POLITECNICO DI TORINO

Master degree course in Aerospace engineering
Master degree thesis

Euclid AOCS: Fine Guidance Sensor modelling and simulation

Supervisors
Prof. Sabrina Corpino
Eng. Andrea Bacchetta
PhysD. Andrea Bosco

Candidate
Marco Rotondo

March 2019

Acknowledgements

I would like to acknowledge all the people that supported me in the writing of this thesis. They all, in an unique way, have been fundamental to the accomplishment of the work.
First of all, I want to thank Andrea Bacchetta and Andrea Bosco that have guided me through all the six months spent in Thales Alenia Space as student. Then, it is worth to thank all GNC division of Turin for welcoming me to the company, always showing respect and helping me to feel part of a team.
A special gratitude goes to the academic relator of the document, Sabrina Corpino, for the continuous openness and trust to the submission of my application to the internship and to the final presentation.
Lastly, but not least, I want to thank my parents to keep advising and giving me the opportunity to follow my ambitions.

Abstract

The present document is written for the Master degree course in Aerospace engineering of Politecnico di Torino, during a six month thesis internship in Thales Alenia Space site of Turin (TO), Italy.
Scope of the work is to build a simulation tool, in Matlab environment, of FGS attitude sensor, supporting tests on the Engineering Qualification Model of the same. The sensor is designed in the context of European Space Agency Euclid mission, whose optic scientific instruments require unprecedented pointing performances. Its main goal is to compensate thermoelastic deformations induced errors on attitude determination, being part of an innovative Attitude and Orbit Control System architecture.
More in details, the content of the document foresees in the first chapter an introduction to Euclid mission objectives and design. So, the second one continues the description, focusing on the architecture and operations in scientific mode of AOCS. Therefore, third chapter goes down to FGS design level, showing its main assemblies and functionalities. Furthermore, a summary of its modes of operation is presented, especially of the tracking ones and related timing sequences.
Fourth chapter, instead, introduces some spacecraft attitude determination techniques and it shows the theoretical bases of the ones applied by FGS sensor.
Fifth chapter treats the main contributors to performances of such a Charge Coupled Device based sensor and it lists measurement requirements and expected budget in terms of errors on attitude angles.
Core of thesis is the sixth chapter. It explicits the test process and the role of the simulator in support to that. There, other scripts written out of simulator, but fundamental in test input files generation, are also cited. They are partially reported in appendices A and B.
Seventh chapter describes the implemented model of FGS and it shows the results of specific scenarios. Consequently, the final chapter comments on them and it explores future applications.
Main script of simulator is reported in appendix C.

Contents

List of Figures

List of Tables

List of acronyms

ADCS Attitude Determination and Control System

AME Absolute Measurement Error

AOCS Attitude and Orbit Control System

AP Acquisition Phase

ASW Application SoftWare

ATM Absolute Tracking Mode

BRF Boresight Reference Frame

BSW Basic SoftWare

CAF Catalogue Auxiliary File

CCD Charge Coupled Device

CDF Cumulative Distribution Function

CDMU Command and Data Management Unit

CHM CHeckout Mode

CTE Charge Transfer Efficiency

ECSS European Cooperation for Space Standardization

EMC ElectroMagnetical Compatibility

EMI Electro-Magnetic Interference

EQM Engineering Qualification Model

ESG Electrical Stimuli Generator

FGS Fine Guidance Sensor

FOV Field Of View

FPA Focal Plane Assembly

FWA Filter Wheel Assembly

GC Galaxy Clustering

GS Ground Station

GWA Grism Wheel Assembly

HDSW Hardware Dependent SoftWare

HEALPIX Hierarchical Equal Area isoLatitude PIXelation

IAU International Astronomical Union

ICRF Inertial Celestial Reference Frame

ISC Input Star Catalogue

ISCF Input Star Catalogue File

MMU Mass Memory Unit

MPS Micro-Propulsion Subsystem

MRF Measurement Reference Frame

NEA Noise Equivalent Angle

NISP Near-Infrared SPectrometer

OBC On Board Computer

OBCF On Board Catalogue File

OBSC On Board Star Catalogue

OBSW On Board SoftWare

OGA On Ground Algorithm

PCDU Power Control and Distribution Unit

PDF Probability Density Function

PEC Proximity Electronic Channel

PEM Proximity Electronic Module

PHM PHoto Mode

PLM PayLoad Module

PM Processing Module

PSF Point Spread Function

RME Relative Measurement Error

RTM Relative Tracking Mode

RW Reaction Wheel

SBM StandBy Mode

S/C SpaceCraft

SCM SCientific Mode

SNR Signal to Noise Ratio

SSE Sun-Spacecraft-Earth

STR Star TRacker

SVM SerVice Module

SUSW Start Up SoftWare

TP Tracking Phase

VIS Visual InStrument

WL Weak gravitational Lensing

Chapter 1

Introduction - Euclid overview

The Euclid mission is part of the European Space Agency Cosmic Vision Program and it sees as prime contractor Thales Alenia Space. The latter is responsible for all the S/C subsystems development apart from the PayLoad Module instruments and science ground segment (assigned to Euclid Mission Consortium). The project has passed the Critical Design Review (CDR) under the lead of the European Space research and TEchnology Centre (ESTEC, Noordwijk, NL) and currently it is undergoing the integration and test phase in Turin (Italy).
Mission's primary objective is to answer some of the biggest scientific questions related to dark matter, dark energy and gravity. It will furthermore help to better understand the physics of the early universe and the formation of cosmic structures. "Euclid survey will show how cosmic acceleration modifies the expansion history and the 3-dimensional distribution of matter"[1] focusing on shapes and accurate redshifts measurements of galaxies. The evolution model from the Big Bang to our epoch, in fact, includes two components of energy density whose nature is unknown: 76% constituting the so-called *dark energy* and 20% of *dark matter*.
Many theoretical ideas are spread in time, but all of them "will only change observational signatures by tiny amounts that can only be decisevely distinguished by using high-precision astronomical surveys covering a major fraction of the sky"[1]. To do so, sophisticated scientific instruments are needed: a visible imager, a near infrared photometer and a slitless spectrograph. They allow to analyze dark universe through two investigations: *weak gravitational lensing* and *galaxy clustering*. The first instrument is called Visual InStrument and it works in 550-900 nm wavelength range, while the second and the third are integrated in Near-Infrared SPectrometer that operates between 920 and 2000 nm. All these receive light from a 1.2 m Korsch telescope.
The mission is scheduled to be launched in late 2022 (from Kourou by a Soyuz ST-2.1B) and will last more than 6 years, performing science operations orbiting around Sun-Earth Lagrangian point 2 (SEL2). It uses a step-and-stare strategy of observation divided in Wide and Deep survey, that ensure respectively primary and

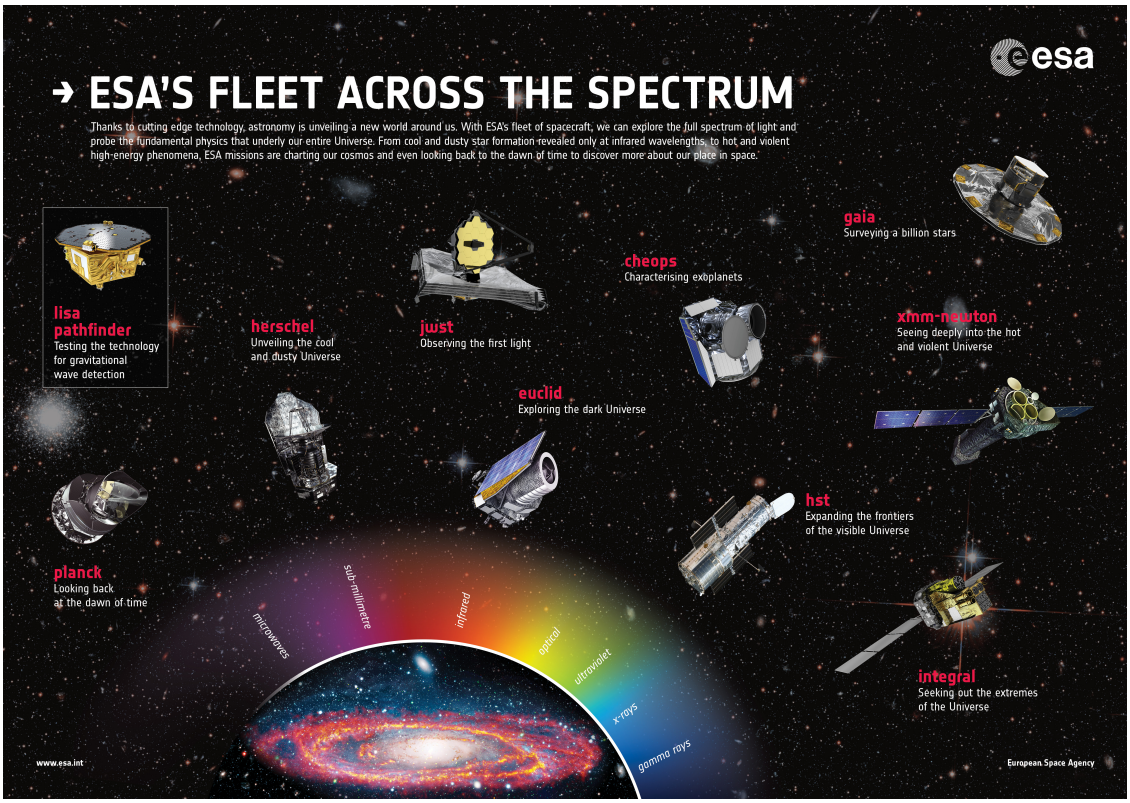


Figure 1.1: ESA fleet across the spectrum (2017) [10]

legacy science. It nominally consists in a dithering pattern of 4 frames and related exposure time. Calibration (non-science) data are also extremely important, due to stringent pointing requirements.

Scientific objectives

Primary objective of the mission is to measure accelerated expansion of the universe using a large-scale structures investigation. Weak Lensing is performed by determination of shape and shear of galaxies, through visual and near-infrared photometry, while Galaxy Clustering exploits the redshift with near-infrared spectrometry. All these are captured by Charge Coupled Devices and HgCdTe detectors, as shown in figure 1.3.
Current *concordance* cosmological model is based on two untested assumptions: dark energy is responsible for universe acceleration, while non-barionic dark matter owns a gravitational field but it does not absorb or emit light. In recent missions the focus was to measure, as precisely as possible, the temperature fluctuations of the Cosmic Microwave Background, like in ESA's Planck. It confirmed some assumptions on structures formation but it weakly probes the subsequent 13 billion

years expansion.
Euclid will be able to address 4 key questions (here simplified):

- Is dark energy a cosmological costant or is it something that evolves dinamically?
- Is the apparent acceleration a breakdown of General Relativity on large scales or is it the probe of a theory failure?
- What is the dark matter? What is the neutrino and relativistic species' contribution?
- Are primordial power spectrum fluctuations described by a Gaussian distribution?

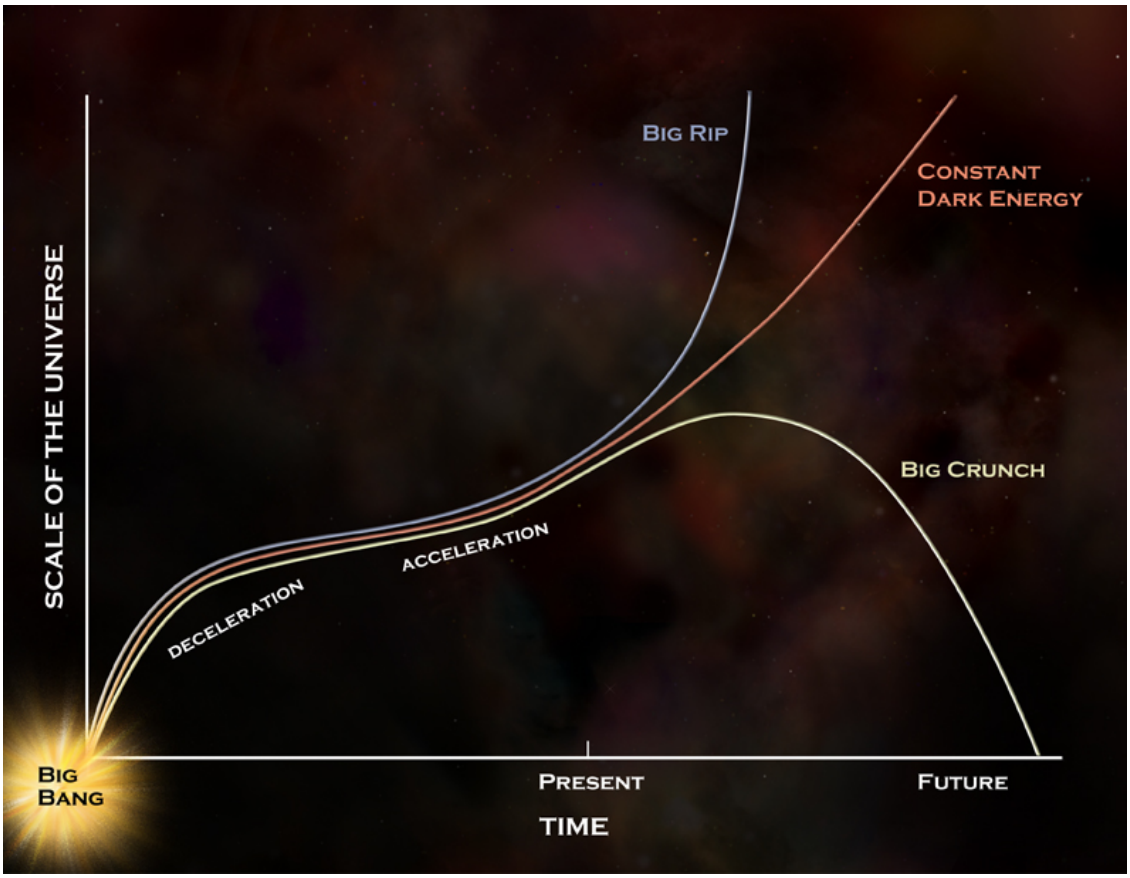


Figure 1.2: Evolution history of the universe [11]

WL can measure the growth history of the universe and dark matter distribution through light, emitted by far galaxies, distorption caused by perturbation in the

path of photons and, using photometric redshifts, it can be done in three dimensions. GC, instead, is performed by 3-D mapping of galaxies and spectroscopic redshifts measuring. There are also secondary results from these observations, however they fall outside the purpose of this thesis.

PAYLOAD					
Telescope	1.2 m Korsch, 3 mirror anastigmat, f=24.5 m				
Instrument	VIS	NISP			
Field-of-View	0.787×0.709 deg^2	0.763×0.722 deg^2			
Capability	Visual Imaging	NIR Imaging Photometry			NIR Spectroscopy
Wavelength range	550– 900 nm	Y (920-1146nm),	J (1146-1372 nm)	H (1372-2000nm)	1100-2000 nm
Sensitivity	24.5 mag 10σ extended source	24 mag 5σ point source	24 mag 5σ point source	24 mag 5σ point source	3 10^{-16} erg cm-2 s-1 3.5σ unresolved line flux
Detector Technology	36 arrays 4k×4k CCD	16 arrays 2k×2k NIR sensitive HgCdTe detectors			
Pixel Size Spectral resolution	0.1 arcsec	0.3 arcsec			0.3 arcsec R=250

Figure 1.3: Payload characteristics [1]

Mission design

As already said, Euclid will launch from Kourou and will perform a direct transfer (about 30 days) to an high amplitude orbit around SEL2, exploiting a Fregat ascent trajectory. Then there will be some correction maneuvers due to launcher dispersion and fine-targeting, so there won't be necessary inserction ones. The final orbit will lie in a plane nearly perpendicular to ecliptic and in such a way the S/C can mantain its orbit without excessive station-keeping and attitude big adjustments. The sun-shield intercepts sun light and keep telescope and all the electronics inside a tolerable range. Science orbit ellipticity, Sun-Spacecraft-Earth angle and visibility from Ground Station will be influenced by launch data and conditions.
This concept allows to scan the sky at high galactic latitudes, in fact preliminary analysis showed that at angles below 30° there could be the risk of not complying with area and depth observation requirements.
"The survey is built by starting observations at the region of higher density of galaxies, starting from the ecliptic poles where the zodiacal background is minimum, so to have best SNR in the early stages of the mission"[2]. Deep surveys, instead, can be used for calibration and interest precise targets close to the poles for visibility reasons.
Each survey is organized in elementary pointing sessions (fields) that in their turn

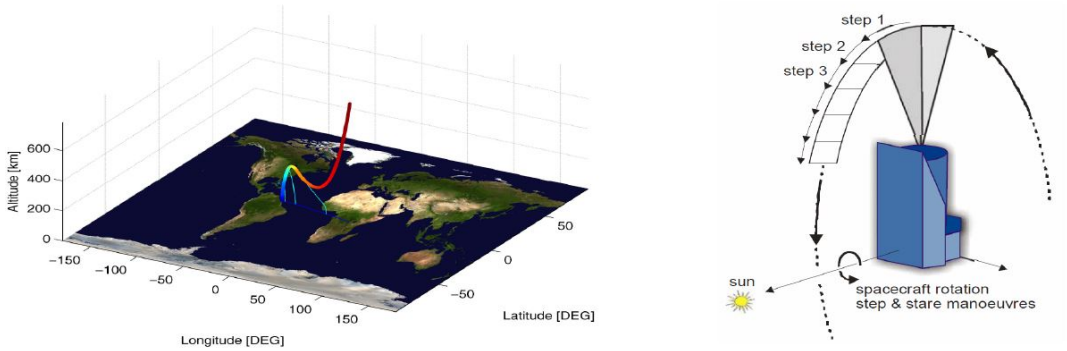


Figure 1.4: Launch phase and S/C orientation [1]

consist in 4 dithers, each of them containing an instruments operating sequence. There, therefore, are field slews up to 1.6° (290 s) and three subsequent dither slews that are about 120" around x body axis and 70" around y body axis (60 s each). The latters are fundamental to fill the gaps between VIS/NISP detectors[1].
To accomplish the attitude control of the overall mission, a combination of star tracker, gyroscope, cold gas micropropulsion, reaction wheels and a Fine Guidance Sensor (design described in chapter 2) is used. Instead, chemical (hydrazine) propulsion is necessary for orbital transfer corrections, station-keeping and large slew maneuvers (180°).

