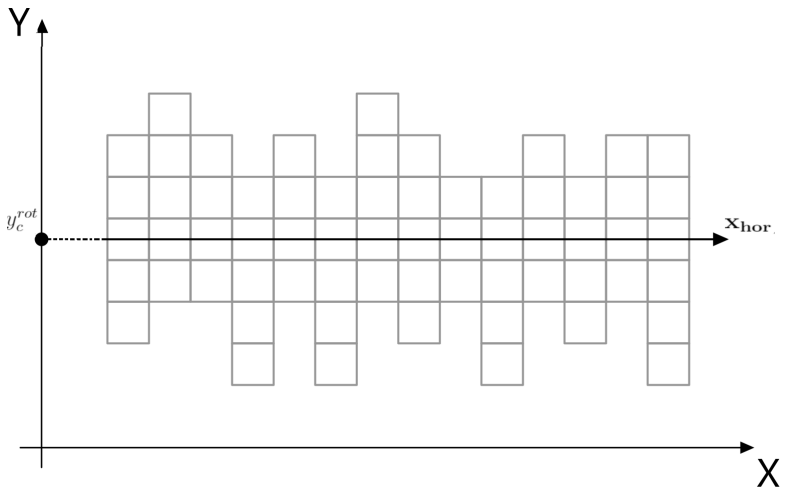


Figure 3.17: Visualization of the rotated streak pixels. The Y coordinate of the central line (y_c^{rot}) is constant, while the horizontal evolution ($\mathbf{x_{hor}}$) follows a straight line.

Then, a loop over the vertical direction, starting from each central line coordinate (see Figure 3.17), allows to determine how many pixels are not null (using **out**, see Alg. 3.9). It is worth noting that with this approach the streak is considered to be symmetric with respect to the central line.

Algorithm 3.9 Streak Thickness Evaluation

- Set the counter: $i = 0$
% Loop over the horizontal length of the streak:
while $i <$ `length`($\mathbf{x_{hor}}$) **do**
  - Set the half-thickness counter ($j = 0$)
  % Loop along vertical extension of the streak until a 0 is encountered:
  while $\mathbf{out_{rot}}[y_c^{rot} + j,\ \mathbf{x_{hor}}[i]] > 0$ **do**
    - Update the increment j: $j = j + 1$
  end while
  - Save the value of j in an array **J**
  - Update the horizontal pixel: $i = i + 1$
end while
- Get the full-thickness evolution value: $\mathbf{thick_{var}} = 2 \cdot \mathbf{J}$

Control Star Photometry

In case of saturated streaks, the light curve loses the information about its physical evolution. Therefore, a reference source in the field has to be identified and studied to determine the apparent magnitude of the streak (cf. Sec. 2.2).
The search and analysis of this control star is performed by using different astronomic Python packages (mainly `astropy` [49] and `photutils`).
The main steps are:

1. Source finding in the field. `DAOStarFinder` (based on the DAOPHOT algorithm [50]) source extractor is used on the background-subtracted image with a mask applied onto the streak (so that it is not considered) and with an initial FWHM set by the user.

2. After the catalogue matching (see below), identification of the lowest magnitude source in the field among the extracted ones. Cropping (with user-desired offset) of the image around the control star.

3. PSF aperture photometry (cf. Sec. 2.2) on the identified star.

The main output of this process is the flux of the control star ($flux_{ref}$), i.e. sum of all

the pixel values in the aperture.
This is the standard procedure for the aperture photometry of stars [51]. Nevertheless, some difficulties arose during the development of this part of the code. In particular, once the control star is identified, the cropping around its position does not guarantee that in the resulting image only one star will be present (especially for crowded fields, see Figure 3.18).

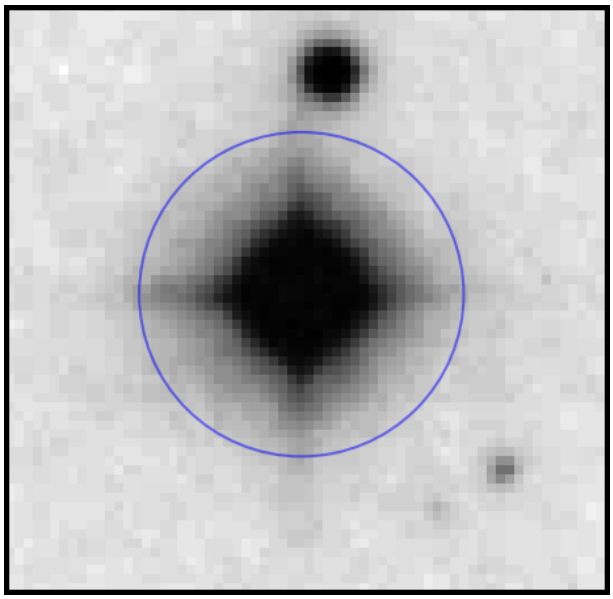


Figure 3.18: Control Star Photometry in a crowded field. If the catalogue matching is done properly, the control star will be at the centre of the cropped image and the aperture (blue circle) will be centred on it.

Algorithm 3.10 Control Star Position

- If more than one star is present in the cropped field, the centroid coordinates will be determined by the source extractor and stored in a matrix (**positions**)
% Loop over each star centroid coordinate:
for pos `in` **positions do**
  if pos is within a 5x5 pixel box around the centre of the cropped image **then**
    - Perform photometry on the selected source
  end if
end for

Moreover, a refinement of the source extraction proved to be necessary, especially in terms of FWHM. Therefore, another `DAOStarFinder` run is performed with a FWHM defined

with `aspylib astro` package [52] fitting a Gaussian circular PSF onto the background-subtracted sources in the cropped field. Finally, if the coordinates found through catalogue matching are correct, the control star will be surely near the centre of the cropped image (see Alg. 3.10) and the analysis can continue.
The overall process is represented in Figure 3.19.

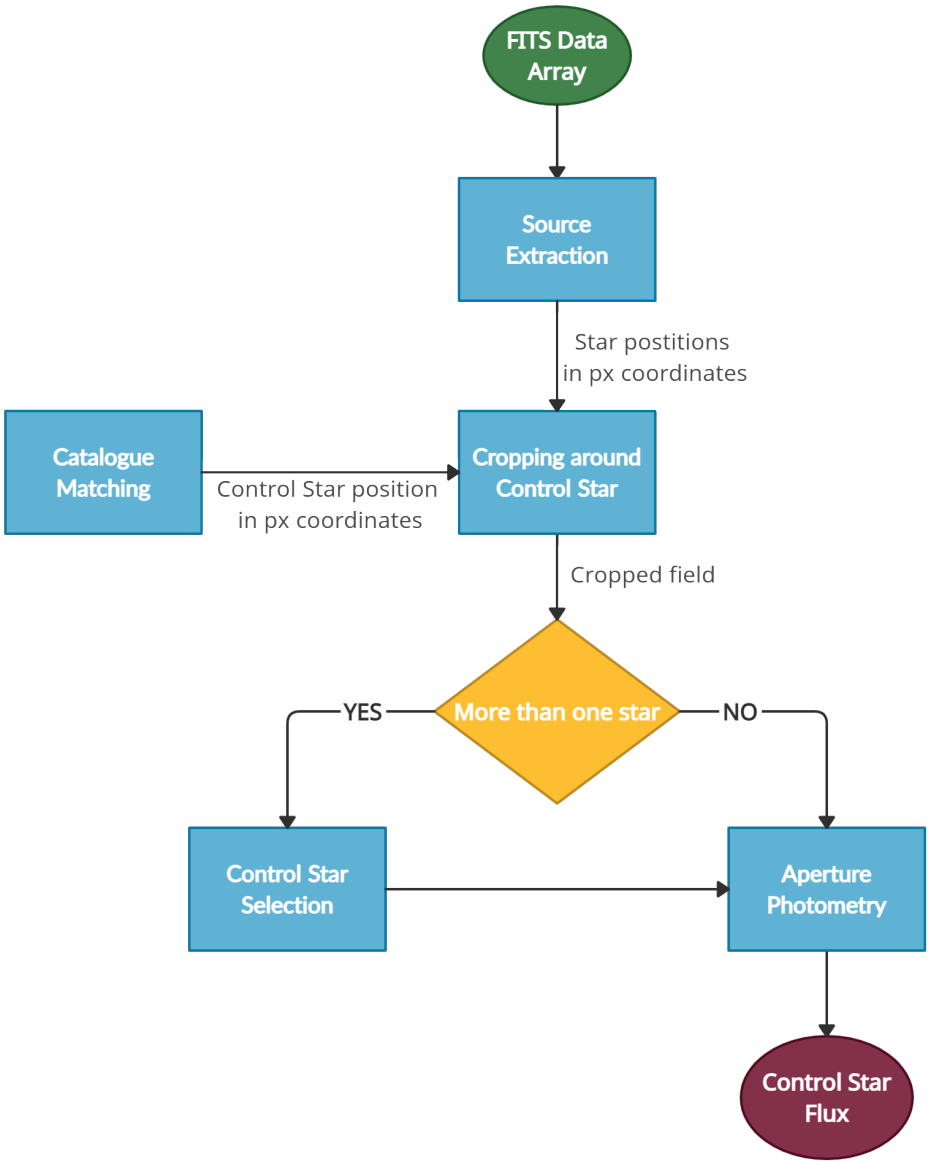


Figure 3.19: Block Scheme of the photometry of the control star. Catalogue query and matching is fundamental in order to determine the location of the control star in the field. The process leads to the evaluation of the flux of the reference star.

Catalogue Matching

As stated above, in order to identify the control star, its apparent magnitude needs to be determined. Such a quantity is usually stored in astronomical catalogues that can be queried online with different methods. The query returns the celestial coordinates (RA and DEC) of a number of sources in a given field, that can be compared with the ones extracted from the image. This procedure is called catalogue matching (cf. Sec. 2.2) and it is of the utmost importance for the proper definition of the control star magnitude.

Algorithm 3.11 Search Radius Definition

- Extract the `WCS` information from the header
- Extract the pixel size (pix_{size}) of the detector from the header (`XPIXSZ`[2])
- Get the focal length (FL) of the telescope from the header (`FOCALLEN`)

% Compute the pixel scale in $arcsec/px$ [53]:

$pix_{scale} = \frac{pix_{size}}{FL} \cdot 206.3$

- Get the image dimension (ID) from the header (`NAXIS1`)

% Get the angle covered by the image:

$dim = pix_{scale} \cdot ID$

% Define the search radius (in $arcsec$) as half that dimension:

$search_{rad} = dim/2$

A well-known query is the `ConeSearch` provided by the Python package `astroquery` [54]. This function requires the user to input the centre and the radius of the field in the image. While the central celestial coordinates can be retrieved from the header (`OBJCTRA` and `OBJCTDEC` fields, see Sec. 2.2), the search radius needs to be computed from some of the header information (see Alg. 3.11).
Then, in order to compare the catalogued sources with the extracted ones, a transformation from pixel to angular coordinates needs to be performed. A function `pix_to_RADEC` is therefore implemented: it uses the `WCS` header info (cf. Sec. 2.2) to transform pixel coordinates into world coordinates through the `pixel_to_world` function. Hence, with `astroquery`'s `match_to_catalog_sky` function, the distances between the positions of the extracted sources and the queried ones are computed, and the matching quality can be evaluated. Finally, the lowest magnitude star can be identified and its apparent magnitude stored as m_{ref}. Moreover, its celestial coordinates can be fed to the control star

[2]This is an unconventional field for FITS headers. Therefore, it can change its name depending on the software used to generate the FITS file.

photometry step, in order to perform the cropping (see Figure 3.20).