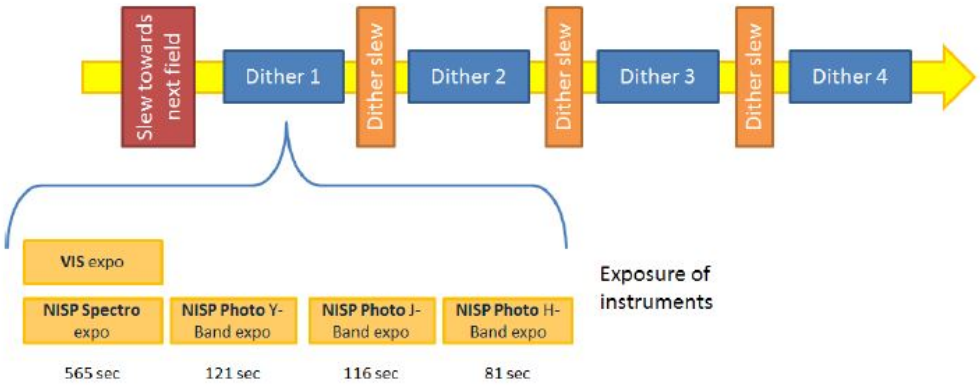


Figure 1.5: Euclid observation plan [1]

For communications needs, three low gain antennas in X band support telecommands and real-time station-keeping, while a steerable high gain antenna in K band

[1]It is, in fact, worth noting that between two adjacent pixels there are non collecting regions that have to be filled in next exposure. Body frame, instead, is defined in chapter 3.

is used to downlink science data (between scans it is repointed to minimize its induced perturbations). Therefore, two GS are chosen: one on Northern and the other on Southern hemisphere. These are envisaged because of dependency of K band link margin on elevation. Then, a solid state mass memory stores and encodes the compressed instrument data. SpaceWire links are enroled of reception of instrument science and housekeeping telemetry from on board computers, collected and distributed via MIL-Std-1553 buses.
Thermal design foresees an active control for payload (heaters operated with pulse width modulation), in order to garantuee high isolation and stability, while between sun-shield and top of SerVice Module a multi-layer insulation is applied. Telescope, then, has a thermal baffle (that mitigates also straylight). Radiators and blankets complete the configuration.
Solar panels are body-mounted on sun-shield coupled with a Lithium-Ion battery. Lastly SVM presents 6 panels and a central cone, in which propellant tanks are located. Each panel has a functional role: Telemetry and Telecommand (TT&C), Attitude and Orbit Control (AOCS), Central Data Management (CDMS) and Electric Power (EPS), payload and Fine Guidance Sensor (FGS) warm electronics (figure 1.7).

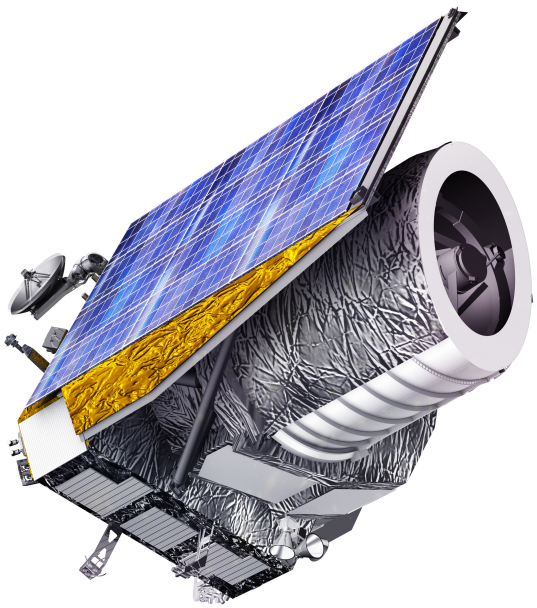


Figure 1.6: Euclid external configuration [10]

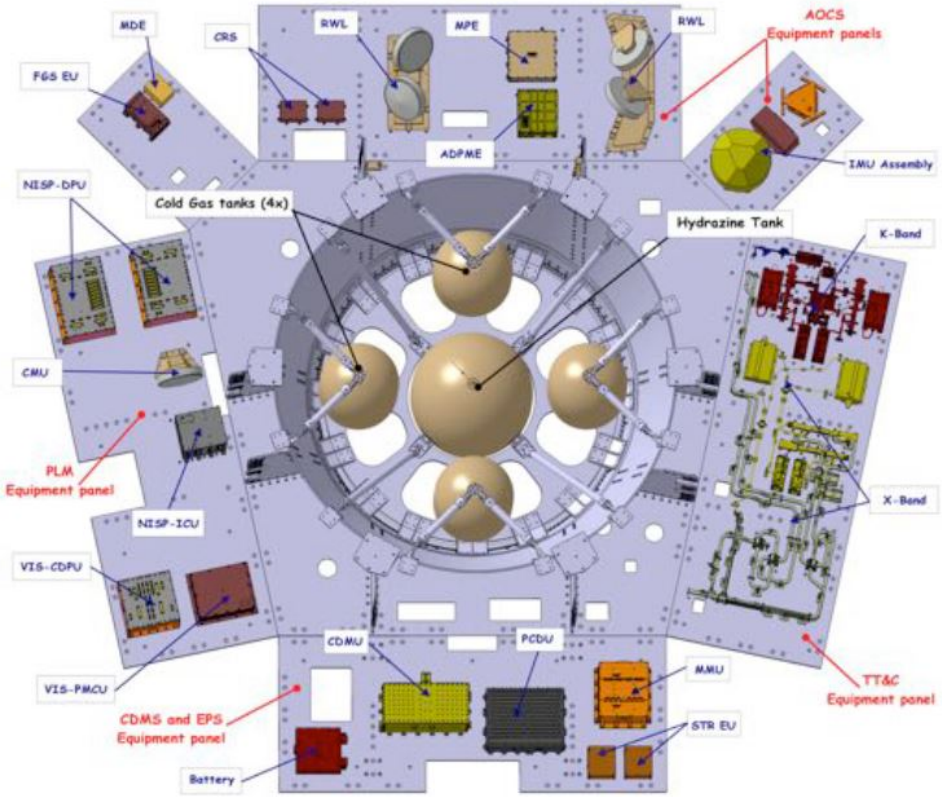


Figure 1.7: SVM equipment accomodation [3]

Chapter 2

AOCS in SCientific Mode

AOCS is a core system in such an observation mission, therefore it is necessary to describe its key elements: its sensors and actuators, their interrelationships and the way they operate in a scientific observation. Furthermore, the functional role of FGS sensor is here highlighted.

2.1 AOCS architecture

The overall configuration must control all the phases of the mission and includes:

- Star TRacker with 3 ortoghonal optical heads;
- Inertial Measurement Unit;
- Coarse Rate Sensors;
- Sun Sensors;
- FGS;
- 5 RWs (4 in tetrahedric configuration + 1 apart);
- redundant micro-propulsion thrusters set (2 x 6);
- redundant chemical propulsion thrusters set (2 x 10);

All these don't work simultaneously in each phase, but some combinations alternate. In transfer Trajectory Control Maneuvers, station-keeping maneuvers and disposal at end of life, two hydrazine thrusters (one for each 10 thrusters' branches) are used. The other eight thrusters, instead, provide force-free torques for angular momentum and attitude control in non-science modes.

Focusing on attitude determination in SCM[1], STR is not sufficient to meet the requirements, so a FGS is foreseen. This because STR is mounted on SVM and consequently subjected to thermoelastic deformation during large slews. For this reason its measurement is not enough accurate and it does not reflect the true telescope orientation.
The key problem in controlling such a mission is the necessity of fast and accurate large slews for survey's plan and, at the same time, very precise pointing and small jitter for quality image requirements. This is translated in a 25 mas (milliarcseconds) Relative Pointing Error over 700 s and 2.5 as Absolute Pointing Error (for the directions perpendicular to the telescope boresight, the most stringent ones), both with 99.7% confidence level.
An important design choice was, in fact, the definition of absolute and relative attitude measurement requirements. It means that S/C shall be capable of determining its attitude respect to Inertial Celestial Reference Frame, using an on-board star catalogue, and after a successfull "locking", it shall periodically monitors its stability, estimating current attitude relative to the locked one.

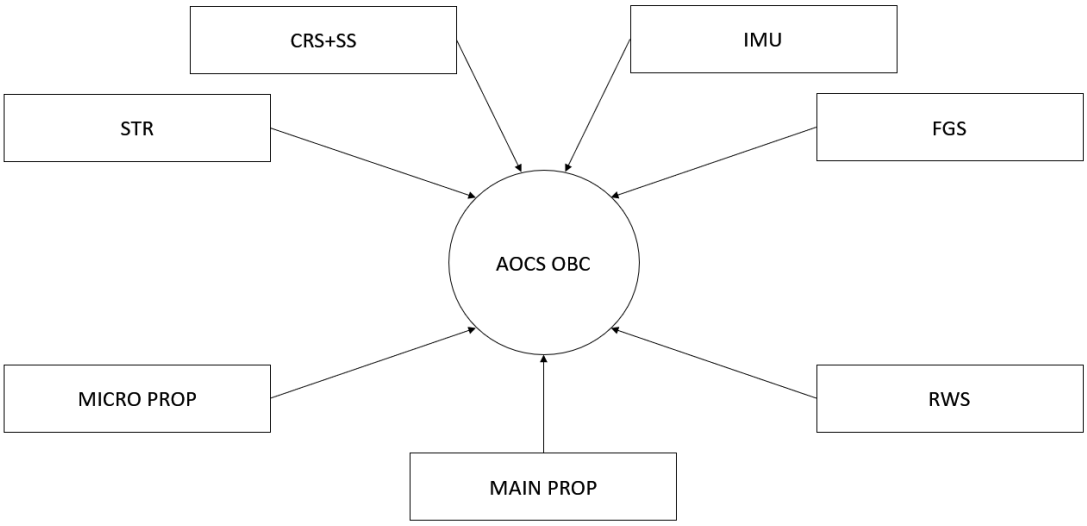


Figure 2.1: AOCS basic architecture

The requested agility implies the presence of high torque authority actuators (the Reaction Wheels), which however are not appealing for very stable pointing performance because of their intrinsic noise (micro-vibrations). For this reason,

[1]As already said, in scientific mode orbit control is not fundamental or innovative in its architecture, due to the fact Euclid will move around L2 and its primary mission requires almost exclusively attitude control.

Thales developed the so-called Hybrid Solution, where RWs and Micro-Propulsion Subsystem are used in synergy to cope with both the driving requirements (agility and stable pointing).
Furthermore, referring to figure 1.5, NISP has to change spectroscopic and photometric fields, producing noise contributors from its Grism Wheel Assembly and Filter Wheel Assembly (respectively for spectrometry measurements and for band selection). They, in fact, must be minimized through an ad-hoc mechanism, with a dedicated RW, that here is not detailed.

2.2 Operations in SCM

AOCS two main operative modes, relevant for control, in mission's SCM are:

- Relative Tracking Mode: FGS provides the relative attitude measurement by means of a relative quaternion error as a difference between the instantaneous attitude and the one measured at time of locking (at the first valid determination);
- Absolute Tracking Mode: FGS provides the absolute attitude with respect to the inertial reference frame (ICRF) as well as the relative attitude measurement (as above). It is based on the use of an on-board star catalogue, that is generated on field basis relying on a database of the entire sky sphere (this last is the Input Star Catalogue), whose management is later explained.

Furthermore, there are three sub-modes:

- Acquisition Phase: in this phase a set of targets, that is then used for attitude measurement, is determined. As input, FGS receives a first indication of attitude and angular rate, provided by STR, and it uses this reference while obtaining another result from its own CCDs. The image processing consists in: image integration, CCD readout (with 2x1 binning[2] and Run-Length-Encoding technique, explained in chapter 3), star like objects reconstruction, star selection, windows selection for next cycle, lock of the current attitude and save of the targets position, computation of exposure time for next cycle based on star magnitudes.
  For ATM it can be commanded in two ways: fine or coarse (respectively for small and large attitude uncertaintes). In the first case an enlarged window of 50 x 50 pixels is observed, while in the second a full CCD readout (similar to

[2]Binning is the readout of CCD by rows using a limited selection of pixels at a time, in this case 2 in vertical and 1 in horizontal. In this way the result is a mean value (in e^-) of the two.

RTM-AP) is performed. Therefore, a pattern recognition algorithm identifies the detected stars. In both cases, important parameters to select targets for next cycles are: position on CCD, magnitude and estimation position accuracy contained in the catalogue.

- Intermediate Cycle: This cycle is necessary to compensate star displacement before next phase. In fact, Euclid S/C is limited to have an angular rate of maximum 0,3 as/s, but it is enough to produce an image shift.
  It also uses enlarged windows of 50 x 50 and the following tasks are executed: attitude and angular rate calculation and validation, stars and windows selection for the tracking phase.

- Tracking Phase: FGS performs an enlarged windows image acquisition with an exposure time between 0.1 s and 1.5 s, then the readout of CCD data. Targets clustering follows, before a target selection procedure. In this one, magnitudes and coordinates on detector are considered, saving up to 10 brightest stars. A further check is performed, to assure that a hot pixel[3] doesn't fall inside the cluster, otherwise it is discarded. Validated targets will be used for attitude calculation.
  All the process is repeated periodically at 0.5 Hz.

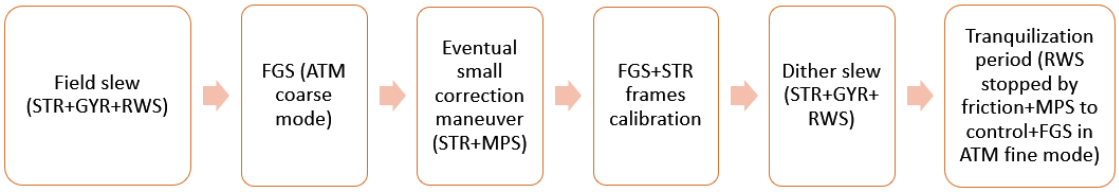


Figure 2.2: Observation concept of Euclid AOCS in scientific mode

From one field to another, the spacecraft performs a slew with star tracker and gyroscope as sensors and reaction wheels as actuators. At the end of the maneuver FGS is enroled to determine the attitude in ATM coarse mode, fundamental for compensation of thermoelastic effect between STR and the scientific payload. For this reason, FGS shares its field-of-view with VIS (design is thorough in chapter 3). If the measurement is within absolute pointing requirements the scientific observation can start, otherwise a small correction maneuver is performed with STR and micro-propulsion.

[3]Hot pixel is called a defective pixel that detects an excessive value of electrons, so its information isn't accurate. It differs from a saturated one, in which it reaches its maximum capacity and neiborough pixels could receive the excessive electrons, resulting in another worse error.

Before starting a new maneuver, STR and FGS reference frames are cross calibrated, in order to allow STR providing the best estimation at the end of the following dither slew. This foresees, at the same way of the field one, the cooperation of star tracker, gyro assembly and reaction wheels.
Afterwards, a tranquilization period is performed, during which the rotors slow until stop with their own friction. To control this phase, micro-propulsors are actuated and FGS operates in ATM fine mode, with an uncertainty of 5 as.