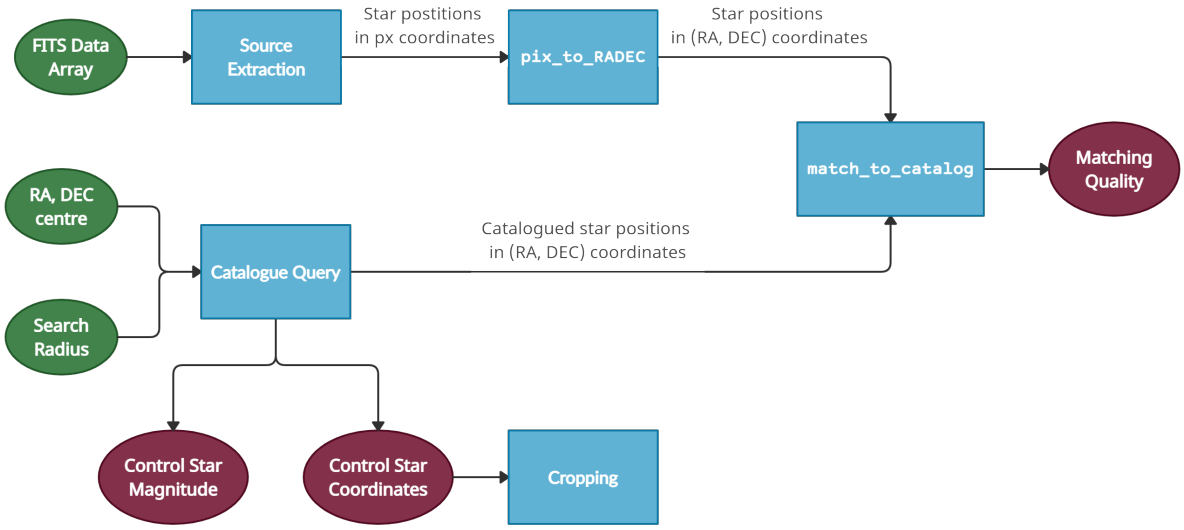


Figure 3.20: Block Scheme of the catalogue matching. The main outputs are the control star magnitude and coordinates. The matching quality is used to check the goodness of the process.

Streak Photometry

The last step left to perform is determining the streak flux variation. A series of rectangular apertures are applied pixel by pixel, sliding horizontally or vertically according to *hor*, along the central line of the streak.

Therefore, horizontal or vertical one-pixel-wide rectangles are placed along the coordinates identified by $\mathbf{x_{central}}$ and $\mathbf{y_{central}}$, with the length given by the maximum value of $\mathbf{thick_{var}}$ (see Figure 3.21). In fact, considering that the analysis is performed for a background-subtracted image, the contribution of the background pixels is almost null, and the streak is certainly contained inside the apertures. The flux variation in each aperture is stored in an array ($\mathbf{flux_{streak}}$).

In order to define the apparent magnitude variation, Eq. 2.21 is applied both to the flux of the control star and to the one of the streak:

$$\begin{cases} I_{m_{ref}} = -2.5 \cdot \log\left(\frac{flux_{ref}}{t_{exp}}\right) \\ \\ \mathbf{I_m} = -2.5 \cdot \log\left(\frac{\mathbf{flux_{streak}}}{T_{exp}}\right) \end{cases} \tag{3.7}$$

Where two different exposure times are considered. In the control star case, the exposure time (t_{exp}) coincides with the overall one of the image (whether real or synthetic). For the streak instead, the pixels in each aperture collect photons only for the instant of time when the object crosses that particular aperture; this implies that the corresponding exposure time T_{exp} is equal to t_{exp} scaled by the number of pixels that made up the streak, since one-pixel-wide rectangular apertures are being considered.

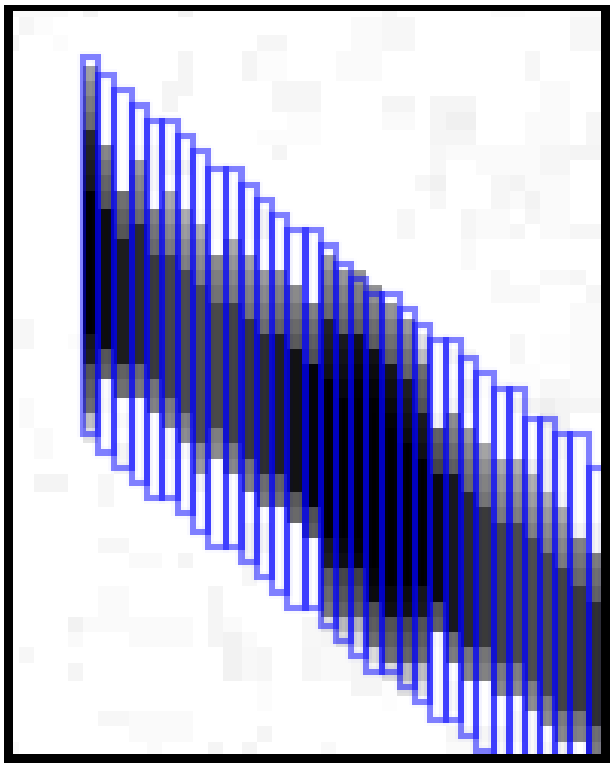


Figure 3.21: Detail of Rectangular apertures (in blue) on the streak. The width of each aperture is of one pixel, while the length is given by the maximum value of $\mathbf{thick_{var}}$. The background pixels give almost no contribution (image in negative colours).

Finally, the apparent magnitude evolution can be determined through Eq. 2.22:

$$\mathbf{m} = m_{ref} + \mathbf{I_m} - I_{m_{ref}} \tag{3.8}$$

SAP - Diagnostics and Output

As it can be seen, SAP is an application that performs multiple tasks and has different outputs. In order to track how SAP worked, three HTML diagnostics pages are produced for each run:

1. *streak_ diagnostics.html*: it contains the main results regarding the streak detection and extraction.
2. *phot_ diagnostics.html*: it contains the main results concerning the control star photometry
3. *streak_ phot_ diagnostics.html*: it contains the main results coming from the photometry of the streak

The last two diagnostics pages are returned only if the streak is saturated.
As for the outputs sent to MATLAB, two *.csv* tables and a *.txt* file are stored:

- A first table (*table.csv*) storing the RA and DEC coordinate evolution (**Ra** and **Dec**) of the streak central line (`pix_to_RADEC` applied to $\mathbf{x_{central}}$ and $\mathbf{y_{central}}$) and the associated time vector ($\mathbf{t}$).
- A second table (*dat.csv*) storing useful information read from the FITS header image: UTC, φ and ϑ. It also embeds the mean value of pixels in the image (n) and the saturation threshold (sat).
- A text file that features the time evolution of the central luminosity (*lum.txt*) or of the apparent magnitude (*mag.txt*) for saturated streaks. In both cases, the real extracted light curve is given to the optimization part of the code.

3.3. Optimization

The last part of the procedure to extract the attitude state of the approximated square flat plate is the optimization between the real light curve extracted from SAP (cf. Sec. 3.2) and the simulated light curve (see Figure 3.22).
The code (completely developed in MATLAB) is made of two parts:

- A script (`main.m`): it reads from the tables coming from SAP the evolution of RA and DEC of the streak (**Ra** and **Dec**), the associated time vector ($\mathbf{t}$), the UTC, latitude (φ) and longitude (ϑ) of the observer, the pixel mean value of the image (n) and the saturation threshold (sat). It builds the array of the real light curve read from the *.txt* coming from SAP. It sets the value of the inertia matrix ($\mathbf{I}$) and of the normal ($\mathbf{N}$). Moreover, it defines the optimization problem variables and settings and runs the solver `ga.m` (see Sec. 1.2).
- A function (`obj_fun.m`): it takes as input the initial attitude conditions to run `dynamics.m` (see Sec. 3.1) and the multiplicative factor k, in order to build the simulated light curve according to Eq. 2.18. It outputs the difference ($cost$) between

the synthetic and the real light curve. It is used as the fitness function to be minimized by `ga.m`.

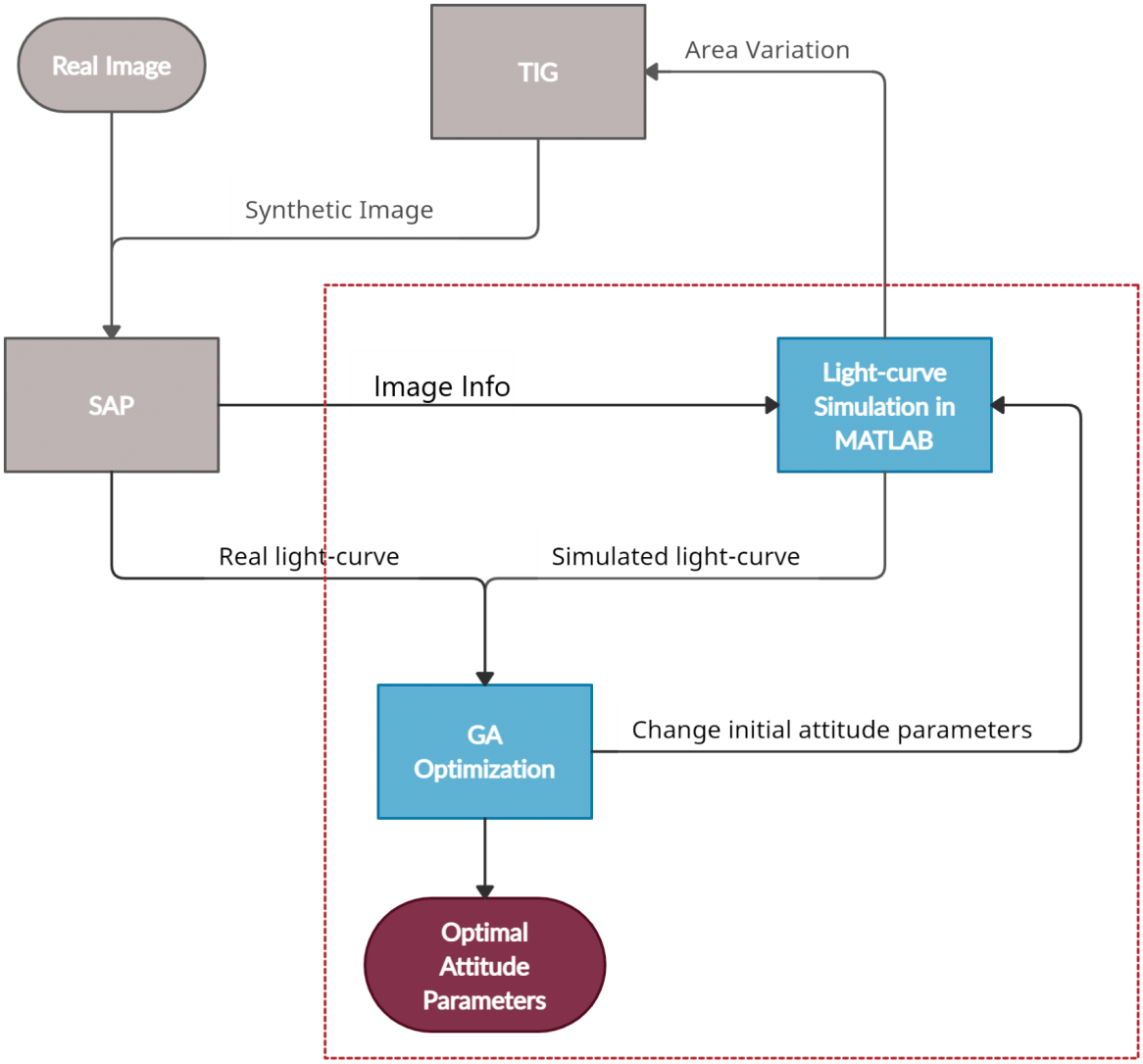


Figure 3.22: Thesis Workflow: Optimization

The cost function, in its simplest form, could be written as the difference in absolute value between the two light curves ($\mathbf{diff_{abs}}$), although implying two main issues:

1. `ga.m` can minimize a function that outputs a scalar value
2. The values in $\mathbf{diff_{abs}}$ are difficult to manage by the optimization solver due to improper scaling of light curves

Consequently, the variation of light intensity is rescaled to the [0,1] range using the saturation threshold (sat) computed in SAP. In this way, the mean cost value during the

optimization is much lower, leading to a better convergence. Then, in order to obtain a scalar value, *cost* is computed as the summation of all the elements in $\mathbf{diff_{abs}}$ (see Alg. 3.12). This decision stems from a trial and error process to achieve the best cost function for the problem at hand.

Algorithm 3.12 Cost Definition

% PROTOTYPE: `[cost] = obj_fun(var_input_opt, param)`

% ——————————————————————————— %

- Run `dynamics.m` with the initial attitude conditions and obtain $\mathbf{c_{tot}}$

% Read the real light curve in terms of central luminosity from the *.txt* file and scale it:

$\mathbf{lum_{real}} = \mathbf{lum_{centre}}/sat$

- Extract the multiplicative factor k from $\mathbf{var_{input,opt}}$

% Scale the shift n:

$n_{scaled} = n/sat$

% Compute the simulated light curve according to Eq. 2.18:

$\mathbf{lum_{sim}} = \mathbf{c_{tot}} \cdot k + n_{scaled}$

% Compute the cost:

$cost = \texttt{sum}(\texttt{abs}(\mathbf{lum_{real}} - \mathbf{lum_{sim}}))$

In order to properly set the `ga.m` problem, the different inputs to the function need to be defined:

1. `fitnessfcn`: fitness function that outputs the problem objective to be minimized. As seen above, `obj_fun.m` is used.
2. `nvars`: number of design (input) variables (x). This number is defined in order to maximize the solver speed. `x` must contain the initial conditions for the dynamics and kinematics simulation of `dynamics.m` and the $\mathbf{c_{tot}}$ upscaling parameters.
3. `Aineq` and `A`: matrices for linear inequality and equality constraints on x (not needed for the problem at hand)
4. `bineq` and `b`: vectors for linear inequality and equality constraints on x (not needed for the problem at hand)
5. `lb`: lower boundaries of x (check Table 3.1)
6. `ub`: upper boundaries of x (check Table 3.1)
7. `nonlcon`: nonlinear constraint function (not needed for the problem at hand)

8. `options`: options structure with different fields. Different settings can be defined, such as: maximum number of generations ($MaxGen$), maximum number of iterations, population, selection, crossover and mutation options, function tolerance ($FunTol$).

Quantity	`lb`	`ub`
ω_{0x}	$-\omega_{lim}$	ω_{lim}
ω_{0y}	$-\omega_{lim}$	ω_{lim}
ω_{0z}	$-\omega_{lim}$	ω_{lim}
ϕ_0	$0°$	$360°$
θ_0	$0°$	$360°$
ψ_0	$0°$	$360°$
k	0	1

Table 3.1: Lower and Upper boundaries of the design variables: ω_{lim} value is determined by performing a Fourier series expansion of the real light curve up to the second order [55] and extracting the fundamental frequency of the resulting curve.

Therefore, the final array of design variables ($\mathbf{var_{input,opt}}$, see Alg. 3.12) is composed by:

- Attitude initial conditions: three initial angular velocities (ω_{0x}, ω_{0y}, and ω_{0z}) and three initial yaw-pitch-roll angles (ϕ_0, θ_0, and ψ_0, cf. Sec. 2.1), so that only six parameters are needed to simulate the dynamics and input in `dynamics.m`, as seen before (see Alg. 3.1).
- Scaling parameters: only the multiplicative factor k, since n is taken as the mean pixel value of the image coming from SAP.

In this way, only seven parameters are iteratively updated by `ga.m`, reducing the solver computational cost. The choice of n can be justified considering that in this way the streak luminosity variation will always be above the background sky pixel values ($\mathbf{c_{tot}}$ always larger than 0), i.e. the streak is actually visible.

If all the inputs are collected in a structure (`problem`), the optimal solution is given by `x=ga(problem)`. Other outputs can be requested to further characterize the solution:

- `fval`: objective function value at the solution
- `exitflag`: integer number that tells the reason why `ga.m` stopped

- `output`: information about the optimization process, returned as a structure with different fields

Along with the GA minimization process, a plot showing the mean and the best fitness value and the progress of the optimization is provided (see Figure 3.23). Moreover, the computation time is tracked through MATLAB `tic`/`toc` functions [56].

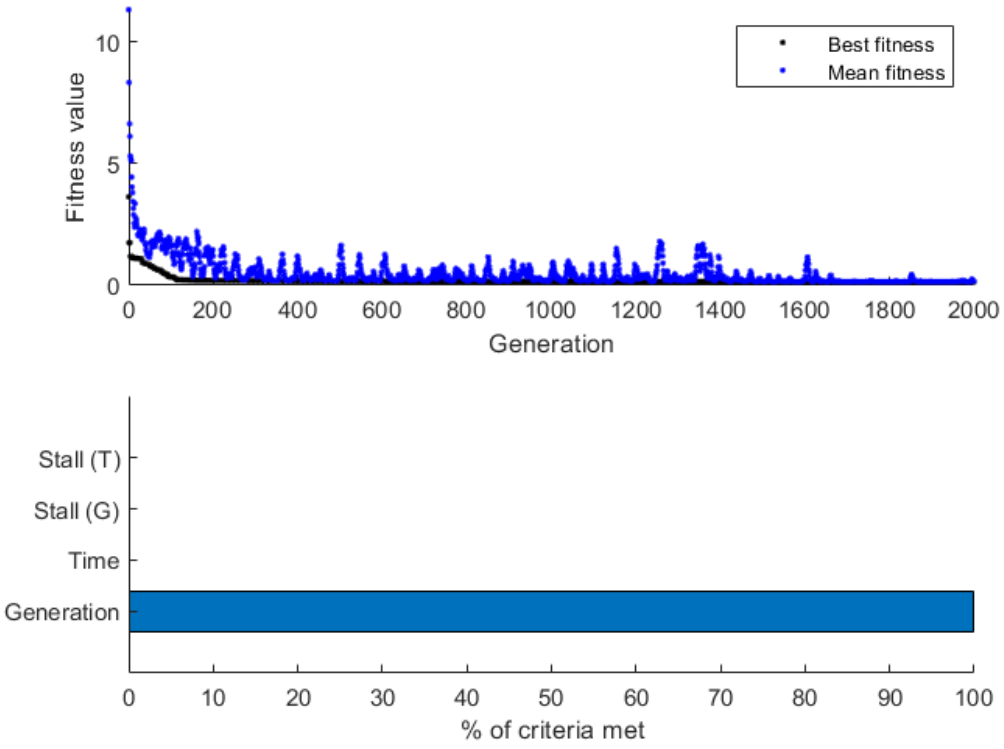


Figure 3.23: GA Progress plot. Mean and best fitness values for each generation are represented in the upper plot, while the stopping criteria (maximum stall time, $FunTol$, maximum time or $MaxGen$) of the run is in the bottom one. For the considered run, the solver stopped after reaching the maximum number of generations.

The optimization output accuracy can be estimated by visualizing the fitting between the real light curve and the simulated one (see Chap. 4).
The aforementioned procedure (summarized in Figure 3.24) can also be extended when considering the apparent magnitude light curve of saturated streaks. In this case, the scaling of real and simulated magnitude variation is modified as it can be seen in Alg. 3.13. Moreover, some additional information is needed from SAP, such as the apparent (m_{ref}) and instrumental ($I_{m_{ref}}$) magnitudes of the control star.

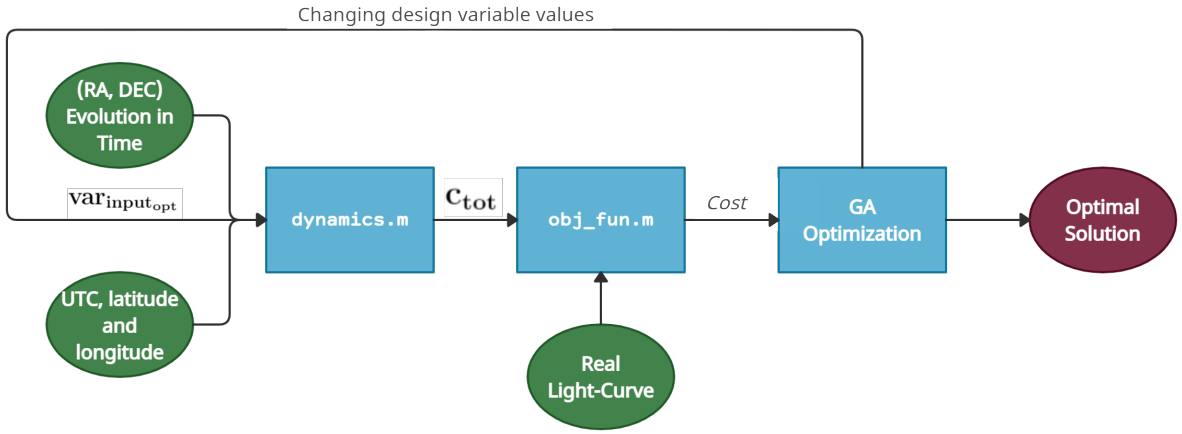


Figure 3.24: Block Scheme of the optimization.