2.3 Star catalogue management

In order to have an inertial reference for absolute attitude determination an unprecedented star catalogue is needed, i.e. the Gaia catalogue Data Release 2. In a mission like Euclid there is, in fact, the necessity to scan each zone of the sky sphere, even if the most requiring part is represented by high galactic latitudes regions.
According to Gaia detectors sensibility, the brightest star is of magnitude 3 while the faintest is about 21. This last is well above the maximum requested from Euclid survey. Indeed, it can be demonstrated that for each detector at least 3 stars have to be found to recognize the pattern, but a number of 9 is used to improve attitude accuracy [4]. So, analyzing Euclid observation's plan and FGS Field Of View, magnitude 18 assures the success of determination all over the mission.
To manage such an amount of data, an on-board complete catalogue would be too demanding on various aspects:

- Storage inside the Mass Memory Unit requires indexing, which can be obtained through algorithms used on ground (ex. HEALPIX tessellation scheme in figure 2.3). They unfortunately need a search engine that is unsuitable for implementation on-board. With ad-hoc algorithms, on the other hand, there will be too long execution times;
- The FGS software should be able to download from MMU the correct file relevant to the observed sky portion and extract stars data. In the case of area of interest closed to the boundaries between files, multiple searches could be required;
- The extracted data have to be transmitted to FGS through very slow bus.

The chosen solution includes a reference catalogue that is maintained on ground and a part periodically uplinked. This becomes an on-board catalogue that resides in FGS memory and contains guide stars and triads[4] for the current observation.

[4] "Triads are sets of data (relevant to a triangle of stars) needed by the FGS at beginning of each observation to recognize the star pattern"[4]. The TRIADS method is described in chapter 4.

It is generated in ASCII format through the so called On Ground Algorithm and converted in binary before being sent to S/C.

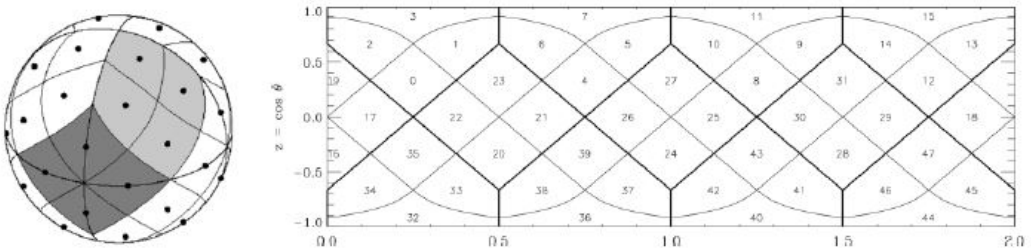


Figure 2.3: HEALPIX tessellation scheme [4]

Then On Board Star Catalogue is updated on board in such a way that it contains multiple files (all with a pre-defined length to simplify the data management), differentiated by an ID for each field. While FGS is in stand-by mode, the selected file is upload on AOCS OBC. Therefore, it is distributed to FGS software (via 1553 bus) and then reconstructed in its original structure, getting ready for attitude determination.

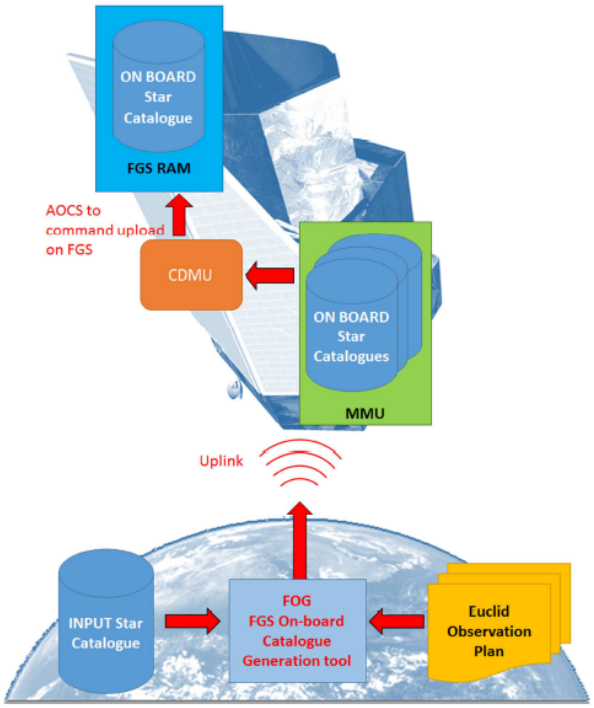


Figure 2.4: Star catalogue management scheme [4]

Chapter 3

FGS design

Peculiarity of FGS is its physical location in PayLoad Module and at the same time its functional dependence to AOCS. FGS detectors are, in fact, located in the same mechanical structure supporting the VIS focal plane assembly, although being independent from the structure of the same.
All the design is driven by the necessity of minimize the effect of thermoelasticity in attitude determination. This because data provided by STR is affected by the bias between itself and payload.
This led to an architecture with FGS detectors that share their focal planes with VIS, located off axis from the telescope. Accordingly to this, the body frame is defined to have x and y axes lying on the common focal plane, while z axis corresponds to the boresight vector of VIS.

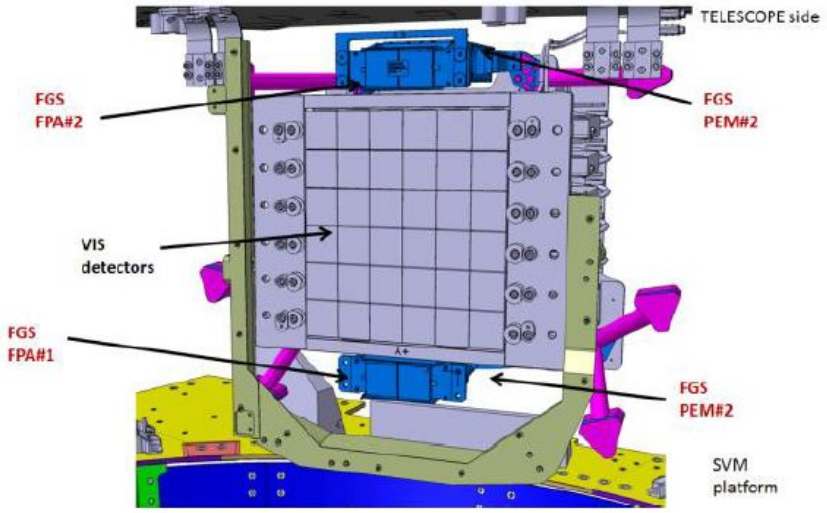


Figure 3.1: FGS location wrt VIS [4]

3.1 General architecture

The general composition foresees:

- 2 Focal Plane Assemblies, one on each side of VIS focal plane (6 x 6 CCDs): a FPA contains two CCDs;
- 2 Proximity Electronics Modules, each one commands two CCDs;
- Thermal and magnetic shields (divided in upper and lower) between FPAs and PEMs;
- 1 Electronic Unit.

Each FPA is connected to PEM with two flex harnesses that constitute the channels for their specific CCD. Each channel then gets commands and sends telemetries via spacewire link with EU. This last provides also the power, in turn, received from Power Control and Distribution Unit and it exchanges data to and from Command and Data Management Unit via MIL-STD 1553 bus.
In nominal conditions two CCDs are active, one for each side of the telescope axis. Reliability, furthermore, is strengthned by EU internal redundancy.

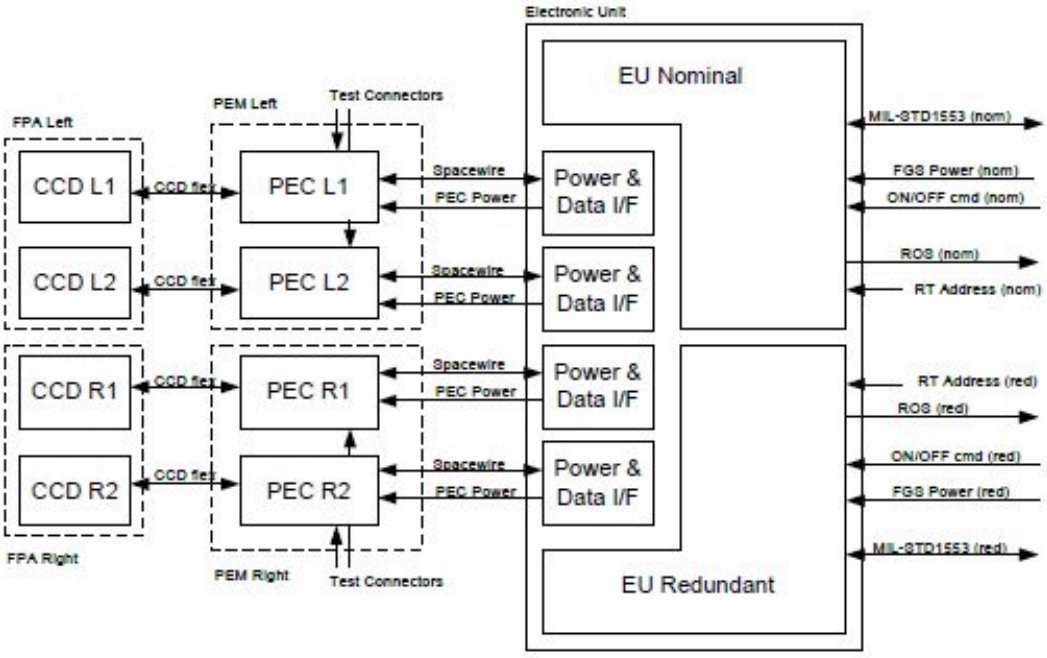


Figure 3.2: FGS electrical architecture [4]

For what concerns structures, two sides of FGS are identical apart from lateral brackets. These hold the sensor up to PLM supports and are asimmetric due to

accomodation constraints. Then, FPA is supported by VIS focal plane and PEM rear part rests upon a radiator.
All this results in an hyperstatic structure (three fixation points per PEM), with a flexibility obtained through the already mentioned brackets. In fact, these are also necessary because module integration flow doesn't permit any regulations respect to the radiator.
In the next sections of this chapter the functionality and design of each element of the architecture are described.

3.2 FPA assembly and CCD readout

FPA assembly, in particular, consists of: SiC plate of two detectors, the detectors themselves, the 4 CCD flexes, the upper Electro-Magnetic Interference shield and the upper thermal one. The shields and the plate are mechanically linked with

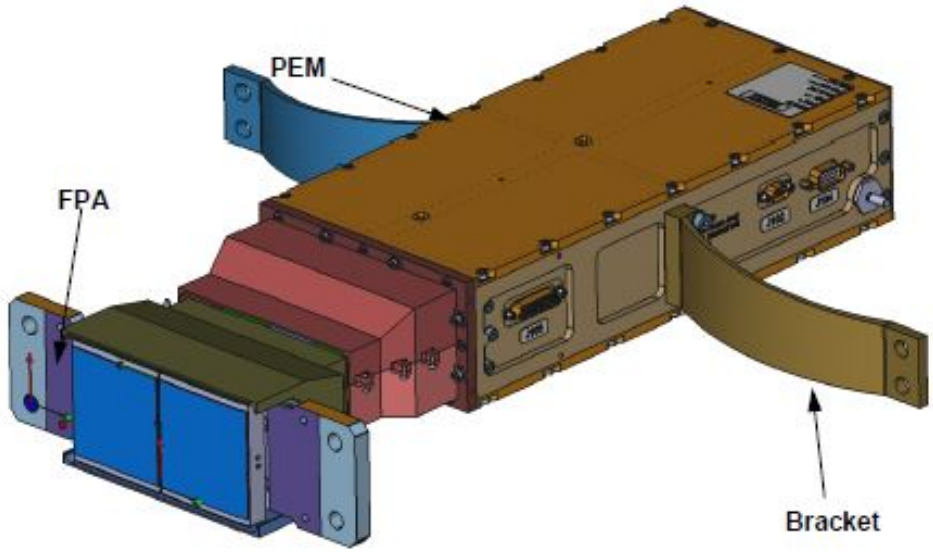


Figure 3.3: FGS external view [Courtesy of Thales Alenia Space]

4 standoffs, while the cables inside are free to move. This is necessary to decouple FPA plate and PEM. So, the shields covers the flexes until detectors surface.
The four standoffs are shaped to minimize the effect of shields deformation on detectors plate. This last, indeed, is made in SiC in order to match perfectly the coefficient of thermal elasticity of the CCDs case and that of the mechanical structure on which FPAs (VIS and FGS) are mounted.
Due to constraints of FPA to PEM connection, the two CCDs are oriented with inverse polarity. Furthermore, errors in position of FPA with reference to VIS plate

and those of detectors respect to FPA plate have to meet specific requirements, because they are fundamental in reconstructing the attitude.
Functionality of the shields is also a key part of design. Thermal isolation, in fact, provides radiative decoupling between warm FGS proximity electronics and cold FPA. EMI shield, instead, is needed to avoid the disturbance to VIS FPA deriving from CCDs high frequency readout.
The shields are supported directly by FPA to make EMI one closely follow CCD size and thermal one to be kept at large distance from PEM.

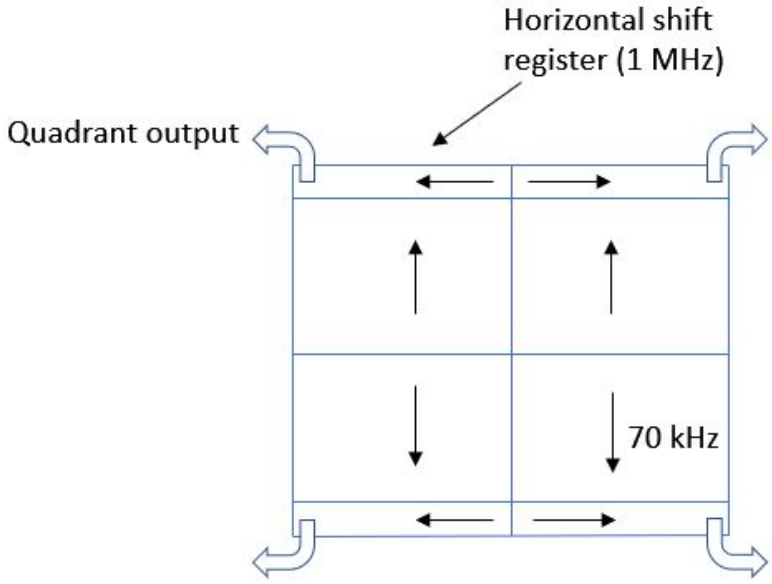


Figure 3.4: CCD readout process

Focusing on detectors they are type 4096 x 4096 pixels, in turn divided in 4 quadrants like in figure 3.4. They are read row by row with a frequency of inter-rows of about 70 kHz and an inter-pixels of about 1 MHz. CCD is affected by typical errors that are later explained (section 5.1) and their output (analogic) is measured in e^- proportionally to detected current.

3.3 PEM assembly

PEM is composed by two Proximity Electronic Channels, their cover, the interface with radiator, cover for connectors and the lower parts of the shields. Then, four standoffs mounted on the connectors cover hold another thermal shield.
Each PEC is constituted by three boards connected together by flexis and screwed

on a stiffner that is a single piece with lateral frame. They are also screwed together and with the external cover to form a box.
The PEM electronics have the following functions:

- to bias the CCD detector;
- to scan CCD;
- to process the analogue signals;
- to convert them into digital information;
- to process digital information;
- to transmit digital data to the EU;
- to receive and execute commands from EU;
- to transmit temperature, PEM house-keepings and other monitoring data;
- to accept test's digital pixel data and to provide scan of test's signals (electro-optical stimulation).

Pixel pre-processing functions derive from star trackers. In particular, they perform: background estimation, pixel thresholding, segment detection and Run-Length-Encoding.
All these are tools for the reconstruction of an image from pixel data. More in details, a specific algorithm detects signal along consecutive pixels that is above current background plus a threshold (electrons level). Then, RLE allows to recognize segments through a CCD row scanning: the segment starts when the in-coming pixel has an over-threshold flag asserted, while a pixel with the same flag de-asserted defines the end of the segment.
In this way a target can be reconstructed unifying adjacent segments as shown in figure 3.5. Furthermore, additional segment parameters are evaluated: pixel coordinates, length, energy, weighted energy and background value. All these data are provided to EU via Spacewire interface.