Algorithm 3.13 Cost Definition (Apparent Magnitude)

% PROTOTYPE: `[cost] = obj_fun_mag(var_input_opt, param)`

% ——————————————————————————— %

- Run `dynamics.m` with the initial attitude conditions and obtain $\mathbf{c_{tot}}$

% Read the real light curve in terms of apparent magnitude from the *.txt* file and scale it with respect to m_{ref} and $I_{m_{ref}}$:

$\mathbf{mag_{real}} = \mathbf{m}/(m_{ref} - I_{m_{ref}})$

- Extract the multiplicative factor k from $\mathbf{var_{input,opt}}$

% No shift n considered. Compute the simulated light curve by scaling $\mathbf{c_{tot}}$, inverting its behaviour (minus sign in logarithm computing the magnitude) and subtract it to 1:

$\mathbf{mag_{sim}} = 1 - \mathbf{c_{tot}} \cdot k$

% Compute the cost:

$cost = \mathtt{sum}(\mathtt{abs}(\mathbf{mag_{real}} - \mathbf{mag_{sim}}))$

4 | Results

In this chapter, the results and performance of the algorithm are analyzed and discussed in detail. The overall procedure of angular velocity extraction is applied both on synthetic and real images.
In the realization of synthetic images, different rotational regimes of space debris are considered, in order to account for different real case scenarios [57]. In particular, three different regimes are arbitrarily defined:

- *Slow Rotation*: maximum angular velocity below 10 $^\circ/s$
- *Medium Rotation*: maximum angular velocity below 90 $^\circ/s$
- *Fast Rotation*: maximum angular velocity below 180 $^\circ/s$

Moreover, both non-saturated and saturated streaks are analyzed and the overall goodness of the different methods is compared.
The real images analyzed are taken from the Pratica di Mare ground station through a 350mm-optical telescope (PdM-MITE, Pratica di Mare - MIlitary TElescope) operated by personnel of Aero-Space System Engineering Group of Flight Test Wing [58] and designed by GMSpazio and Officina Stellare. These images were acquired in an observation campaign of September 2017.
Concerning the machine on which the tests were performed, it is a PC with an Intel i7-6500U CPU@2.5 GHz with a 16 GB RAM.

4.1. Synthetic Images

As previously seen in Sec. 3.1, the creation of synthetic images requires two main inputs to TIG: FITS file containing the background image and header metadata; simulated dynamics evolution ($\mathbf{c_{tot}}$) as the output of `dynamics.m`.
Then, when feeding the image to SAP, the user needs to set certain fixed values inside the code (cf. Sec. 3.2). Finally, the settings of the optimization solver (cf. Sec. 3.3) are defined according to the identified rotational regime.
The main distinction between the generated synthetic images will be given by the attitude

initial conditions and the multiplicative factor k.
For each rotational regime, among the various synthetic images that were generated, a representative one is selected. The results will show the effectiveness of the method (quality of the fitting and retrieval of the attitude state) and the variability between different runs on the same image. Moreover, the differences between rotational regimes will be outlined.
In general, a key role is also played by the simulation time that is set in SAP (cf. Sec. 3.2), since the solver speed is penalized when it increases. Nonetheless, longer exposure times imply a longer dynamics evolution, leading to a better attitude estimate.

Slow Rotation

In Figure 4.1, the synthetic image of a slow rotating debris generated by TIG is presented. The extension of the streak is determined by the passage extracted from SCOOP, while the colouring is given by the simulated dynamics and the scaling performed in TIG.

Figure 4.1: Slow-rotating debris synthetic image.

The initial conditions for the dynamics simulation and the multiplicative factor k are summarized in Table 4.1, along with the results obtained through the optimization for a given run.

	ω_{0x} [°/s]	ω_{0y} [°/s]	ω_{0z} [°/s]	ϕ_0 [°]	θ_0 [°]	ψ_0 [°]	k [-]	$\|\omega_0\|$ [°/s]
Simulated Dynamics	0.1	1.5	4	240	245	165	0.1526	4.27
Optimized Dynamics	0.15	1.39	4.03	106	351	341	0.1191	4.26

Table 4.1: Slow Rotation - Results

In general, the optimization for the norm of the initial angular velocity is extremely accurate, achieving a difference between the simulated and the optimized value below 0.1 °/s. In particular the relative error between the two can be defined as:

$$err = \left| \frac{\|\boldsymbol{\omega}_0^{sim}\| - \|\boldsymbol{\omega}_0^{opt}\|}{\|\boldsymbol{\omega}_0^{sim}\|} \right| \tag{4.1}$$

For the result in Table 4.1, $err = 0.2\%$.
On the other hand, the initial attitude parameters are not properly fitted; this is mainly due to the presence of multiple solutions related to the light curve fitting. In fact, in Figure 4.2, it can be seen that the solver has reached convergence.

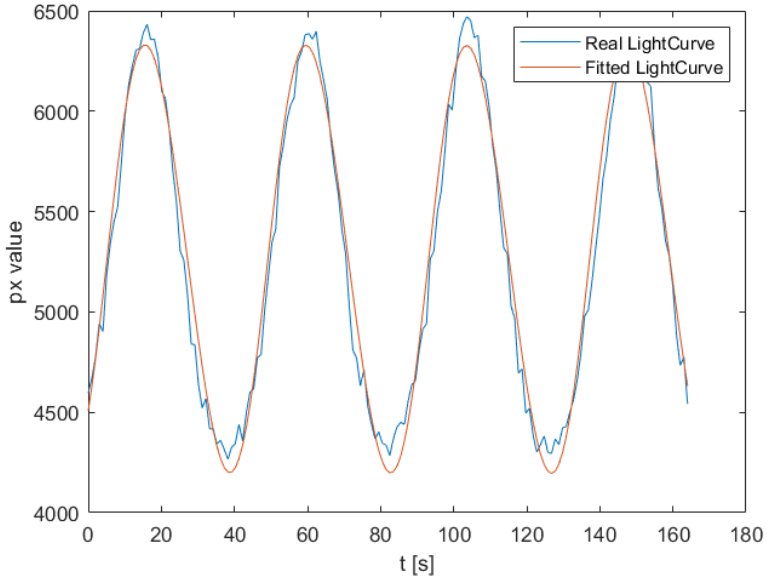


Figure 4.2: Comparison of the real (in blue) and fitted (in red) light curves. Even if the initial attitude parameters are different from the simulated ones, the curves are optimally fitted ($cost = 0.24$) and the solver correctly stops.

When running `obj_fun.m` with the simulated dynamics initial conditions and the multiplicative factor k used for the generation of the synthetic image, the cost measure may be higher than the one obtained through optimization. This is clearly due to the presence of some noise in the signal and underlines once more the possible presence of more than one optimal solution for the problem.

In order to obtain a statistical relevance of the results, the optimization with `ga.m` is carried out multiple times for the selected image. For this specific image, the following parameters are set for each run: $MaxGen = 1000$, $FunTol =$ 1e-6, and $\omega_{lim} = 8°/s$. The value of ω_{lim} is identified by taking a Fourier series expansion of the input real light curve (cf. Sec 3.3) and evaluating its period.
The results can be characterized in terms of err value for each run (see Figure 4.3). The spreading of the results is quite contained and err is always kept below 10%. For each run, the mean running time is approximately 12 minutes, that is quite contained considering that the dynamics is simulated for more than 150 seconds.

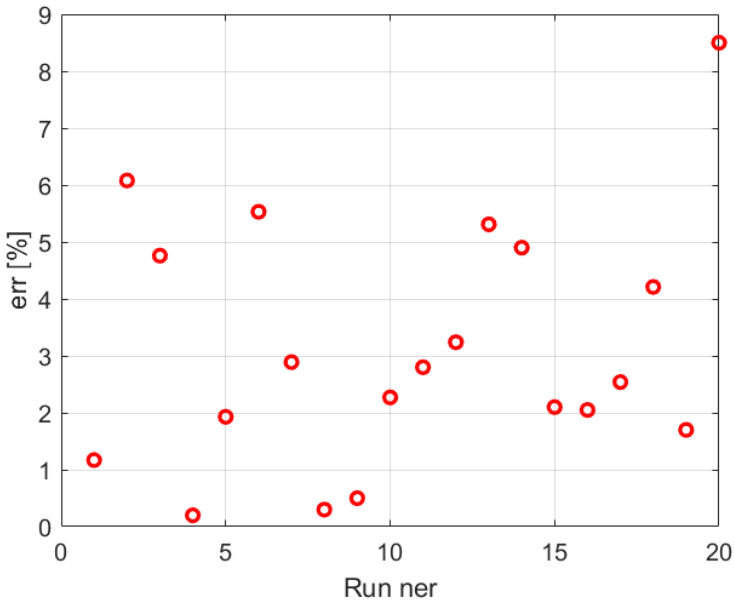


Figure 4.3: Error in multiple runs (20) of the optimization solver on the given image.

Nevertheless, the behaviour of the results when considering this kind of parameter is inevitably influenced by the absolute value of the considered angular velocity. Therefore, a classical statistical approach to the results is employed by computing the cumulative distribution function (CDF) of the relative error, in order to evaluate its changes between the different rotational regime cases.

The CDF of err can be defined as:

$$F_{err}(x) = P(err \leq x) \tag{4.2}$$

Each point of the CDF represents the probability that err will take a value less than or equal to x.

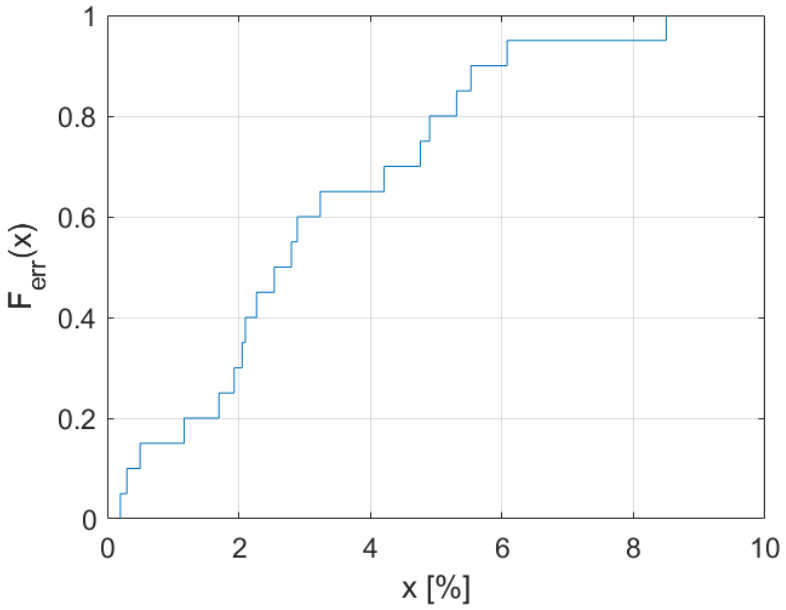


Figure 4.4: CDF of the relative error for slow rotating debris.