3.4 EU

The electronic unit is located in SVM and is enroled of elaboration and control of FGS. Each of its two redundant processing modules performs the following tasks:

- to interface the S/C on-board computer;
- to interface the power distribution module;

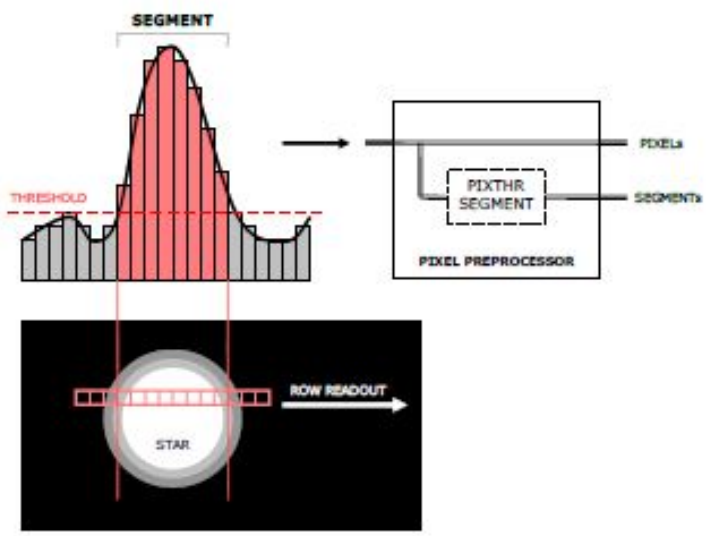


Figure 3.5: Pixel pre-processing concept [Courtesy of Thales Alenia Space]

- to interface PEMs to receive pre-processed pixel data and to command different operative modes;
- to provide the processing power in terms of centroiding, tracking and attitude calculation.

The two Processing Modules are operated in cold redundancy while most functions related to PEM powering and PEM interfaces are in hot redundancy, since also power distribution modules are redundant.
Modules are horizontally stacked in a box, with a base plate that has the double function to fix the unit to the SVM panel and to drain the heat of the unit to the conductive heat sink.

3.5 Software and mode of operations

In EU the On Board SoftWare that manages the attitude sensor is stored and executed. It deals with all the data passing through MIL-STD 1553 bus and Spacewire link.
OBSW consists of two elements: the Basic SoftWare and the Application SoftWare. The first is, in turn, divided in:

- Start Up SoftWare that implements the start up code (initialization) and the MaiNtenance Mode;
- Hardware Dependent SoftWare which provides the library for Application SoftWare.

The second, indeed, foresees the following modes:

- StandBy Mode;
- CHeckout Mode;
- Relative Tracking Mode;
- Absolute Tracking Mode;
- PHoto Mode.

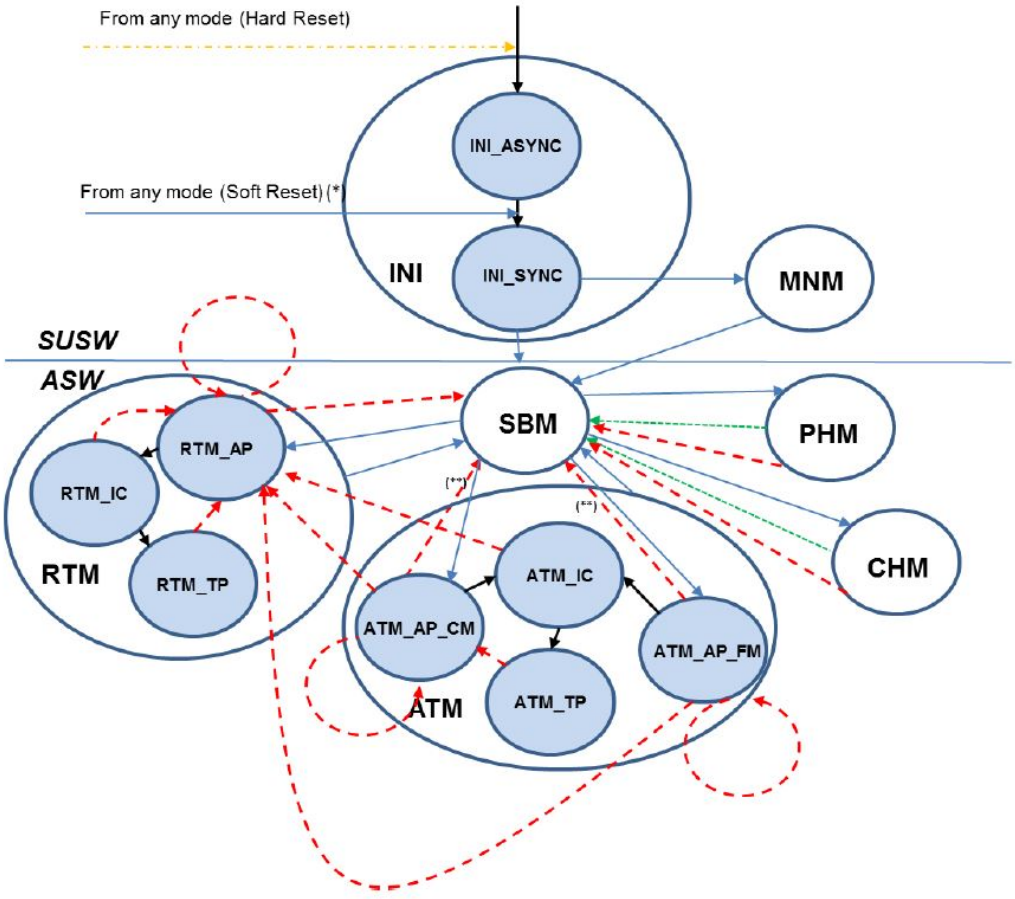


Figure 3.6: OBSW state transitions [Courtesy of Thales Alenia Space]

While BSW is running, PECs are switched off. The initialization mode enters automatically after each reset or switch on and performs initial checks. After that, a transition to SBM or MNM follows. This latter allows to download, check and

modify the on board RAM and ASW EEPROM.
During SBM, FGS is ready to pass to CHM, RTM, ATM and PHM. On telecommand, the selected PECs are activated, then, after an autonomous health check, scanning of CCDs in window mode is executed (identifying also hot pixels). It is also loaded the catalogue for ATM and it is possible to change a set of parameters like thresholds, number of stars etc.
In PHM, FGS is able to capture and send a part of CCD image, for data acquisition or download. The mode ends after acquisition phase is terminated and the results are transmitted to CDMU. So, it can be divided in acquisition and transmission phase. The first can be completed with raw pixel data in window mode, without RLE. An alternative, instead, uses the encoding technique and can be commanded also as full frame acquisition in segment mode (the same approach of STR). The transmission is commanded in full o low resolution, in order to or not occupy FGS memory.
CHM is used for on ground tests and in flight troubleshooting. The checkouts are more detailed respect to the ones in SBM and can implement a failure isolation strategy. Important is the charge injection capability that can verify different CCD parameters without external stimulation. During it, voltage levels in every CCD are switched on and off alternatively to produce a square waveform. This capability, furthermore, can be used in case CCDs are affected by in orbit radiation and have to be cleaned.

3.6 Tracking Modes timing sequences

Since RTM and ATM functions are already explained, it is here described the timing sequence of their subphases.

- RTM/ATM-AP starts with a reset considering CCDs in SBM. After that the exposure takes time: 1.5 s in nominal conditions, less if the targets are in a particularly crowded zone of the sky. The sequence is synchronized respect the half-exposure time as shown in figure 3.7. After this, there is the readout phase with a fixed period and the remaining time until 6.09 s is filled by the image processing (and future windows selection).

- RTM/ATM-IC follows the same transitions of AP, with the difference that the exposure time is nominally of 1.6 s. The processing will be performed in order to stay within 3 seconds duration for the entire cycle.

- RTM/ATM-TP foresees the readout in window mode with areas of 13 x 13 pixels size. The cycle has a period of 2 seconds and the key parameters are

the exposure time maximization, the sufficient time margin for processing operations and the maximum latency of a data. It, in fact, has to be not older than 1.3 seconds before being used to compute attitude states.
As shown in figures 3.8 and 3.9, mid exposure time is fixed at the fifth communication frame. The readout starts after exposure is considered concluded and it is followed by the processing. The exposure is arranged to a shorter value in case of particularly crowed fields and an idle state (not shown) and a cleaning phase happen before.
The phase terminates with the windows addressing: for next iterative cycle, up to 10 windows (not overlapping among them) containing a target are identified. It is considered a margin from CCD external borders and the four quadrants intersection.

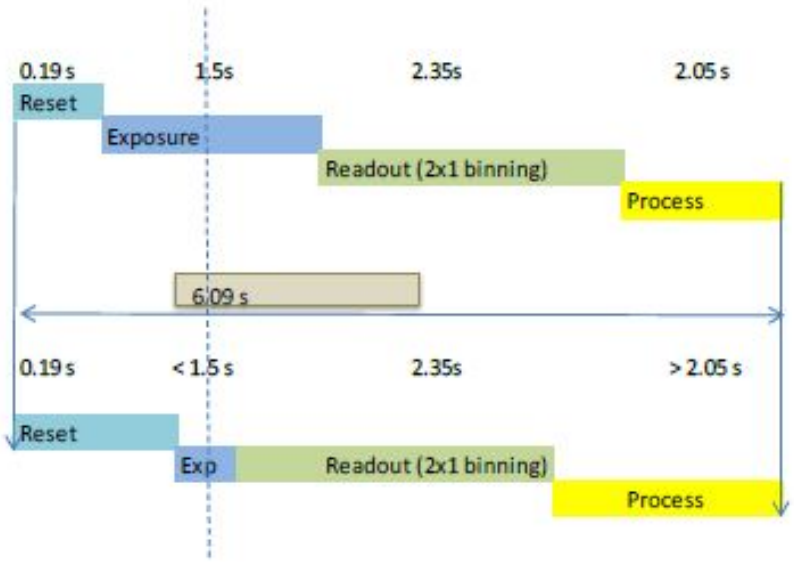


Figure 3.7: Acquisition timing sequence [Courtesy of Thales Alenia Space]

In ATM there are small differences, in particular in: the differentiation between acquisition coarse and fine modes, the use of catalogue.
For the first, coarse is used when a large level of uncertainty in attitude knowledge is requested, so segments of full CCD are acquired (it lasts about 6 s like in RTM). In this case, it is also used a pattern recognition algorithm that compares measured and reference triads. The process, then, can be carried on by two CCDs at a time or using fused data of the two, depending on the availability of targets.

Fine mode uses enlarged windows [1], up to 10, and a simplified pattern recognition based on the interdistances. It means that each combination of couple of stars is identified with an inter-stars distance that is put in relation with catalogue information (the entire cycle lasts about 3 s).
For what concerns catalogue utilization, it is divided in a triads one and a star one. The first is used for recognition purposes, the second for attitude computation.
In intermediate cycle and tracking, for each window, after target clustering, a target selection procedure is performed. Magnitude, distance from observer, hot pixel affection and a probability quality index are taken into consideration.
Anyway, the core functionality is the *attitude lock*. This is called the procedure through which an attitude is obtained at the first valid (with at least three detected targets/stars) cycle. After it, the set of selected stars remains the same for the next tracking cycles and a quaternion is computed, each time, relative to the locking one.
New quaternions are validated through a congruence check for angular rates, and the same rates are checked to be inside an expected range.

[1] Every time a window mode is used, a congruence check is executed in order to avoid windows overlapping.

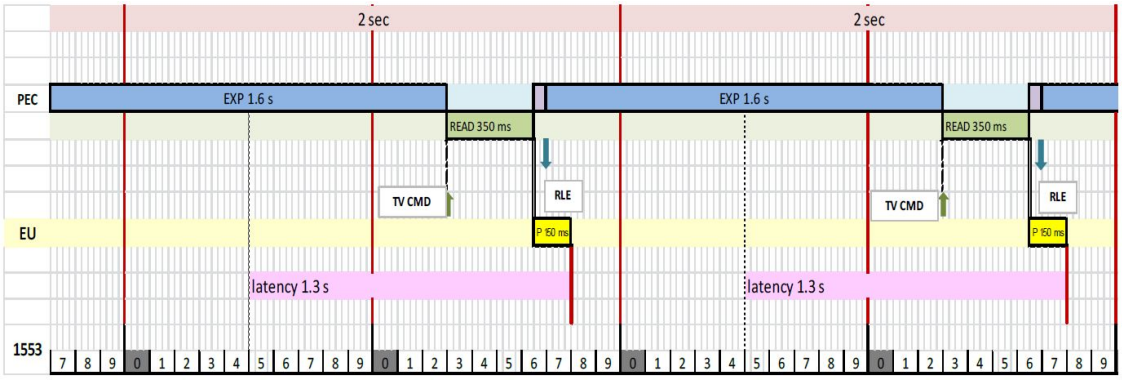


Figure 3.8: RTM/ATM-TP timing at 0.5 Hz and fixed exposure time [Courtesy of Thales Alenia Space]

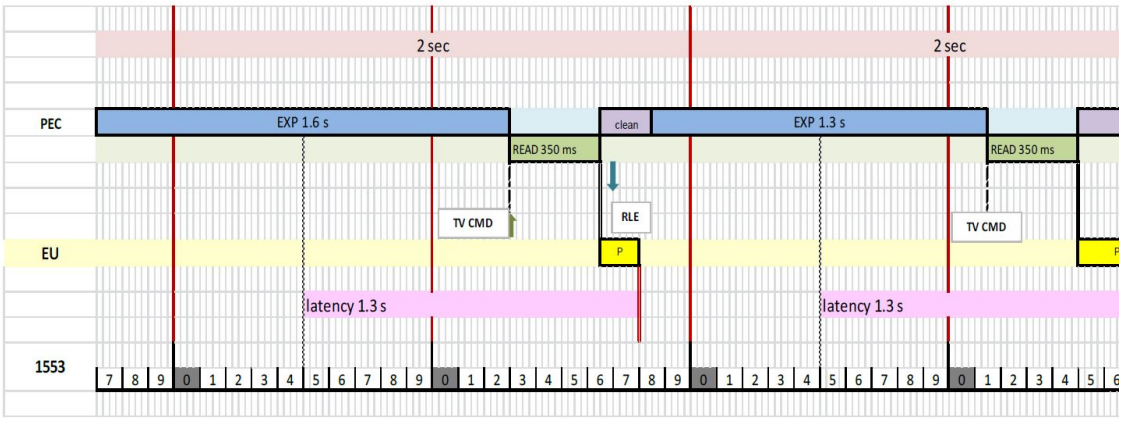


Figure 3.9: RTM/ATM-TP timing at 0.5 Hz and variable exposure time [Courtesy of Thales Alenia Space]

Chapter 4

Attitude determination techniques

In this chapter the attitude determination methods used by FGS are described. TRIADS is used for pattern recognition, while q-method, solution to the so called Problem of Wahba, makes possible to compute the S/C quaternion components. Before that, however, it is necessary to introduce the inertial and body reference frames of interest.

4.1 Reference frames

As well known, the goal of ADCS in a S/C system is to determine and control the orientation of body frame respect to an inertial one. Hence the definition of these two elements is fundamental.
From all the possible inertial ones, for Euclid mission the International Celestial Reference Frame was chosen. This is due to the fact that stars present in the catalogue (referring to the one that is not yet transmitted on board) are identified by right ascension and declination in the same frame.
ICRF is defined by $R_J\{O, \vec{i_J}, \vec{j_J}, \vec{k_J}\}$ where O is the center of mass of the solar system and the plane $\{O, \vec{i_J}, \vec{j_J}\}$ lies on that of the equator at the vernal equinox of epoc J2000. For this reason, the subscript is J while the vector $\vec{i_J}$ is pointing to Aries constellation at the same time. Its graphical representation is in figure 4.1.
So, calling the right ascension α and declination δ, the cartesian coordinates of a vector $\vec{r_J}$ can be defined as:

$$\vec{r_J} = r \begin{bmatrix} \cos\delta\cos\alpha \\ \cos\delta\sin\alpha \\ \sin\delta \end{bmatrix} \tag{4.1}$$

The body frame is represented by VIS reference frame that has the z axis aligned with the optical axis and the main plane, in which x and y axes lie, corresponds to that of the VIS focal plane (described in chapter 3).

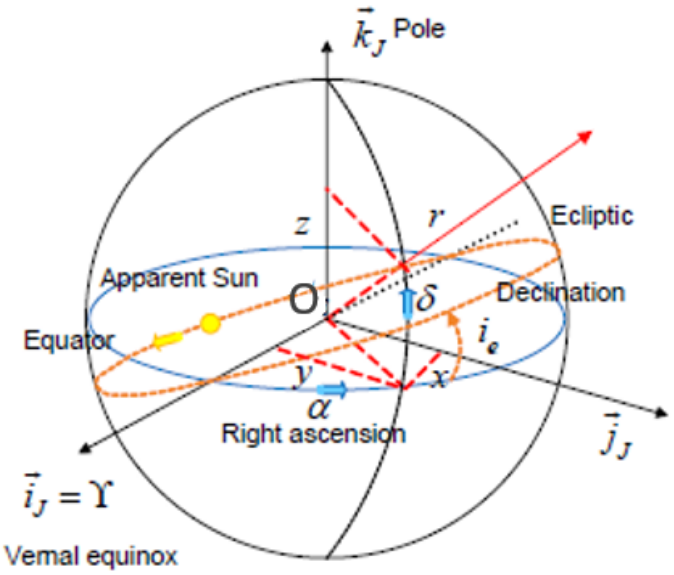


Figure 4.1: ICRF reference frame [5]

In the following sections, for convention, a direction in the inertial frame is known from catalogue and is called $\vec{r}$; the measured one, instead, is to be determined and is called $\vec{m}$ (figure 4.2). Both indicate the true direction $\vec{s}$ and, respect to that, have an error primarily caused by catalogue uncertaintes and sensor noise.
For each pair of directions $(\vec{r_k}, \vec{m_k})$, the attitude R_b^i (rotation matrix 3 x 3 from body to inertial) is given by the following expression:

$$\vec{r_k} = R_b^i \, \vec{m_k} \tag{4.2}$$

Since catalogue error is usually very small, $\vec{r}$ can be confused with $\vec{s}$. Furthermore, since all the named vectors are unitary and lie on the unit sphere, the error of the measured direction is normal to $\vec{s}$:

$$\begin{aligned} \vec{m} &= \vec{s} + \vec{\delta s} \\ \vec{s} \cdot \vec{\delta s} &= 0 \\ \vec{\delta s} &= \delta s_1 \vec{s_1} + \delta s_2 \vec{s_2} + \delta s_3 \vec{s} \end{aligned} \tag{4.3}$$

Error can be treated to have a gaussian random distribution with zero-mean value and covariance matrix:

$$\begin{bmatrix} \delta s_1 \\ \delta s_2 \end{bmatrix} \begin{bmatrix} \delta s_1 & \delta s_2 \end{bmatrix} = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \tag{4.4}$$

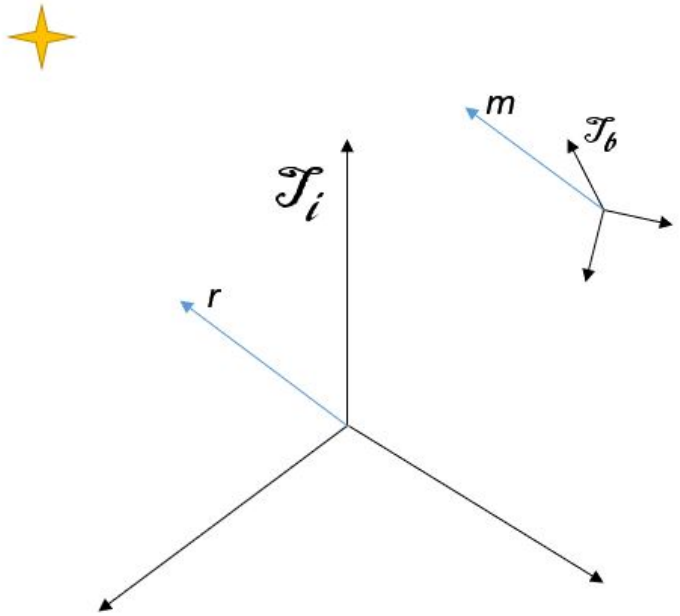


Figure 4.2: Directions convention for body and inertial directions

where σ is the standard deviation, considered the same for the two components. It can also be calculated the covariance S of $\vec{\delta s}$ respect to the true vector components (similar to measured components) [5]:

$$S = \sigma^2(\vec{s_1}\vec{s_1}^T + \vec{s_2}\vec{s_2}^T) = \sigma^2(I - \vec{s_3}\vec{s_3}^T) \cong \sigma^2(I - \vec{m}\vec{m}^T) \tag{4.5}$$

4.2 Determination methods

There are different methods to determine the attitude, the most used are:

- Cone intersection method;
- TRIADS method;
- Problem of Wahba.

The first two need a pair of not aligned directions $(\vec{r}_{1,2}, \vec{m}_{1,2})$, while the third is an optimization problem that uses the pair with the associated smallest uncertainty. As already said, the last two methods are used by FGS algorithms, so the description of the cone intersection method is not a content of this thesis.

TRIADS method

This procedure reaches a closed-form solution of the attitude matrix. To do so, a TRIAD orthogonal frame of reference is built:

$$R_t = \left\{ C, \vec{t_1} = \vec{s_1}, \vec{t_2} = \frac{\vec{s_1} \times \vec{s_2}}{|\vec{s_1} \times \vec{s_2}|}, \vec{t_3} = \vec{t_1} \times \vec{t_2} \right\} \tag{4.6}$$

A frame derives from measured directions (m_1, m_2) and another comes from (s_1, s_2). They bring to the following rotation matrix:

$$\begin{aligned} R_t^b &= \begin{bmatrix} t_{b_1} & t_{b_2} & t_{b_3} \end{bmatrix} \\ R_t^i &= \begin{bmatrix} t_{i_1} & t_{i_2} & t_{i_3} \end{bmatrix} \end{aligned} \tag{4.7}$$

and so it is immediate to compute the attitude from:

$$R_b^i = R_t^i \, (R_t^b)^T \tag{4.8}$$

This method does not look for the minimum error, i.e. not for minimal covariance. The latter matrix can be expressed as the one relative to the error vector (error on three Euler angles) $\delta\theta = (\delta\theta_1, \delta\theta_2, \delta\theta_3)$ [5]:

$$\begin{aligned} S_{\delta\theta} &= \begin{bmatrix} \delta\theta_1 \\ \delta\theta_2 \\ \delta\theta_3 \end{bmatrix} \begin{bmatrix} \delta\theta_1 & \delta\theta_2 & \delta\theta_3 \end{bmatrix} = \\ &= \sigma_1^2 I + \frac{1}{|\vec{s_1} \times \vec{s_2}|^2} \Big((\sigma_2^2 - \sigma_1^2) \vec{s_1} \vec{s_1}^T + \sigma_1^2 \vec{s_1}^T \vec{s_2} (\vec{s_1} \vec{s_2}^T + \vec{s_2} \vec{s_1}^T) \Big) \end{aligned} \tag{4.9}$$

where $\sigma_k^2 = \sigma_{r_k}^2 + \sigma_{m_k}^2$ is the sum of the variances of the reference and measured directions.

However, only the geometric tools of this method are used by FGS, exclusively for pattern recognition. So, an ad-hoc TRIAD catalogue (section of the OBSC) is stored on board and is updated at each observation, defining univocal TRIAD elements by triplets of stars with:

- spherical angles between j^{th} and k^{th} with respect to j^{th} and i^{th} stars;
- angular distance of the same;
- magnitudes;
- indexes.

Once the triplet is validated, i.e. its three stars are matched with those present in memory, the quaternion computation can start.

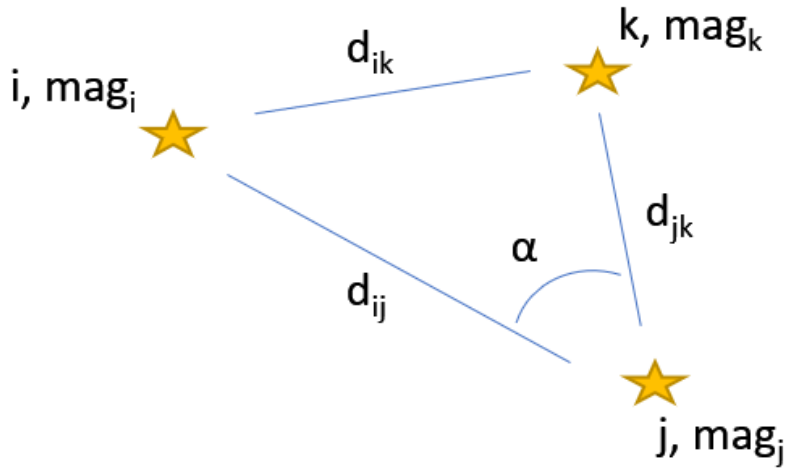


Figure 4.3: TRIAD element

Problem of Wahba and q-method

Since TRIADS is not an optimization method, the solution can not be found in case of more than two directions. Hence the need of solving the so called *problem of Wahba* [1].
This procedure allows to determine the set of directions that minimizes the error. This is done assigning to each pair (r_i, m_i) a weight such that:

$$\sum_{i=1}^{n} w_i = 1 \tag{4.10}$$

So, the solution is found searching the minimum of the square error functional [5]:

$$J(R) = \sum_{i=1}^{n} w_i |\vec{r_i} - R\vec{m}_i|^2 = [...] = 1 - \sum_{i=1}^{n} w_i \vec{r_i}^T R\vec{m}_i \tag{4.11}$$

where $R = R_b^i$ is the attitude matrix and $W = \sum_i^n w_i \vec{m}_i \vec{r_i}^T$ is the term where the weights act (the "weighted matrix").
In this case the error covariance is computed as for the previous method:

$$S_{\delta\theta} = \frac{1}{4}\sigma^2 \Big(I - \sum_{i=1}^{n} w_i \vec{s}_i \vec{s_i}^T \Big)^{-1} \tag{4.12}$$

In literature many solutions with many accuracies are presented. The main three are:

[1] So called because it has been afforded for the first time by Wahba in 1965 [5].

- direct solution;
- q-method;
- QUEST method.

The first one uses lagrangian multipliers, the second applies them exploiting quaternions, the third is a simplified form of the second.
The q-method is the one used for attitude determination in FGS algorithms and it is here described. It replaces the Euler parameters with quaternion components $q = \begin{bmatrix} \vec{q} & q_0 \end{bmatrix}$ ($\vec{q}$ the quaternion vector and q_0 the scalar value), using the Rodriguez formula [5]:

$$R(q) = 2\vec{q}\vec{q}^T + (2q_0^2 - 1)I + 2q_0\vec{q}\times \tag{4.13}$$

with:

$$\vec{q}\times = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{4.14}$$

The functional can be now expressed isolating the quaternion [...]:

$$J(q) = q^T \begin{bmatrix} w & \mathrm{w}^T \\ \mathrm{w} & W + W^T - wI \end{bmatrix} q = q^T Q q \tag{4.15}$$

where $w = \mathrm{tr}(W)$ and $\mathrm{w} = \sum_i^n w_i \vec{m}_i \times \vec{r}_i$.
Anyway the solution passes through the lagrangian multipliers λ. The functional in lagrangian form can be written, in fact, as:

$$L(q, \lambda) = \frac{1}{2} q^T Q q + \frac{1}{2}\lambda(q^T q - 1) \tag{4.16}$$

and so the minimum is found through the following derivatives:

$$\begin{aligned} \frac{\partial L}{\partial q} &= (Q - \lambda I)q = 0 \\ \frac{\partial L}{\partial \lambda} &= q^T q - 1 = 0 \end{aligned} \tag{4.17}$$

It means that, since the functional has to be minimized, the search is moved to the maximum eigenvalue of Q, i.e. $\lambda_{\max}$. Then, the quaternion is the normalized (with unitary module) eigenvector corresponding to $\lambda_{\max}$.
Finally, the result can be expressed in Euler angles passing from 4.13 and from attitude matrix definition.

Chapter 5

FGS performance contributors

From the practical point of view, measurements are affected by errors that can be classified on the causes of the same. It is worth thorough the argument for star trackers and FGS in particular, since they are functionally similar.
Before this, however, it is necessary to analyze their basic element, which is the CCD.

5.1 CCD characteristics and noise

The charge-coupled-device is a solid state device that can record the intensity of incident light and map it as a function of the coordinates on its surface. This is divided in rows and columns of *pixels*.
For what concerns the structure, instead, it is composed by 2 main strates. The first is the so called substrate and it is a doped silicon to obtain a p-type semiconductor (it usually contains phosphorus). The second, that is also the uppermost part, is a n-type equivalent (using boron). Above these two, an insulator material is used, usually i.e. silicon dioxide, and above it the electrodes, obviously conductors.
To understand how an external electron is captured, it is necessary to investigate what happens when the two semiconductor parts come in contact. At the beginning the free charges diffuse across the boundary until a *depleted* region is formed. This becomes completely free of charges and so it can support an electric field.
This latter, in the p-type depleted region, is directed downward and this yields an upward force on any free electron created by an external photon arrival in the region (exposure to light). In this way it is attracted toward and into the n-type region.
The transformation from photon to electron is due to the photoelectric effect, in which a light wave transfers energy, in the form of a kinetics one, to hitten atoms electrons until they escape and generate ions. Energy of a photon can be expressed

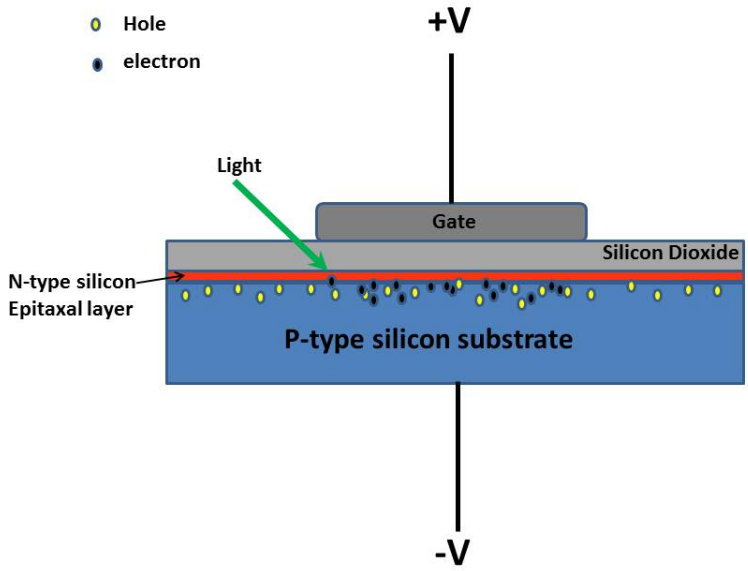


Figure 5.1: Side view of CCD structure [12]

by Planck equation:

$$E = hf \tag{5.1}$$

where $h = 6.626 \cdot 10^{-34}$ J·s is the Planck coefficient and f the wave frequency. So, a detector is defined through its quantum efficiency, i.e. the ratio of new electrons out of incident photons.
In addition, each pixel is constituted by three of the structures just described and therefore is linked to three electrodes. These can be commanded such that the potential can vary in a desired direction. All the labeled "1"/"2"/"3" electrodes of a row/column are connected to a common bus in order to flow charges in a synchronized way (figure 5.2).
Then, they are collected in a shift register containing the data stream that, before being processed, is sent to a field-effect transistor (capture and amplification) and translated from electrons to digital numbers by an analog-to-digital converter. Before restarting the exposure, the entire process can be repeated multiple times to clear the CCD from residual charges. In fact, some electrons may remain trapped between pixels during the readout, lowering the so-called Charge Transfer Efficiency, i.e. the performances of the device.
Moreover, it must be kept in mind that each CCD has a range of frequencies of detection, as well as a maximum level of electrons accumulated on a pixel. This leads to the already cited problem of the pixel saturation.
Besides the above ones, CCDs are also subjected to many others systematic and

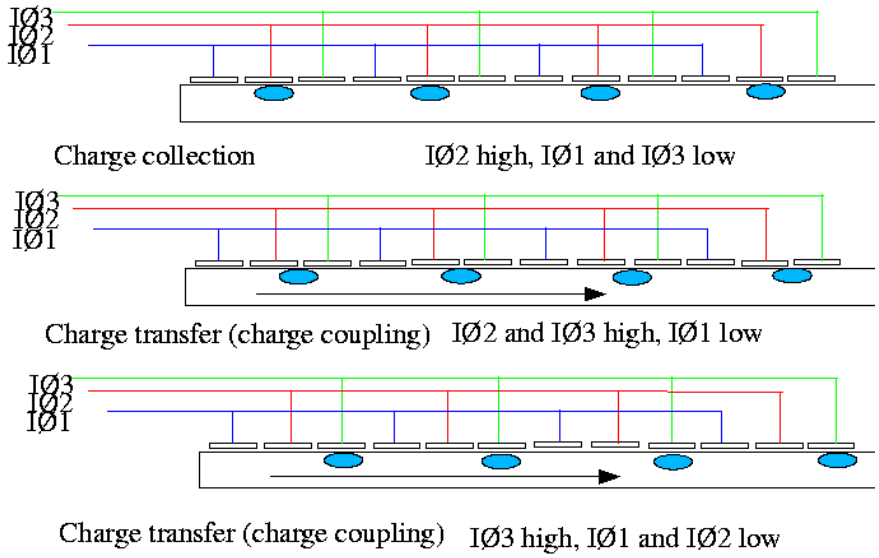


Figure 5.2: Charge transfer operation [13]

statistical errors. These can derive, for example, from an uncorrect exposure time. In fact a too low value can create problem of sensitivity because of no sufficient time for charge accumulation leading to insufficient Signal to Noise Ratio; on the other hand more time means more signal and the correlated photonic noise. Furthermore long exposure summed to the readout time can achieve obsolete measures.
Another danger is represented by cosmic rays and other energetic particles that affect the result of sensed electrons. They in fact can ionize the atoms and produce big amounts of new free energy. A remedy to minimize this effect is a more frequent readout.
In general the noise correlated to CCDs can be divided in two main fields:

- Noise on the image itself ("shot noise"): it is due to the fact that the detection of photons is a statistical process. If an image is acquired multiples times, the results will be slightly different and follow a Poisson probabilistic distribution. This means that the number of detected electrons is not the true value, but it must be considered to have an uncertainty of plus/minus the square root of itself;

- Thermally generated noise: it results from the internal process of CCD, for example the temperature of the circuit can generate secondary electrons and the so called *dark current*. This error has also a Poisson distribution.