The CDF can be employed to better visualize the statistical distribution of err for the selected image. In fact, its steepness is an estimate of the accuracy in the angular velocity determination. In Figure 4.4, it can be seen that, in most of the cases (80%), err is below 5%.

Medium Rotation

The same procedure can be followed in the analysis of medium rotation synthetic images (see Figure 4.5). In this case, the angular velocity regime is incremented and the corresponding results[1] with respect to the simulation parameters are presented in Table 4.2.

[1]In the tables, always the best result among the different GA runs is shown.

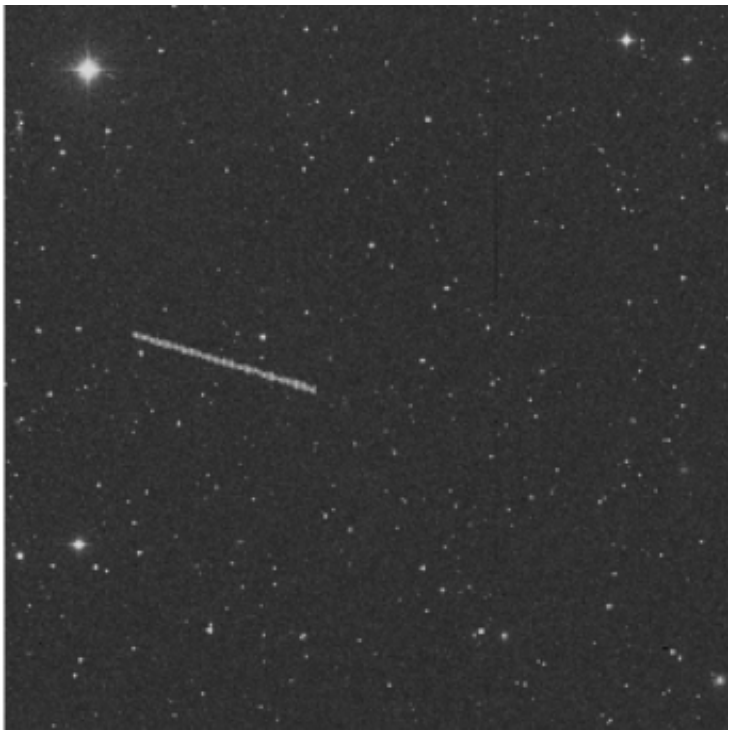


Figure 4.5: Medium-rotating debris synthetic image.

	ω_{0x} [°/s]	ω_{0y} [°/s]	ω_{0z} [°/s]	ϕ_0 [°]	θ_0 [°]	ψ_0 [°]	k [-]	$\|\omega_0\|$ [°/s]
Simulated Dynamics	25	11	50	0	0	0	0.2289	56.97
Optimized Dynamics	9.45	25.61	50.14	124	43	8	0.2263	57.08

Table 4.2: Medium Rotation - Results

These results highlight that the z-component of the angular velocity is the best fitted one, while the x and y ones are inverted. This is a common behaviour shown by the solver, occurring independently from the target rotational regime, and is closely related to the the symmetry of the flat square plate that approximates the real debris. Clearly, this is not considered an issue, since the x and y axis are completely interchangeable in this framework.

Once again, the norm of the angular velocity is properly fitted (the relative error is rather limited: $err = 0.19\%$).

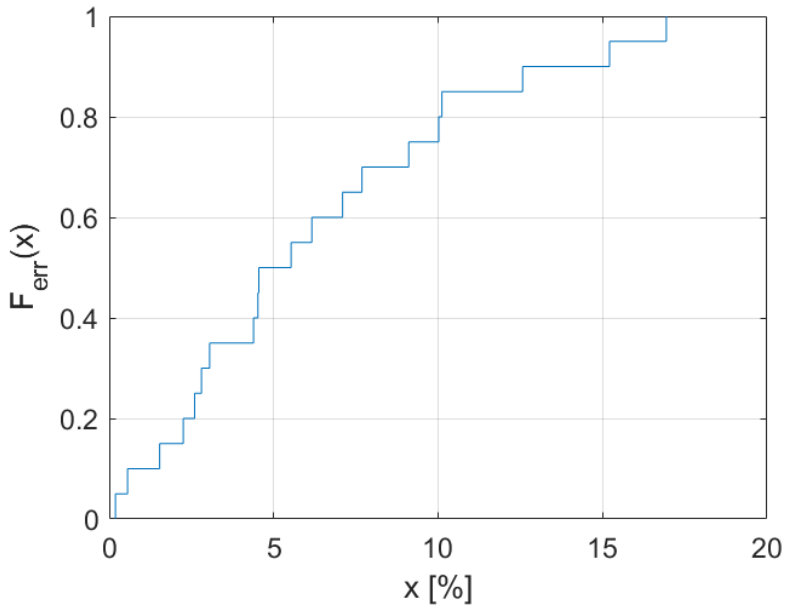


Figure 4.6: CDF of the relative error for medium rotating debris.

As it can be seen in Figure 4.6, the CDF is less steep with respect to the slow rotating case (cf. Figure 4.4). Moreover, the values of *err* are increased, meaning that in some of the results the angular velocity is off by more than 1 °/s. Nevertheless, the solver is able to find accurate solutions for an acceptable number of times (approximately 70% of probability to get a result 5°/s off). It is worth noting that the CDF is computed for 20 values of *err* (20 runs each). This number of samples may not be completely indicative of the overall behaviour, but is considered a good compromise between the accuracy of the statistics and the time required by each GA run (for the medium rotation synthetic image considered, approximately 23 minutes per run were required for the 8-second simulated dynamics).
The parameters used for `ga.m` for the image in question are:

Parameter	Value
$MaxGen$	2000
$FunTol$	1e-9
ω_{lim}	67 °/s

Fast Rotation

For fast rotating debris (see Figure 4.7), the optimization is carried out in the same way as the other rotational regime cases.

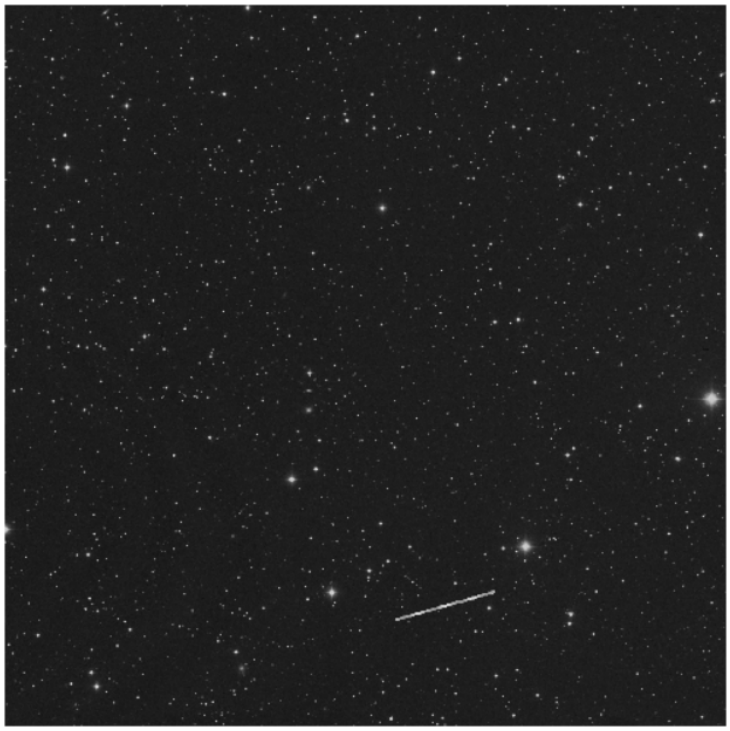


Figure 4.7: Fast-rotating debris synthetic image.

In this kind of synthetic images, the values of the angular velocity are associated to fast spinning objects in LEO (cf. Sec. 4.2): the observation times are rather limited and the evolution of the dynamics is quite contained.
The parameters set for `ga.m` are the same as the medium rotating case, with an increased $\omega_{lim} = 129°/s$. The mean run time is approximately 18 minutes, which is quite high considering that the dynamics evolution is set at 1 second.

For the selected image, the initial simulation conditions and the optimized results are summarized in Table 4.3.

	$\boldsymbol{\omega_{0x}}$ [°/s]	$\boldsymbol{\omega_{0y}}$ [°/s]	$\boldsymbol{\omega_{0z}}$ [°/s]	$\boldsymbol{\phi_0}$ [°]	$\boldsymbol{\theta_0}$ [°]	$\boldsymbol{\psi_0}$ [°]	**k** [-]	$\|\boldsymbol{\omega_0}\|$ [°/s]
Simulated Dynamics	81	50	120	200	15	45	0.1831	153.17
Optimized Dynamics	76.47	57.44	120.13	191	20	42	0.1827	153.55

Table 4.3: Fast Rotation - Results

Once again the results show the trend to fit more properly the z-axis (major inertia one,

see Sec. 2.1).

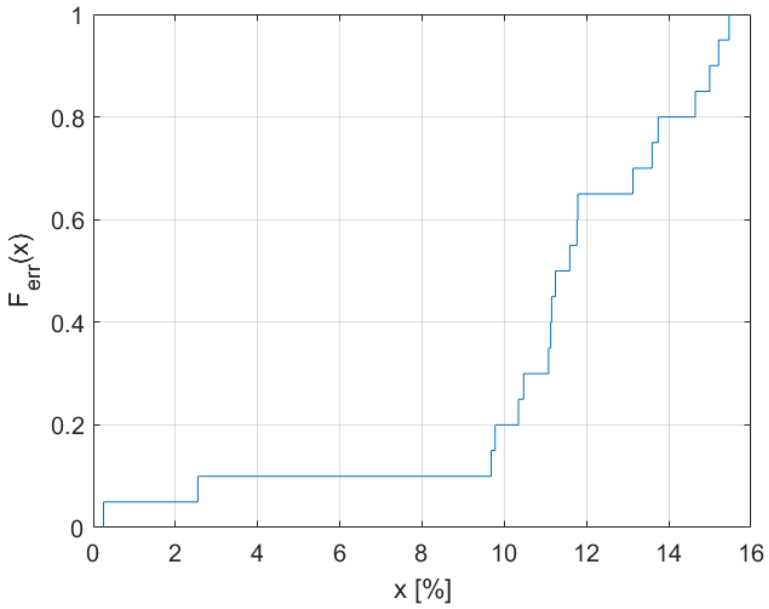


Figure 4.8: CDF of the relative error for fast rotating debris.

The relative error for the norm of the angular velocity in Table 4.3 is again quite low ($err = 0.25\%$). Nevertheless, the statistical behaviour of the CDF shows an increased worsening of the solver performance with the increase of the angular velocity values (see Figure 4.8).

Saturated Streaks

As seen in Chap. 3, the generation and analysis of synthetic images containing saturated streaks (Figure 4.9) is also carried out in this work. The reasoning behind the code is the same as the one used for non-saturated streaks. The main differences are the incremented computational burden withstood by SAP (cf. Sec. 3.2) and the different nature of the input light curve in terms of apparent magnitude evolution with time (cf. Sec. 3.3).
A slow rotational regime is considered in the selected image, and the initial conditions and optimized results relative to it are presented in Table 4.4.
In general, it is evident that in case of saturated streaks the performance (with the same settings as the slow rotation case) slightly decreases. Nevertheless, the norm of the angular velocity extracted is less than 1 °/s off the simulated value, which can be considered an acceptable result.

Figure 4.9: Saturated synthetic image.

	ω_{0x} [°/s]	ω_{0y} [°/s]	ω_{0z} [°/s]	ϕ_0 [°]	θ_0 [°]	ψ_0 [°]	$\|\omega_0\|$ [°/s]
Simulated Dynamics	5	2	8	0	80	200	9.64
Optimized Dynamics	2.75	-2.32	-8.25	281	272	89	9.03

Table 4.4: Saturated - Results

Another visible feature, that occurs for all the target rotational regimes, is the opposite sign between the optimized and the simulated angular velocities. Evidently, this is due to the possibility for the solver to choose indifferently a positive or a negative z-axis for the flat plate (cf. Sec. 2.1). This additional degree of freedom, even if it implies this sign inversion, appeared to be an important feature for the solver ability to reach convergence, since it allows GA to search for more diverse solutions and not to be stuck in sub-optimal ones.

The CDF for *err* in different runs is less steep with respect to the results of a non-saturated streak (Figure 4.10). This decline in the performance is clearly due to the increased loss of information caused by saturation (see Sec. 2.2). This causes the solver to converge to

worse solutions with a run time similar to the one of non-saturated slow rotation synthetic images (approximately 9 minutes).

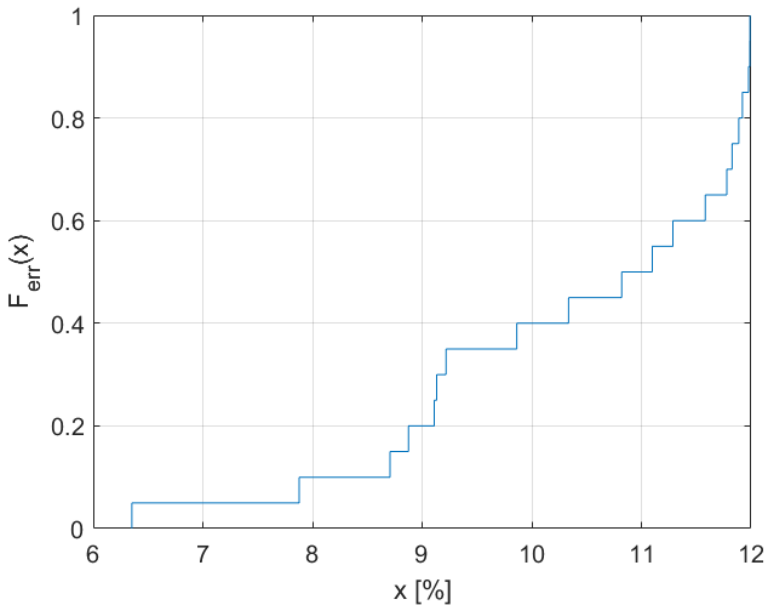
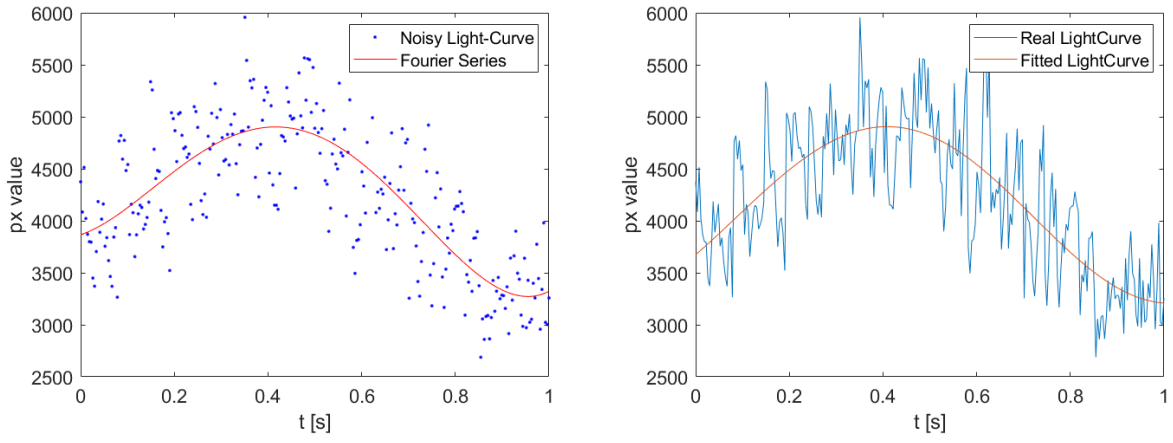


Figure 4.10: CDF of the relative error for saturated streaks.

4.2. Real Images

Once the code is tested against synthetic images, the analysis switches to real ones. In this case, the results of the optimization can be directly shown and the overall process is lighter in terms of tasks to be performed.
The image (see Figure 4.11) acquired by PdM-MITE contains a rather faint streak left by a debris orbiting in LEO (exposure time of 1 second). In particular, through a simulation on `sorvegliatispaziali.inaf.it` satellites map [59], the object passing in the image FOV at the FITS header UTC is identified as the COSMO SkyMed 2, an ASI (Agenzia Spaziale Italiana) artifical LEO satellite launched in 2007.
Since the image does not have any `WCS` info in the header, this is obtained with `Astrometry.net` web service (cf. Sec. 2.2). Then, it is fed to SAP that extracts the real light curve.
As it can be seen from Figure 4.12, the curve is quite noisy and therefore a filtering of the signal is performed. In fact, it was noted that by using a filtered version of the noisy real light curve, the performance of the optimization solver improved. Moreover, a re-sampling of the signal is carried out to decrease the number of points in the light curve, increasing the solver speed; the selection of the number of points is done taking into consideration possible aliasing problems.

Figure 4.11: Real image from the PdM-MITE 2017 observation campaign. The rendering of the image is in logarithmic scale in order to better visualize the faint streak.



(a) (b)

Figure 4.12: Light curve filtering and results of the fitting. **(a)** Real noisy light curve and filtered curve through Fourier series expansion. **(b)** Results of the optimization: real light curve in blue, optimized one in red.

The parameters used for `ga.m` are: $MaxGen = 1000$, $FunTol =$ 1e-6, and $\omega_{lim} = 360°/s$. The value of ω_{lim} is set to grant a higher degree of freedom to the GA optimization during

the design variable update. This was done after observing that a more stringent limitation on the angular velocities led to an early poor convergence of the solver.
In Table 4.5, the results of the optimization are shown.

ω_{0x} [°/s]	ω_{0y} [°/s]	ω_{0z} [°/s]	ϕ_0 [°]	θ_0 [°]	ψ_0 [°]	$\|\omega_0\|$ [°/s]
-11.5	2.8	149	233	216	30	149.47

Table 4.5: Real Image - Results

The results are rather consistent with a fast spinning object in LEO and, as seen before for fast rotating debris, the solver performance decreases with the increased value of angular velocity both in terms of accuracy of the prediction and of optimization speed (average run time of 11 minutes for just one second of simulation of the dynamics). Nevertheless, for the identified object (COSMO SkyMed 2), the value of ω_{0z} seems quite high. The short-evolution dynamics does not allow a more precise estimate.
The fitting (see again Figure 4.12) can be considered accurate even if some of the fast variations in the real light curve cannot be followed by the reflection model and simplified geometry (cf. Sec. 2.1) used in the optimization problem.

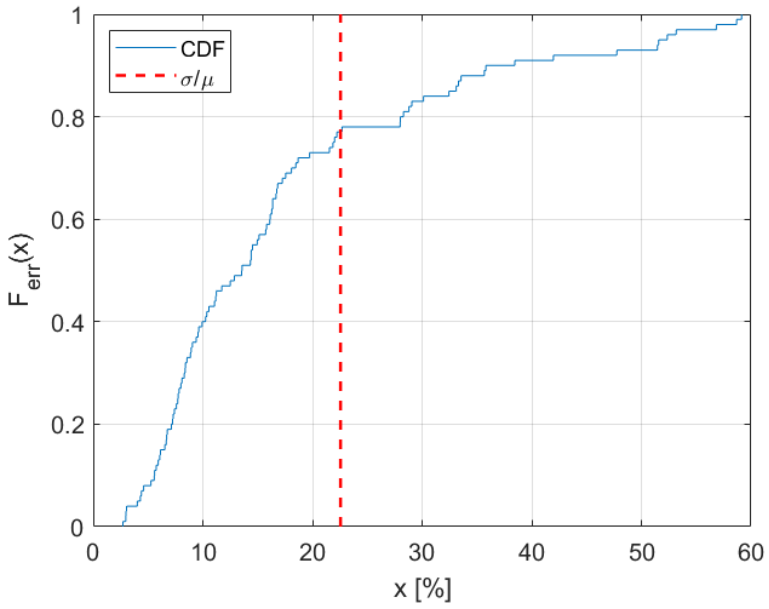


Figure 4.13: CDF of the relative error for real streaks. The CDF is quite steep, even if err is not really contained. Nevertheless almost 80% of the results are within 1σ of μ.

The optimization is performed for 100 times on the same image to evaluate the variability

of the results. To achieve so, a new relative error is defined as:

$$err = \left| \frac{\|\boldsymbol{\omega_0}\| - \mu}{\mu} \right| \tag{4.3}$$

Where $\mu = 140.01°/s$ is the statistical mean of the angular velocities norm. Then, it is possible to evaluate the CDF of err and consider the probability of having a $\|\boldsymbol{\omega_0}\|$ that differs more than 1σ (standard deviation, $\sigma = 31.58°/s$) from μ.
The results for 100 runs, as it can be seen in Figure 4.13, are quite variable. This is mainly due to the short evolution of the extracted light curve that may have different optimal fittings and to the high rotational speeds, leading to worse results (cf. Sec. 4.1).