Anyway, an overall observation of a CCD output presents the so called Fixed Pattern Noise. It is composed by a dark-signal non uniformity, i.e. with no external illumination, independent from the power applied and it manifests locally as an

offset from the average of the array. Then, another term that describes the gain between optical power and the electrical signal output , i.e. the photo response non uniformity, varying from pixel to pixel.
Both are corrected respectively with dark field and flat filed techniques. The first estimates the error measuring the response of the detector with no illumination (dark), while the second with uniform illumination (flat).
Furthermore, FPN is related, in addition to environmental conditions and exposure time, to the pixel size, the material and interference of the proximity circuit.
The ultimate source of noise of the CCD is determined by the readout noise. It is due to the on-chip amplifier which converts e^- in voltage level and it rises with the sampling frequency. It can also be expressed as a root mean square of the signal.
The technique, used to minimize it, is the Correlated Double Sampling. This consists in measuring twice the output and from the comparison of the two, estimating the error. In CCD case, the voltage at the time of reset (reference value) is subtracted to the one at the end of integration period.

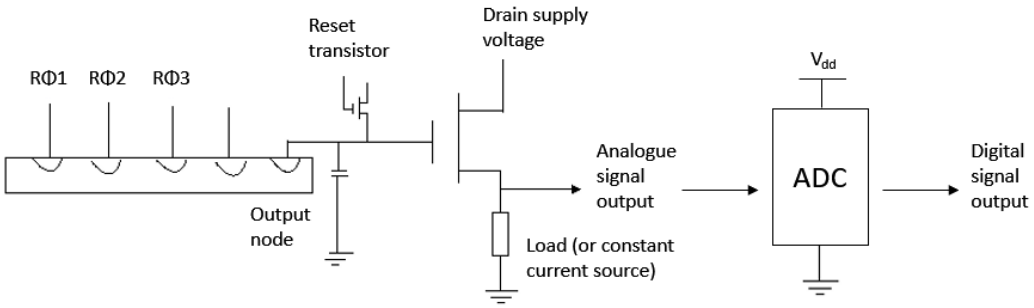


Figure 5.3: Typical CCD output circuit [13]

5.2 FGS performances

However, a star tracker or an attitude determination sensor, more in general, is affected not only by an high frequency noise, but also by other bias-type effects. According to ECSS standard [6], they are:

- Aberration of light due to the reciprocal motion between target and observer;
- Mechanical bias: on-ground calibration residual or launch-induced misalignments (vibration, depressurization, etc.), after-launch aging (Boresight Reference Frame respect to Measurement Reference Frame);

- FOV spatial error: Point Spread Function[1] variability across the FOV, residual of calibration of focal length and optical distorption, residual of aberration of light, CTE (with degradation caused by radiation), star catalogue error in position of targets;
- Pixel spatial error: detector non uniformity (FPN, straylight, star signal photonic noise), star centroiding error (rate influence);
- Thermo-elastic error: BRF – MRF stability due to optical head temperature or gradient caused by conductive and radiative effects.

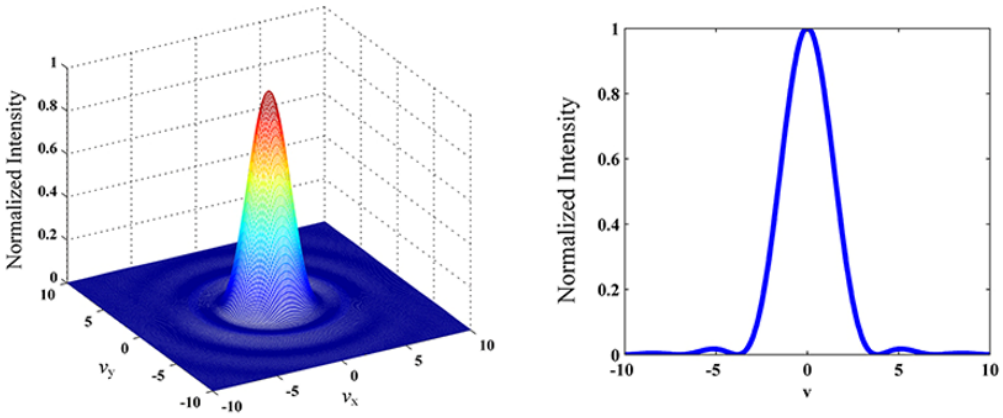


Figure 5.4: PSF typical profile [9]

Hence, FGS strengths regard the minimization of the mechanical and, particularly, of the thermoelastic bias. This is achieved integrating the sensor in the PLM and sharing the focal plane with VIS. Moreover the material is fundamental, since CCDs case, FPAs and VIS focal plane are all made in SiC.
In particular, for FGS, because of its CCD based architecture, noise is also considered. Overall contributors are:

- CCD readout noise;
- Electronic noise due to the operational amplifier (intrinsic of the circuit);
- Analogue to Digital Converter noise (intrinsic);
- Straylight noise due to disturbing sources (ex. Sun, bright stars);
- EMC noise due to the proximity with VIS (that is operated simultaneously).

	AME @99.7%	RME @99.7% (over 700 s)
x axis	0.6”	0.021”
y axis	0.6”	0.021”
z axis	8.7”	1.5”

Table 5.1: FGS performance requirements

It results in unprecedented measurement accuracies. These are obviously different for absolute and relative modes. Since they have different requirements resumed in Table 5.1.
More in details, Absolute Measurement Error is defined as the difference (in Euler angles) between measured and reference attitude at each time of derivation. Relative Measurement Error, on the other hand, is determined as the deviation from the mean value of the absolute measurements during a period of time.
Both the requirements are considered to be valid at 99.7%, i.e. three times the standard deviation (also called *root mean square* result). AME and RME are visually explained in figure 5.5.

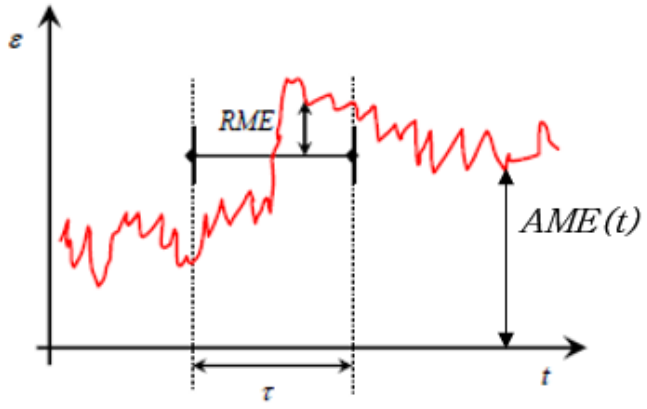


Figure 5.5: AME and RME definition [6]

To comply these stringent requests, performance budgets are needed, starting from the noise contributions measured in *electrons*, as shown in Table 5.2. To take some

[1]It is the function that approximates the distribution of illumination intensity across the focal plane. Its ideal profile is presented in figure 5.4.

Contributor	Noise rms [e^-]
CCD CDS	12.8
Electronics amplifier	12.5
ADC	24
Straylight	11
EMC	2.4
Overall (rss)	33.5

Table 5.2: FGS noise budget

margin, the budget rises to 40 e^- rms.
High-frequency errors can be then translated in the so called *Noise Equivalent Angle* ([mas]). On it, the exposure time plays a fundamental role. In case of dim stars, in fact, a bigger exposure time is requested. This effect was taken into consideration in modes timing definition.
Here shown the resulting probability of detection performed with a Montecarlo simulation that comprehends, in addition to NEA, also the errors introduced by RLE algorithm and CTE [2]:

Magnitude	Static conditions	Dynamic conditions (exposure time control)
<=16	100%	100%
17	99.9%	99.9%
18	99.9%	93.1%
19	92%	49%

Table 5.3: FGS expected probability of detection

For what concerns the low-frequency contributions, uncertaintes (listed in Table 5.4) in terms of [mm] or [as] are considered, knowing that a pixel side is equivalent to 0.1 as. Then, for CCD positioning and tilt, a distinction between self and relative (for a detector respect to the other) knowledge is done.

[2]It is possible that a dim target randomly disappears during tracking mode. For this reason a selection algorithm focuses on filtering only the 10 brightest stars, when it is possible.

Parameters	Max error in uncalibrated conditions
Focal length	20 mm
CCD tilt	143 as
CCD relative tilt	286 as
CCD positioning	0.1042 mm
CCD relative positioning	0.2084 mm

Table 5.4: FGS maximum expected low frequency errors

At this point, the final performance budget can be obtained for uncalibrated or calibrated conditions. For the first case, the results do not satisfy the requirements. In fact, as shown in Table 5.5, low frequency errors are relatively high for the absolute mode. This because FGS does not need to know its detectors position, respect to the center of FGS_{RF}, in RTM.

RME @99.7% (over 700 s)	X["]	Y["]	Z["]
HF error	0.0020	0.0040	0.2171
LF error	0.0001	0.0001	0.0078
Resultant	0.0021	0.0041	0.2249

AME @99.7%	X["]	Y["]	Z["]
HF error	0.0264	0.0507	2.6405
LF error	2.7127	1.2857	94.4142
Resultant	2.7391	1.3364	97.0547

Table 5.5: Overall performance budget in uncalibrated conditions

Uncalibrated is intended to be without in flight calibration. In fact, on ground, during integration of FGS inside PLM, a regulation is foreseen through a laser measurement based on the reflecting surfaces of the two focal plane assemblies.
Instead, calibration represents a problem when the reference frame is not physically defined and it is difficult to implement a practical method. However the following two are identified:

- Self-calibration: a CCD is taken as a reference while the other three refer

their position to it. In this way the low frequency errors are limited to the relative ones;

- Cross-calibration: FGS is calibrated respect to VIS, through the comparison between the sensor and instrument measurements.

Both and in sequence are chosen to recover from z axis' out of specifications in ATM, while in RTM the first is enough[3]. So, a full in-flight calibration is necessary for ATM, while RTM can be used as a sort of *backup* mode in case of failed absolute determinations. The obtained performances are listed below.

RME @99.7% (over 700 s)	X["]	Y["]	Z["]
HF	0.0020	0.0040	0.2171
LF	0.0001	0.0001	0.0001
Resultant	0.0021	0.0041	0.2172

Table 5.6: FGS expected RME with self-calibration

AME @99.7%	X["]	Y["]	Z["]
HF	0.0264	0.0507	2.6405
LF	0.4100	0.2100	11.1700
Resultant	0.4364	0.2607	13.8105

Table 5.7: FGS expected AME with self-calibration

AME @99.7%	X["]	Y["]	Z["]
HF	0.0264	0.0507	2.6405
LF	0.3711	0.0947	4.5331
Resultant	0.3975	0.1454	7.1736

Table 5.8: FGS expected AME with cross-calibration

[3]To recover means to add a numerical correction to the measurement.

Chapter 6

Purpose and scope of FGS simulator

In order to verify the compliance of the performances in different scenarios, an Engineering Qualification Model of the S/C is built. It performs SW and HW/SW tests, using an Electrical Stimuli Generator and the models of all the equipments to be stimulated. Typical cases are responses to specific commands or to forced mode transitions.
For what concerns FGS, important tests regard the ASW response to sample images. Therefore, it is fundamental to describe the process of image generation and software test.

6.1 Test generation

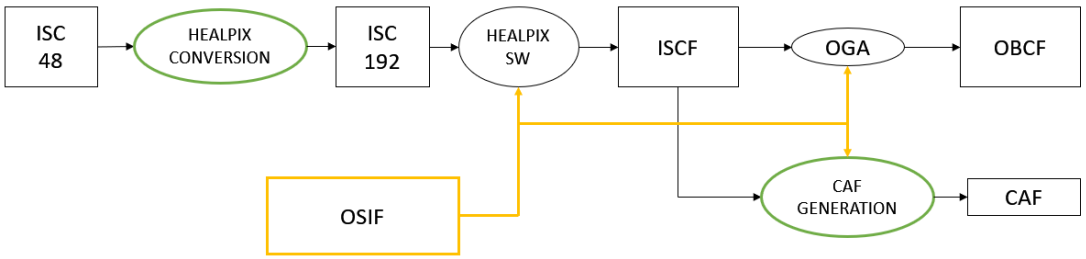


Figure 6.1: Test generation process

As shown in figure 6.1, different algorithms (rounded blocks) and files (squared blocks) precede the test.
The yellow diagram represents the support to all the process and it derives from

Euclid observation plan and the related AOCS dynamics simulations. It consists in an Observation Sequence Input File, i.e. a sequence of quaternions. Each one corresponding to the attitude of the S/C in a specific dither. If, for example, a full observation has to be tested, four quaternions are requested.
In green borders, instead, the algorithms elaborated with the present thesis are highlighted. These and all the black flow are explained in the next sections.

HEALPIX conversion

The Input Star Catalogue is the starting point. It is provided by the astronomic observatory of Turin and it consists of 48 ASCII files, one for each HEALPIX zone of the sky sphere.
This package of information would be too heavy for ground stations processing times during mission operations. These latter, in fact, foresee a periodical generation of the on board star catalogue.
So, it is necessary to subdivide the original 48 files into 192 smaller ones and eventually (only for operations phase and not for on ground tests) to translate them in binary format.

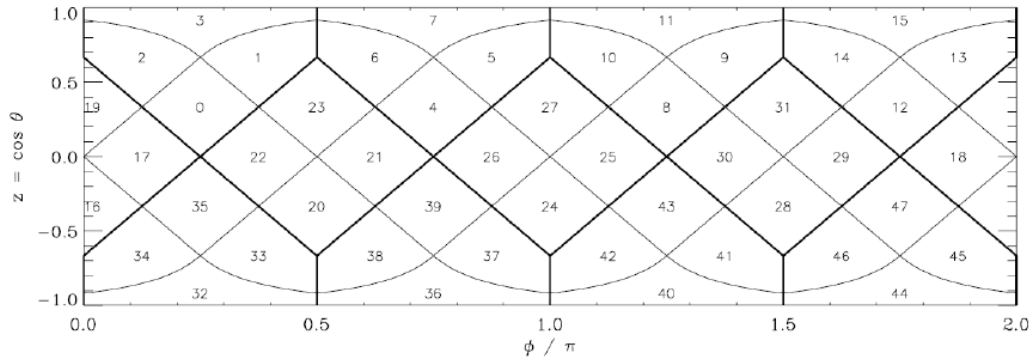


Figure 6.2: Cylindrical projection of the Healpix division of the sphere using nested scheme with $N_{side} = 2$ [7]

It is done considering that the base resolution for HEALPIX tessellation is constituted by $N_\theta = 3$ pixel layers between north and south pole of the sphere and by $N_\phi = 4$ equatorial pixels. In this standard scheme the number of pixels would be:

$$N_{\text{pix}} = 12N_{\text{side}}^2 \tag{6.1}$$

where $N_{side} = 1$ represents the resolution of the grid and it is the number of divisions along the side of a base-resolution pixel. So, for the 48 case $N_{side} = 2$ while for the

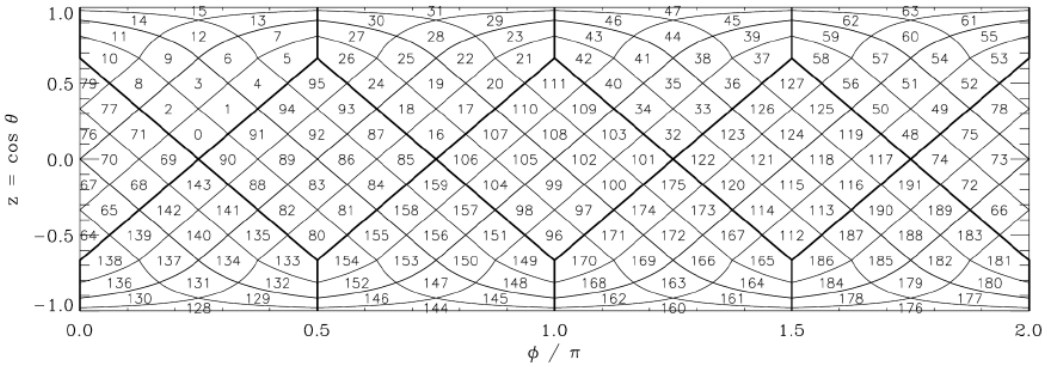


Figure 6.3: Cylindrical projection of the Healpix division of the sphere using nested scheme with $N_{side} = 4$ [7]

192 case $N_{side} = 4$.
Furthermore, another expression introduces the parameter *level*:

$$N_{side} = 2^{level} \tag{6.2}$$

This is equal to 0 for the base resolution, 1 for 48 pixels and 2 for 192 pixels.
The importance of this parameter is that there is a relation between the HEALPIX index, the target identifier and *level* [8]:

$$\text{Healpix\_ID} = \frac{\text{source\_ID}}{2^{35}4^{12-level}} \tag{6.3}$$

This derives from the scheme of enumeration, that for our case is *nested*, meaning that "the pixels are arranged in twelve tree structures and each one is organised according to quadrilateral tree pixel numbering" [8], as shown in figures 6.2 and 6.3.
"HEALPIX_conversion.c" is the main script in C language that accomplish these tasks. It is reported in attachment A.
ISC format is the *comma separated values* in which every text row contains the following arguments related to a star/target:

- Source ID;
- Right ascension α [degree];
- Declination δ [degree];
- Error on right ascension [as];
- Error on declination [as];

- Proper motion in right ascension [Mas/yr];
- Proper motion in declination [Mas/yr];
- Error on proper motion in right ascension [Mas/yr];
- Error on proper motion in declination [Mas/yr];
- Magnitude;
- Error on magnitude;
- Classification flag (0/1 as star/non-star);
- Neighbour[1] conditions flag;
- Magnitude variability flag.