5 | Conclusions

The attitude determination of a tumbling space debris is a quite difficult challenge. Among the many available methods, only few are able to extract a complete state and rather focus on the determination of the period of rotation. In this thesis, a light curve inversion for attitude reconstruction is carried out on both real and synthetic images.
The generation of synthetic images, performed using TIG and the dynamics simulator, outputs FITS files that resemble the characteristics of a real streak left by a tumbling debris. Moreover, SAP successfully extracts, analyzes, and post-processes the streak present in any input FITS image. The optimization part of the code is able to identify different rotational regimes of the tumbling object and conduct the analysis accordingly. The performance of the optimization generally decreases with the increase of the rotation rate (fast rotating debris), and if the streak is saturated. For noisy data, filtering of the signal proved to be necessary in order to improve the performance of GA. The randomness of the results of the latter is taken into consideration by performing multiple runs on the same light curve, in order to consider a statistics of the different results of the fitting.
The analysis proved to be quite successful in the determination of the norm of the angular velocity vector and, in some cases, of its components. On the other hand, attitude orientation parameters are usually not identified. This is mainly due to the:

- Approximation of the generic space debris to a flat square plate
- Multiplicity of optimal solutions for the problem at hand

Nevertheless, since the code in this thesis is intended to be used for a preliminary analysis of an unknown rotating space debris, this is not considered an issue.
The use of synthetic images was fundamental for the validation of the code and for the set up of its main properties. The results for real images containing tumbling space debris are quite variable and their accuracy should be checked. Hence, it could be useful to test the code on a wider range of real images (e.g. with real saturated streaks inside) with the chance to validate the results against known objects. Moreover, more complex models could be used for the reflection and shaping of the debris when a refined analysis is required.

Bibliography

[1] European Space Agency. Space Debris by the Numbers. https://www.esa.int/Safety_Security/Space_Debris/Space_debris_by_the_numbers. [Accessed: 22-10-2021].

[2] ESA Space Debris Office. ESA's Annual Space Environment Report, 27 May 2021. https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf.

[3] European Space Agency. About Space Debris. https://www.esa.int/Safety_Security/Space_Debris/About_space_debris. [Accessed: 22-10-2021].

[4] D. J. Kessler and B. G. Cour-Palais. Collision frequency of artificial satellites: The creation of a debris belt. Journal of Geophysical Research: Space Physics, 83(A6):2637–2646, 1978.

[5] R. Cipollone and A. De Vittori. Machine Learning Techniques for Optical and Multi-beam Radar Track Reconstruction of LEO Objects. Master's thesis, Politecinco di Milano, School of Industrial and Information Engineering, Department of Aerospace Science and Technology, Milan, Italy, 2020.

[6] A.Morselli. High order methods for Space Situational Awareness. PhD thesis, Politecinco di Milano, School of Industrial and Information Engineering, Department of Aerospace Science and Technology, Milan, Italy, 2014.

[7] J. Šilha. Reviews in Frontiers of Modern Astrophysics: From Space Debris to Cosmology, pages 1–21. Springer International Publishing, Cham, 2020.

[8] European Space Agency. Active Debris Removal. https://www.esa.int/Safety_Security/Space_Debris/Active_debris_removal. [Accessed: 22-10-2021].

[9] L.D. Blacketer, H.G. Lewis, and H. Urrutxua. A Technique for Rotation Vector Position Determination for Tumbling Rocket Bodies. In First Int'l. Orbital Debris Conf., Houston (TX), USA, 9-12 December 2019.

[10] P. Denchev, P. Magnusson, and Z. Donchev. Lightcurves of nine asteroids, with pole and sense of rotation of 42 Isis. Planetary and Space Science, 46(6):673–682, 1998.

[11] M. Graham, A. Drake, G. Djorgovski, A. Mahabal, C. Donalek, V. Duan, and A. Maher. A Comparison of Period Finding Algorithms. Monthly Notices of the Royal Astronomical Society, 434:3423, July 2013.

[12] M. Kaasalainen, J. Torppa, and K. Muinonen. Optimization Methods for Asteroid Lightcurve Inversion: II The Complete Inverse Problem. Icarus, 153(1):37–51, 2001.

[13] E. F. Linder, J. Šilha, T. Schildknecht, and M. Hager. Extraction of Spin Periods of Space Debris from Optical Light Curves. In 66th International Astronautical Congress, Jerusalem, IL, 12-16 October 2015.

[14] B. Wallace, P. Somers, and R. Scott. Determination of Spin Axis Orientation of Geosynchronous Objects Using Space-Based Sensors: An Initial Feasibility Investigation. In 11th Advanced Maui Optical and Space Surveillance Technologies Conference, Maui (HI), USA, 14-17 September 2010.

[15] F. Piergentili, G. Zarcone, L. Parisi, L. Mariani, S. H. Hossein, and F. Santoni. LEO Object's Light-Curve Acquisition System and Their Inversion for Attitude Reconstruction. Aerospace, 8(1), 2021.

[16] R. Kanzler, T. Schildknecht, T. Lips, B. Fritsche, J. Šilha, and H. Krag. Space Debris Attitude Simulation - IOTA (In-Orbit Tumbling Analysis). In 16th Advanced Maui Optical and Space Surveillance Technologies Conference, Maui (HI), USA, 15-18 September 2015.

[17] MathWorks©. Nonlinear Optimization. `https://it.mathworks.com/help/optim/nonlinear-programming.html`. [Accessed: 26-10-2021].

[18] H.D. Curtis. Orbital Mechanics for Engineering Students, chapter 5, pages 253–261. Elsevier Ltd, 4th edition, 2020.

[19] H.D. Curtis. Orbital Mechanics for Engineering Students, chapter 11, pages 557–600. Elsevier Ltd, 4th edition, 2020.

[20] P. A. Tipler. Physics, chapter 12. Worth Publishers, 1st edition, 1976.

[21] J. D. Biggs. Attitude Dynamics and Control Lecture Notes. Politecnico di Milano, School of Industrial and Information Engineering, Department of Aerospace Science and Technology, 2020.

[22] Y. Matsushita, R. Arakawa, Y. Yoshimura, and T. Hanada. Light Curve Analysis

and Attitude Estimation of Space Objects Focusing on Glint. In First Int'l. Orbital Debris Conf., Houston (TX), USA, 9-12 December 2019.

[23] A. Michael and S. Peter. An Anisotropic Phong Light Reflection Model. Journal of Graphics Tools, 5, January 2001.

[24] M. Massari. Optical Instruments PPTX Presentation. Politecnico di Milano, School of Industrial and Information Engineering, Department of Aerospace Science and Technology, April 2020.

[25] H. Karttunen, P. Kröger, H. Oja, M. Poutanen, and K. J. Donner. Fundamental Astronomy, chapter 3, pages 47–54, 69–71. Springer International Publishing, 5th edition, 2007.

[26] NASA. Atmospheric electromagnetic transmittance or opacity. `https://commons.wikimedia.org/wiki/File:Atmospheric_electromagnetic_transmittance_or_opacity.jpg`. [Accessed: 13-11-2021].

[27] G. Claus and S. Boris. Particle Detectors. Cambridge Monographs on Particle Physics, Nuclear Physics and Cosmology. Cambridge University Press, 2nd edition, 2008.

[28] P. Seitzer. Satellite Visibility and Brightness. SIA-AAS Webinar, May 2020. `https://aas.org/sites/default/files/2020-06/Seitzer_SIAAAS_2020May26.pdf`.

[29] W. D. Pence, L. Chiappetti, C. G. Page, R. A. Shaw, and E. Stobie. Definition of the Flexible Image Transport System (FITS), version 3.0. A&A, 524:A42, 2010.

[30] HEASARC. FITS Image Viewer Page. `https://fits.gsfc.nasa.gov/fits_viewer.html`. [Accessed: 29-10-2021].

[31] Astrometry.net. Web Sevice for Image Calibration. `http://nova.astrometry.net/upload`. [Accessed: 04-11-2021].

[32] ESA Science and Technology. Lightcurve of Asteroid 2867 Steins. `https://sci.esa.int/web/rosetta/-/40860-lightcurve-of-asteroid-2867-steins`. [Accessed: 25-10-2021].

[33] AAVSO. Variable Stars and Phase Diagrams. In Variable Star Astronomy. `https://www.aavso.org/sites/default/files/education/vsa/Chapter12.pdf`. [Accessed: 25-10-2021].

[34] W. Romanishin. An Introduction to Astronomical Photometry Using CCDs. `https://www1.phys.vt.edu/~jhs/phys3154/CCDPhotometryBook.pdf`, March 2002.

[35] J. A. Blake, et al. DebrisWatch I: A survey of faint geosynchronous debris. Advances in Space Research, 67(1):360–370, 2021.

[36] J. Allworth, L. Windrim, J. Wardman, D. Kucharski, J. Bennett, and M. Bryson. Development of a High Fidelity Simulator for Generalised Photometric Based Space Object Classification using Machine Learning. In 70th International Astronautical Congress (IAC), Washington D.C., USA, 21-25 October 2019.

[37] M. Masias, J. Freixenet, X. Lladó, and M. Peracaula. A review of source detection approaches in astronomical images. Monthly Notices of the Royal Astronomical Society, 422(2):1674–1689, April 2012.

[38] S. Tonkin. Astronomy Star Catalogues: Which to Use and When. `https://www.skyatnightmagazine.com/advice/skills/astronomy-star-catalogues-which-to-use-and-when/`. [Accessed: 15-10-2021].

[39] Vijay C., Sourabh K., and Sumit C. A Review on Genetic Algorithm: Past, Present, and Future. Multimedia Tools and Applications, 80, February 2021.

[40] Zbigniew M. and Marc S. Evolutionary Algorithms for Constrained Parameter Optimization Problems. Evolutionary Computation, 4(1):1–32, March 1996.

[41] V. Mallawaarachchi. Introduction to Genetic Algorithms. `https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3`. [Accessed: 30-10-2021].

[42] MathWorks©. ga. `https://it.mathworks.com/help/gads/ga.html`. [Accessed: 21-10-2021].

[43] Orbital and Celestial Mechanics Website. Orbital Mechanics with MATLAB. `http://celestialandorbitalmechanicswebsite.yolasite.com/`. [Accessed: 01-11-2021].

[44] G. Purpura. SCOOP PPTX Presentation. Politecnico of Milan, Faculty of Space Engineering, February 2020.

[45] D. A. Vallado, P. Crawford, R. Hujsak, and T. S. Kelso. Revisiting Spacetrack Report N.3. In AIAA/AAS Astrodynamics Specialist Conference, Keystone, CO, 21-24 August 2006.

[46] European Southern Observatory. ESO Online Digitized Sky Survey. `https://archive.eso.org/dss/dss`. [Accessed: 01-11-2021].

[47] K. Dae-Won. ASTRiDE: Automated Streak Detection for Astronomical Images. https://ui.adsabs.harvard.edu/abs/2016ascl.soft05009K, May 2016.

[48] L. Bradley, et al. . astropy/photutils: 1.0.0, September 2020.

[49] Astropy Collaboration, et al. The Astropy Project: Building an Open-science Project and Status of the v2.0 Core Package. Astronomical Journal, 156(3):123, September 2018.

[50] P. B. Stetson. DAOPHOT - A computer program for crowded-field stellar photometry. Astronomical Journal, 99:191, March 1987.

[51] S. Littlefair. Observational Astronomy Lectures: Photometry I and Photometry II. University of Sheffield, Department of Physics and Astronomy, 2015. http://slittlefair.staff.shef.ac.uk/teaching/phy241/.

[52] J. Caron. aspylib: 3.0.0. http://www.aspylib.com/, 2019.

[53] First Light Optics Ltd. Astronomy Tools: CCD Calculators. https://astronomy.tools/calculators/ccd. [Accessed: 03-11-2021].

[54] G. Adam, et al. astroquery: An Astronomical Web-querying Package in Python. Astronomical Journal, 157(3):98, March 2019.

[55] MathWorks©. Fourier Series. https://it.mathworks.com/help/curvefit/fourier.html. [Accessed: 29-10-2021].

[56] MathWorks©. tic. https://it.mathworks.com/help/matlab/ref/tic.html. [Accessed: 29-10-2021].

[57] J. Šilha, J-N. Pittet, M. Hamara, and T. Schildknecht. Apparent rotation properties of space debris extracted from photometric measurements. Advances in Space Research, 61(3):844–861, 2018.

[58] G. M. D. Genio, J. Paoli, E. D. Grande, F. Dolce. Italian Air Force Radar and Optical Sensor Experiments for the Detection of Space Objects in LEO Orbit.

[59] INAF. Sorvegliati Spaziali: Mappa dei satelliti. https://sorvegliatispaziali.inaf.it/mappa-dei-satelliti/. [Accessed: 13-11-2021].

[60] K. Burnett. Converting RA and DEC to ALT and AZ. http://www.stargazing.net/kepler/altaz.html. [Accessed: 08-11-2021].

A | Appendix A

As seen in Sec. 2.1, it is useful to transform the celestial coordinates of a space object (RA and DEC) into its local ones (Az and El) for a given observer location and time. For this purpose, a function is developed in MATLAB (`RADEC2AzEl.m`), and its working principle is based on the following procedure [60].
First of all, the inputs needed are:

- **Ra** and **Dec** evolution arrays
- UTC (in `YYYY/MM/DD hh:mm:ss`), latitude (φ) and longitude (λ) of the observer (both in decimal degrees)

The first step is to link the streak observation to a reference date. For most modern astronomical computations, J2000 is used. J2000 corresponds to 12:00 hrs UT of 1st January 2000, and the observation UTC can be related to it in term of days passed since (d). Once this operation is done, it is possible to determine the local sidereal time (LST) of the observer in decimal degrees with the following formula:

$$LST = 100.46 + 0.985647 * d + \lambda + 15 * UT \tag{A.1}$$

Where UT is the Universal Time in decimal hours; the constants inside are defined as:

- 100.46°: value needed to make the expression yield the correct value for GST at 12h UT on 1 January 2000
- 0.985647 °/day: number of degrees the Earth rotates in one mean solar day
- 15 °/h: number of degrees the Earth rotates with respect to the mean fictitious Sun every hour

Then, the value in degrees of LST is brought in the range 0° to 360° by adding or subtracting multiples of 360°.
From LST, the local hour angle (LHA) (angle between observer meridian and meridian

of the mean Sun) is computed for **Ra** at a given instant of time ($\hat{t}$):

$$LHA = LST - \mathbf{Ra}(\hat{t}) \tag{A.2}$$

Referring to Figure 2.2, the spherical triangle in Figure A.1 is identified.

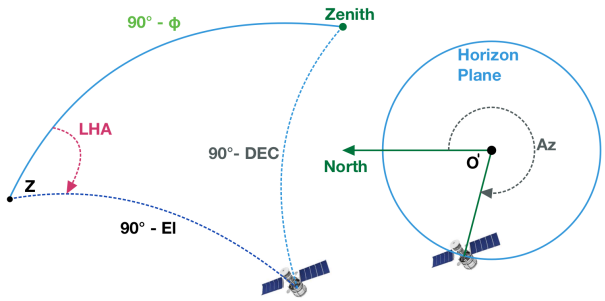


Figure A.1: Spherical Triangle

Now, through spherical trigonometry it is possible to solve for Az and El:

$$\begin{cases} El = \arcsin\left(\sin\left(DEC\right)\sin\left(\varphi\right) + \cos\left(DEC\right)\cos\left(\varphi\right)\cos\left(LHA\right)\right) \\ Az = \begin{cases} \arccos\left(\frac{\sin\left(DEC\right) - \sin\left(El\right)\sin\left(\varphi\right)}{\cos\left(El\right)\cos\left(\varphi\right)}\right) & \text{if: } \sin\left(LHA\right) < 0 \\ 360^\circ - \arccos\left(\frac{\sin\left(DEC\right) - \sin\left(El\right)\sin\left(\varphi\right)}{\cos\left(El\right)\cos\left(\varphi\right)}\right) & \text{if: } \sin\left(LHA\right) > 0 \end{cases} \end{cases} \tag{A.3}$$

List of Figures

List of Tables

List of Symbols

Acronyms

- A/D: Analog to Digital
- ADR: Active Debris Removal
- ASI: Agenzia Spaziale Italiana
- Az: Azimuth
- B: Body
- CCD: Charged Coupled Device
- CMOS: Complementary Metal-Oxide Semiconductor
- CP: Crossover Point
- DCM: Direction Cosine Matrix
- DEC: Declination
- DFT: Discrete Fourier Transform
- DISCOS: Database and Information Characterizing Objects in Space
- DSS: Digitized Sky Survey
- ECEF: Earth-Centered Earth-Fixed
- ECI: Earth-Centered Inertial
- EE: Euler's rotation Equations
- ESO: European Southern Observatory
- El: Elevation
- FFT: Fast Fourier Transform
- FITS: Flexible Image Transport System
- FOV: Field Of View
- FWHM: Full Width at Half Maximum
- GA: Genetic Algorithm
- GEO: Geostationary Earth Orbit
- GST: Greenwich Sidereal Time
- HDU: Header and Data Unit
- ISS: International Space Station
- J2000: Julian day in reference of 01-01-2000 at 12:00:00 UTC
- LEO: Low Earth Orbit
- LHA: Local Hour Angle
- LST: Local Sidereal Time
- PA: Principal Axes
- PAB: Phase Angle Bisector
- PC: Principal Components

- PD: Payload Debris
- PF: Payload Fragmentation debris
- PL: Payload
- PM: Payload Mission related object
- PSF: Point Spread Function
- PdM-MITE: Pratica di Mare - MIlitary TElescope
- RA: Right-Ascension
- RB: Rocket Body
- RCS: Radar Cross Section
- RD: Rocket Debris
- RF: Reference Frame
- RFr: Rocket Fragmentation debris
- RM: Rocket Mission related object
- RR: Real Run
- SAP: Streak Analysis Pipeline
- SCOOP: SpaceCraft and Objects Observation Planner
- SST: Space Surveillance and Tracking
- TH: Topocentric Horizon
- TIG: Tracklet Image Generator
- TLE: Two-Line Element
- TR: Trial Run
- UI: UnIdentified
- UTC: Coordinated Universal Time
- WCS: World Coordinate System

Reference Symbols

- Alg.: Algorithm
- Chap.: Chapter
- Eq.: Equation
- Sec.: Section

Mathematical Notation

Quantity	Notation
Scalar	a
Vector	$\mathbf{a}$
Matrix	$\mathbf{A}$

Built-In Functions

Function	Description	Prototype
`abs`	It returns the absolute value of a number	`angle2dcm(`a`)`
`angle2dcm`	It transforms attitude angles into DCM	`angle2dcm(`**angles**`)`
`DAOStarFinder`	It performs source extraction according to the DAOFIND algorithm	`DAOStarFinder(FWHM, threshold)`
`Euler`	It sets up the system of differential equations both for the dynamics (EE) and kinematics	`Euler(`**t, vect$_0$, I**`)`
`ga`	It performs the optimization of a problem with GA	`ga(problem)`
`linspace`	It creates a vector with specified bounds and length	`linspace(start, end, n_els)`
`match_to_ catalog_sky`	It evaluates the distance between catalogue and extracted sources	`coord_img.match_to_ catalog_sky(coord_cat)`
`pixel_to_world`	It transforms pixel coordinates into celestial ones (RA, DEC)	`WCS.pixel_to _world(x, y)`
`range`	It creates a vector given the bounds and the step	`range(start, end, step)`
`size`	It returns the dimension of the input array	`size(`**a**`)`
`sum`	It sums the components of a vector along a specified direction	`sum(a, dir)`
`sun`	It returns the Sun position in ECI for a given day	`sun(day)`

Acknowledgements

First of all, I wish to show my deepest gratitude to my advisor professor Mauro Massari, who followed me patiently during the development of this work. I would also like to thank the Italian Air Force for the precious images from the Pratica di Mare ground station used in the thesis. Moreover, I would like to pay my special regards to Andrea De Vittori, Riccardo Cipollone and Giovanni Purpura, who followed and helped me in the writing of the code, and for their precious advice about the drafting of the thesis and, generally, for their extreme availability.

Un ringraziamento speciale va ai miei genitori che hanno continuamente supportato i miei cinque anni di studio al Politecnico e più in generale di permanenza a Milano. Siete un esempio per me e senza il vostro continuo appoggio e sostegno non sarei mai riuscito a farcela. Mi avete reso l'uomo che sono oggi e spero di avervi reso orgogliosi.

Ringrazio anche mio fratello Filippo: la tua complicità ed amicizia mi hanno permesso di affrontare questi anni lontano da casa con maggiore leggerezza. Non sai quanto sono contento di poter condividere questo momento con te.

Ringrazio poi Giulia che è sempre stata al mio fianco senza mai farmi pesare la distanza. Il tuo amore e la tua disponibilità mi hanno permesso di affrontare questi anni con una maggiore serenità e con la consapevolezza di avere accanto una persona speciale; ti amo.

Più in generale, sento di dover ringraziare la mia intera famiglia, a partire dai miei nonni, che spero di aver reso orgogliosi con questo traguardo. In particolare, un pensiero va a mio nonno Franco, al quale dedico questo lavoro, perchè so quanto gli sarebbe piaciuto il cammino che ho intrapreso. Ai miei nonni materni, Maria e Fabio, e a mia nonna Serena, grazie per essermi sempre stati accanto con affetto incondizionato.

Un pensiero lo dedico anche a tutti gli zii e le zie, insieme ai miei cuginetti, Pietro e Olivia: avete reso ancora più piacevole ogni mio ritorno a Roma negli anni.

Una dedica va anche a tutti i miei compagni di studio negli anni qui al Politecnico, con i quali sento di aver stretto dei profondi rapporti di amicizia che, nonostante le distanze, rimangono ben saldi. Ringrazio anche i miei amici di sempre a Roma ed in particolare il mio compagno di disavventure Giovanni, con il quale nei primi anni a Milano ho condiviso l'esperienza da fuorisede.

Un ringraziamento va poi anche al Politecnico di Milano e ai miei vari professori avuti nel corso degli anni. Senza le loro sfide, il mio cammino sarebbe certamente stato meno entusiasmante.