Accordingly to ICRF, star's spherical coordinates are expressed as a pair, right ascension and declination, without a distance coordinate. HEALPIX uses, instead, another convention, in which $\delta \in [-\frac{\pi}{2}; \frac{\pi}{2}]$ is substituted by $\theta \in [0; \pi]$, i.e. co-latitude of the sphere, zero at the north pole and π at the south one (see figure 6.4).
So, after reading, row by row, the original ISC files, it is necessary to make the following translation:

$$\theta = -\delta + \frac{\pi}{2} \tag{6.4}$$

Then, it is possible to compute the new HEALPIX ID, i.e. to determine the output file of each target/row (from 1 to 192)[2].

CAF generation

To pass from ISC to the so-called Input Star Catalogue File, the HEALPIX software is used. It is designed to define the area of interest, meaning to select the targets of the specific observation.
Then, the process is splitted because a branch refers to the EQM simulation and another to the one by exploiting the FGS simulator (figure 6.9). This last uses the On Ground Algorithm that writes the On Board Catalogue File (format explained

[1]It means that in the same window of observation two sources can be present. It is a useful information, since in these cases the fainter star is renamed as neighbour and it is discarded or not by FGS software, depending on the stars density in the interested sky region. For a successful attitude determination, in fact, at least 3 stars are needed while it is decided that maximum 10 constitute the input set for the q-method.

[2]It is worth noting that information does not change between input and output, furthermore if the ASCII/binary conversion is not requested, neither the format.

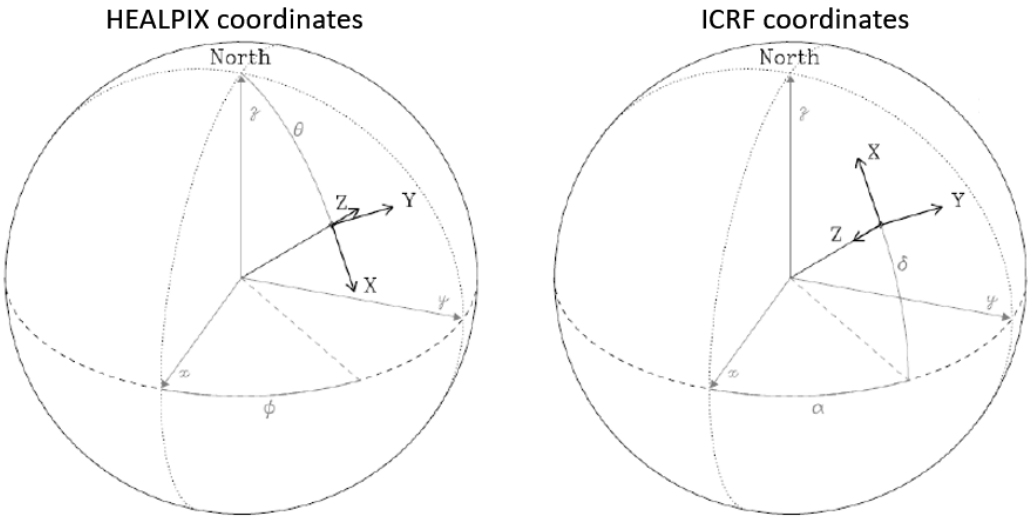


Figure 6.4: HEALPIX and ICRF convention for spherical coordinates [8]

in next chapter).
As regards the other line, multiple files have to be generated. In particular the so called Catalogue Auxiliary File.
It is a matrix containing $N \cdot 5$ elements, where $N \leq 100000$ is the number of stars belonging to a single observation (four dithers) and falling inside FGS FOV. The format is the *space separated values* and it is the following:

```
0.65522646045806  0.11204203832182 0.74707755097064
0.05

0.64743825076634  0.10204717781122 0.75525498008645 17.9407 0
0.64745200599802  0.10167136876323 0.75529387174990 12.3619 0
0.64746155507734  0.10176417220199 0.75527318763007 15.6843 0
0.64749930660934  0.10167960492300 0.75525221342484 18.2261 0
0.64728855690881  0.10180566689222 0.75541586578754 16.9586 0
```

Figure 6.5: CAF layout

In the first row the reference attitude is reported, i.e. the unitary vector corresponding to the initial quaternion. The second, instead, is a validity range of the area around the reference attitude.
Then, each row of the matrix contains the three direction cosines of a star in inertial frame, followed by its magnitude and a flag that refers to the type of object (star, large object, neighbour, etc.).

The format is the same of ISCF, that however is a $N \cdot 14$ type. Instead, the necessary inputs are the right ascension (α), the declination (δ), the magnitude, the flag and the initial quaternion of the observation. This is obtained from Euclid observation plan and provided in the same format.
First of all, the last input needs to be translated in (α_0, δ_0), passing from attitude matrix (from inertial to body) R. The latter is explicited using Rodriguez formula (4.13):

$$R = \begin{bmatrix} q_r^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y + 2q_zq_r & 2q_xq_z - 2q_yq_r \\ 2q_xq_y - 2q_zq_r & q_r^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z + 2q_xq_r \\ 2q_xq_z + 2q_yq_r & 2q_yq_z - 2q_xq_r & q_r^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{6.5}$$

Consequently reference normalized vector is computed. It represents the inertial vector v_{IRF} that in body coordinates matches the boresight vector $v_{MRF} = [0\,0\,-1]$:

$$\begin{aligned} v'_{IRF} &= R^T v_{MRF} \\ v_{IRF} &= \frac{v'_{IRF}}{|v'_{IRF}|} \end{aligned} \tag{6.6}$$

Then, another useful translation regards ISCF and it is the one from (α, δ) to vector components:

$$\vec{v} = \begin{pmatrix} \cos\alpha\cos\delta \\ \sin\alpha\cos\delta \\ \sin\delta \end{pmatrix} \tag{6.7}$$

Since ISCF contains a lot more stars than those falling inside FGS FOV, a selection algorithm is needed.
Its first step is the projection from ICRF to FGS_{RF} of the input vectors. So, it is useful to calculate the attitude matrix T from the quaternion of the current dither $q_i = [q_x, q_y, q_z, q_r]$ still using (4.13).
From this point, it is immediate to compute the coordinates in body frame:

$$\vec{v_b} = T\,\vec{v} \tag{6.8}$$

Then, to compute them in the focal plane, the focal length $f = 24500\,\text{mm}$ is considered and geometrically it follows:

$$\begin{aligned} x_{\text{FGS}} &= \frac{v_{b_x}}{v_{b_z}} f \\ y_{\text{FGS}} &= \frac{v_{b_y}}{v_{b_z}} f \end{aligned} \tag{6.9}$$

An example of the result is in figure 6.7.
As it is shown in figure 6.6, instead, CCDs have different orientation of the frames, due to their mounting on the structure. So a mounting matrix A_d of each CCD respect to FGS_{RF} is computed. Re-indexing[3] detectors as $R_1 = 1$, $R_2 = 2$, $L_1 = 3$

[3]These indexes remain the same for the rest of the thesis, appendices included.

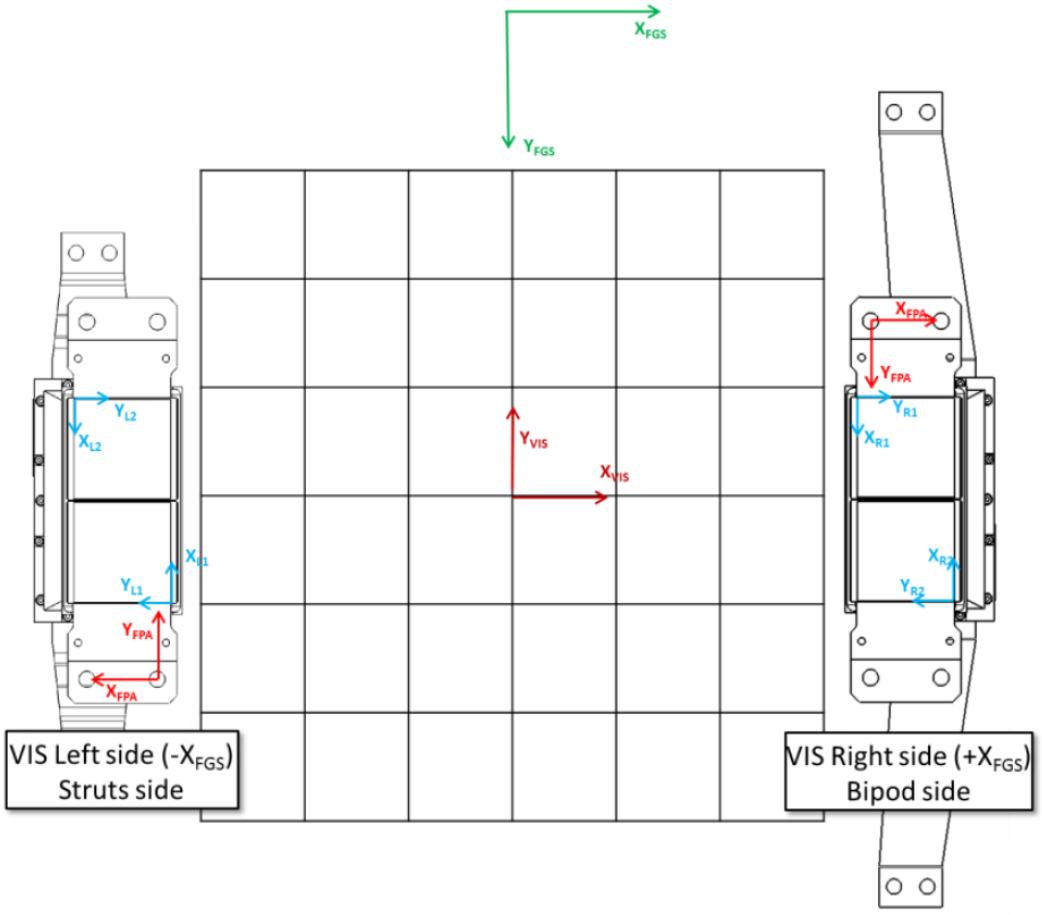


Figure 6.6: FGS_{RF} (green) and DET_{RF} (blue) [Courtesy of Thales Alenia Space]

and $L_2 = 4$:

$$
\begin{aligned}
A_1 = A_4 &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\
A_2 = A_3 &= \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \\
\begin{pmatrix} x_{\mathrm{DET}_d} \\ y_{\mathrm{DET}_d} \end{pmatrix} &= A_d^T \begin{pmatrix} x_{\mathrm{FGS}} \\ y_{\mathrm{FGS}} \end{pmatrix}
\end{aligned}
\tag{6.10}
$$

The following step is the selection of the targets that can be seen by each detector surface. This is considered to be 4096 x 4096 pixels, with each squared pixel with sides of 0.1 as. At the first dither the same area is considered 90 as bigger and at the other exposures only 5 as bigger. These to take account of pointing uncertainties. Finally the resulting targets are saved in order to have a CAF file per observation.

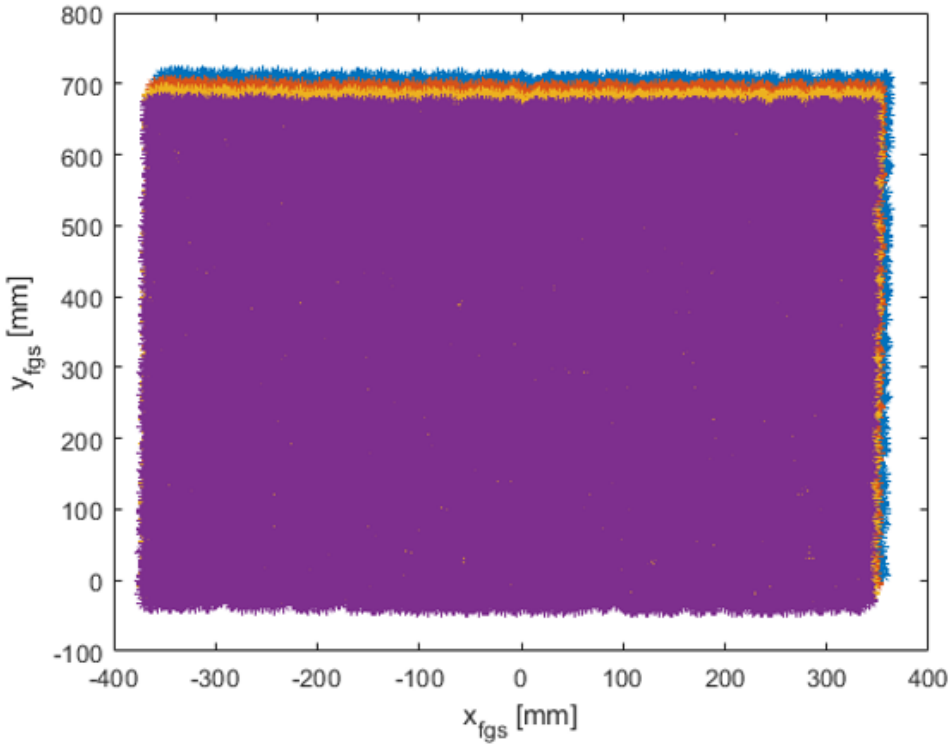


Figure 6.7: Example of star projection in FGS_{RF} for 4 dithers

"CAFgeneration.m" is the main Matlab script written to generate the above described output and it is reported in attachment B.

6.2 Test implementation and simulator role

As already said, the tests made on EQM are supported by the simulator (figure 6.9). This procedure allows to, in case of not expected results from test, identify the point of failure (or just warning) in an easier way.
In fact, as better explained in the next chapter, the simulator includes several algorithms and it explicits the way they are called in sequence. With this procedure, line by line debugging helps the identification of the problem.
Furthermore, also in case of successful tests, it is possible to verify in a fast way the robustness of the results obtained with ESG stimulation or just to predict real FGS behaviour under specific scenarios:

- NEA and catalogue uncertaintes;
- bias on focal length;

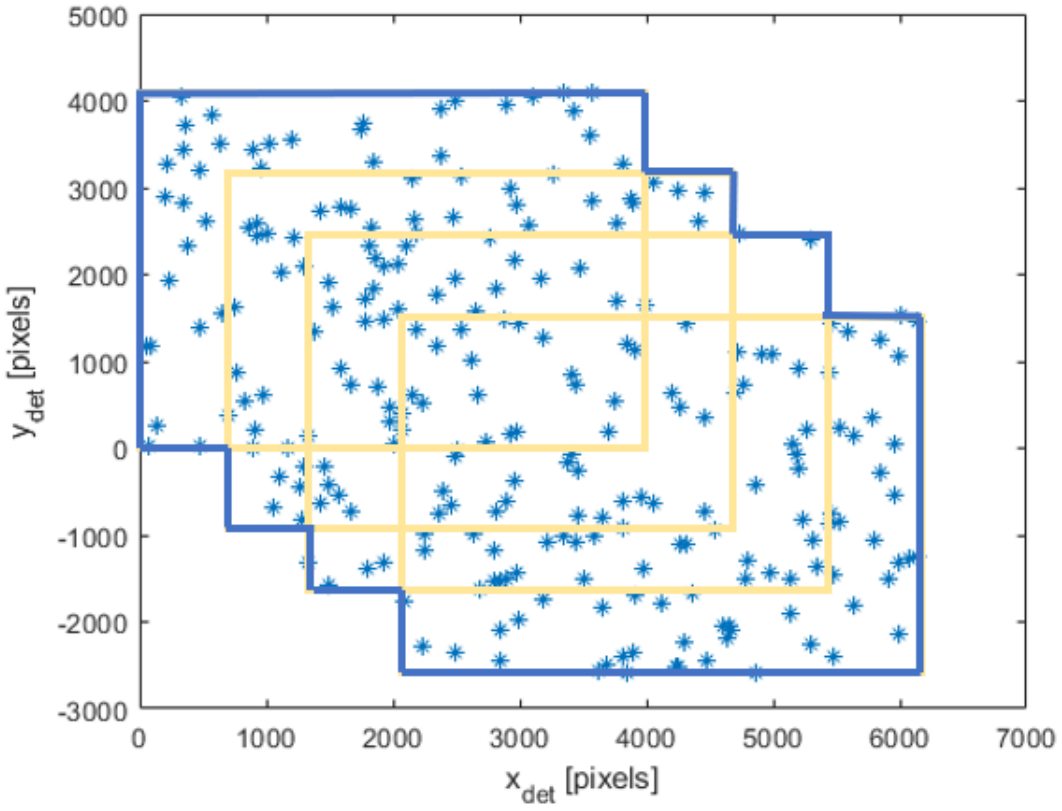


Figure 6.8: Example of star selection in a single DET_{RF} for 4 dithers

- bias on CCDs position in focal plane;
- bias in terms of tilt of detectors;
- optical distortion;
- temporary disappearance of a star.

FGS simulator is designed for RTM and ATM modes. Furthermore it simulates algorithms working during a whole dither observation. It means that more phases and FGS cycles are accounted for.
More in details 2 AP and a TP are simulated. IC, in fact, is not particularly distinguished from AP, since the simulator is based on a projection from the star catalogue and it does not rely on an external image.
It is worth to specify that RTM, from the simulator point of view, is functionally equivalent to ATM with acquisition commanded in Coarse Mode. This because the propagation of the attitude to the next cycle is not implemented. Consequently the restricted windows on CCDs are not predicted. Stars on detectors derive, instead, from the input catalogue and interest all the pixels at each cycle, like a full frame

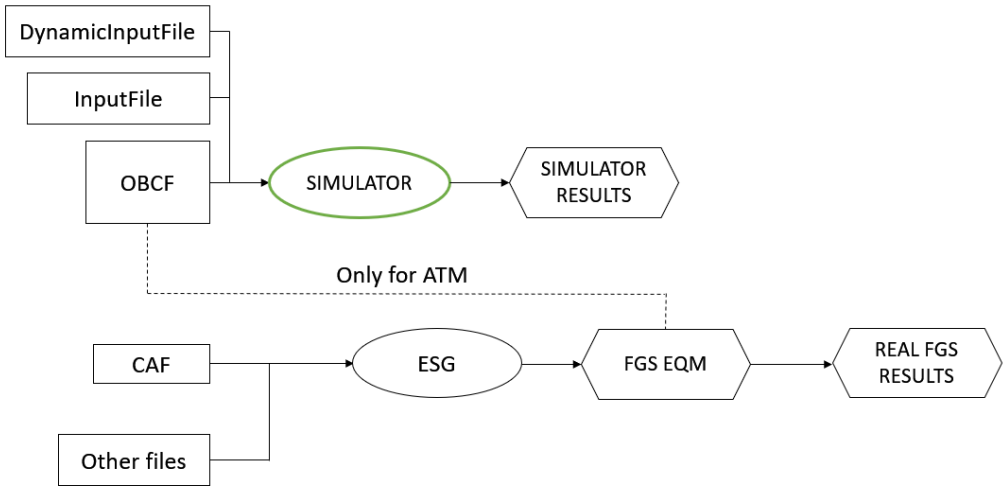


Figure 6.9: Parallel test concept

acquisition.
For what concerns ESG, it receives CAF and others files, so it is enroled of generating the image on the detectors. It takes the role of the real sky.
An important consideration regards the frequency of operations. It differs between ESG and simulator: the first monitors the response of real FGS at 1 Hz, while the second receives dynamic inputs (sequence of quaternions and angular rates) discretized at 0.5 Hz, that is the period of the FGS tracking cycle.
In other files, here not specified, there are information regarding telemetries and telecommands, as well as the times when they are executed. For convention, ESG and simulator are considered to be synchronized. In such a way, the simulator can refer to the same history of commands.
Taking as reference the dynamic results of AOCS, it is possible to extract a plot of the angular rate over time for a single observation (figure 6.11). From there, it is possible to understand when FGS is commanded to RTM (or ATM)[4].
Then, simulator replicates the three phases: executing 1 AP at the time of the command to tracking mode, another one after 6 seconds (when IC would start) and multiple tracking cycles, starting 4 seconds later (IC period) and every 2 seconds until the next command takes place, that is usually the transition to SBM. This last is planned to be before a slew starts. During it, in fact, star tracker is the enroled sensor by AOCS.
After the simulation, the objective is to evaluate simulator and real FGS responses.

[4]For RTM there is no difference between fine or coarse.

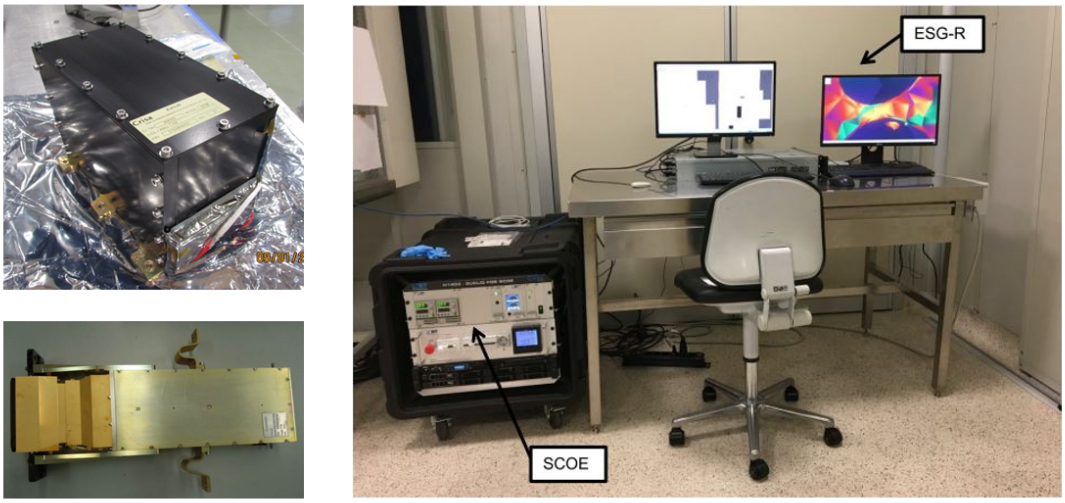
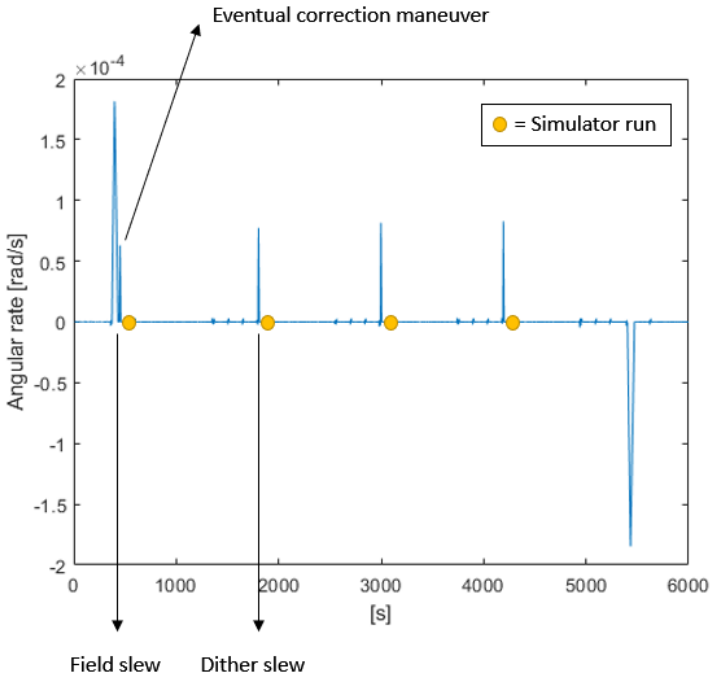
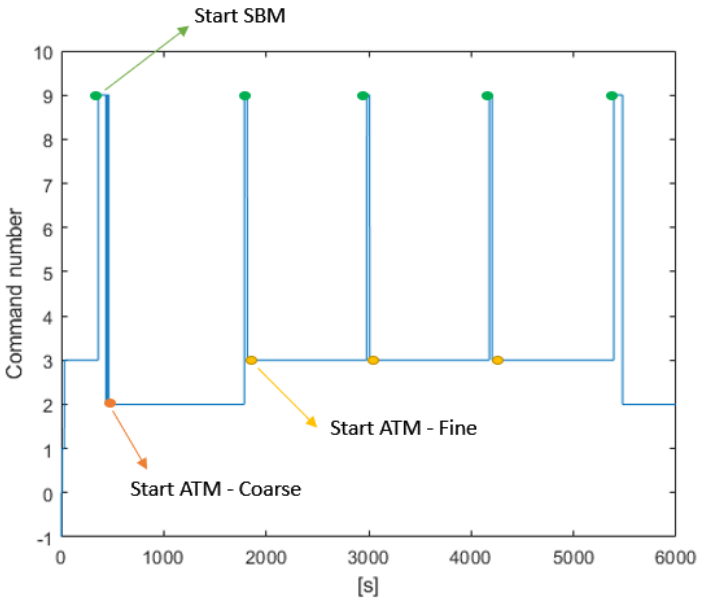


Figure 6.10: Test configuration: EU (upper left), FGS assembly (lower left), ESG (right) [Courtesy of Thales Alenia Space]

It means that AME and RME must be compliant to requirements and error budget listed in chapter 5 and to be comparable between both the parallel branches of test. Results of simulator for a whole observation and a reference pointing are reported and commented on in section 7.3.



(a)



(b)

Figure 6.11: History of a typical observation

Chapter 7

Modelling and simulation

The model of FGS is here explained, including its inputs and outputs, in addition to the main script. Furthermore, the results obtained through the simulation of specific scenarios are reported and compared to those foreseen by error budget.
Simulator main script is reported in appendix C, while auxiliary functions are only described from a theoretical point of view.

7.1 Input files

The simulator receives multiple files as input:

- OBCF;
- DynamicInputFile;
- InputFile.

So, it is worth describing them in the order explicited above.

OBCF

As already anticipated, OBCF is the result of the On Ground Algorithm. It includes useful data to link the targets to the single detector, at the interested observation.
For each CCD, in fact, information regards triads and guide stars. Only these latter are important from simulator's point of view, since, in it, pattern recognition is already implicitly verified. This because guide stars are projected from the inertial frame (coordinates provided by OBCF) to the body one and the same are considered to be detected by FGS. The real FGS, instead, detects the stars from the real sky (or an external source like ESG) and then it recognizes the pattern.
OBCF format is the *comma separated values*. In particular, targets information is

organized in $N \cdot 9$ matrices, where N is the number of targets detectable by a single CCD. So, four matrices need to be read.
Each row has different integer arguments, to be multiplied by a factor to obtain the value in international system of units or the true dimensionless value (table 7.1).

Parameter	Multiplying factor
index of CCD (from 1 to 4)[1]	1
index for targets inside the single CCD[2]	1
star magnitude	1
inertial coordinate u_x	$1/10^9$ [m]
inertial coordinate u_y	$1/10^9$ [m]
inertial coordinate u_z	$1/10^9$ [m]
right ascension accuracy σ_α	$1/10^{12}$ [mrad]
declination accuracy σ_δ	$1/10^{12}$ [mrad]
auxiliary flag (1 by default)	1

Table 7.1: OBCF row structure

DynamicInputFile

DynamicInputFile provides the expected S/C attitude evolution in time, during an observation period. It is in *space separated values* and it is constituted by a matrix $N \cdot 8$, where N is the number of dynamic inputs (provided at 1 Hz) and along each row there are:

- time [s];
- the 4 components of the expected quaternion;
- the 3 foreseen angular rate components [rad/s].

This text file is fundamental to synchronize simulator and real FGS. To simplify the problem, no time gap is considered between ESG command and FGS execution.

[1] See section 6.1 for reference.

[2] It is not the identification number of Gaia, but an index from 1 to N number of total targets in the CCD.

InputFile

InputFile is a ".m" file that contains a list of parameters to be initialized to distinguish different test scenarios, also including the names of the other two input files and of the output folder.
Some parameters can be grouped, since they regard the geometry of FGS. In particular:

- the focal length $f = 24500$ [mm] that depends on telescope design;
- the offset distance ($x_{0_{det}}$, $y_{0_{det}}$ [mm]) of the detectors respect the center of FGS_{RF} (see figure 6.6):

$$x_{0_{det}} = [206\ 256\ -206\ -256]$$
$$y_{0_{det}} = [303.931\ 413.931\ 413.931\ 303.931]$$

- the polynomial coefficients for optical distortion (α, β), also depending on telescope design, the same for all the targets[3]:

$$x'_{\text{fgs}} = -\alpha_0 + \alpha_1 x_{\text{fgs}} + \alpha_2 y_{\text{fgs}} + \alpha_3 x_{\text{fgs}}(x^2_{\text{fgs}} + y^2_{\text{fgs}}) + \alpha_4 x_{\text{fgs}}(x^2_{\text{fgs}} + y^2_{\text{fgs}})^2 - \alpha_5 x^2_{\text{fgs}} - \alpha_6 x_{\text{fgs}} y_{\text{fgs}} - \alpha_7 y^2_{\text{fgs}}$$

$$y'_{\text{fgs}} = -\beta_0 + \beta_1 y_{\text{fgs}} + \beta_2 x_{\text{fgs}} + \beta_3 y_{\text{fgs}}(x^2_{\text{fgs}} + y^2_{\text{fgs}}) + \beta_4 y_{\text{fgs}}(x^2_{\text{fgs}} + y^2_{\text{fgs}})^2 - \beta_5 y^2_{\text{fgs}} - \beta_6 x_{\text{fgs}} y_{\text{fgs}} - \beta_7 x^2_{\text{fgs}}$$

- the axis orientation matrix of each CCD, already seen in chapter 6:

$$A_1 = A_4 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$
$$A_2 = A_3 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

- the mounting matrix M of each CCD respect to FGS, function of the *tilt* angle τ (0 [rad] by default) that is the rotation of the detector in the focal plane, respect to the designed position (expressed by A):

$$M = \begin{bmatrix} \cos\tau_{CCD} & \sin\tau_{CCD} \\ -\sin\tau_{CCD} & \cos\tau_{CCD} \end{bmatrix}$$

[3]They are taken as [0 1 0 0 0 0 0 0] by default, i.e. no distortion.

All these can be modified to introduce the bias-type errors described in chapter 5. They can mainly represent a mounting error, a launch vibrations' induced error or a thermoelastic deformation. This latter is minimized by the design of FGS.
Other parameters strictly depend on the design of the sensor or the ASW and can not be modified:

- the pixel side dimension PIXsize = 12 [μm];
- the margins from external borders (*border* = 100 [pixels]) and internal cross section (*qborder* = 13 [pixels]) of the CCD, to be taken into account when selecting the targets to track;
- the number of maximum trackable targets Ntrack_max = 20;
- the minimum NSmin = 3 and maximum NSmax = 10 number of targets on which the q-method is applied;
- the minimum Texp_min = 0.1 [s] and maximum Texp_max = 1.6 [s] exposure time;
- the minimum Texp_step = 0.01 [s] step of discretization of the exposure time;
- the reference star signal S0 = $1.8{\cdot}10^{10}$ [e^-/s];
- the ratio of reference signal Speak = 0.5 on the peak pixel (with highest e^- value);
- saturation threshold for a pixel THsat = $19{\cdot}10^4$ [e^-];
- maximum displacement of the image on FGS focal plane DeltaSmax = 0.3 [as] that limits the exposure time[4];
- period of FGS tracking cycle Delta_time = 2 [s].

Instead, other parameters are necessary to include a detection probability:

- vector abscissa = [-5.000 -4.999 -4.998 ... 4.998 4.999 5.000] of a Probability Density Function;
- magnitudes mi = [16 17 18 19] defining performances;
- probabilities det_p0_ap = [100 100 100 78] of detecting targets with mi magnitudes in AP, at 1.5 s of exposure and 0 "/s of angular rate;

[4]If the displacement is bigger, the image risks to exit the FGS window FOV in the next tracking cycle.

- probabilities det_p1_ap = [100 99.9 98.9 77.2] of detecting targets with mi magnitudes in AP, at 1.5 s of exposure and 0.3 ”/s of angular rate;
- probabilities det_p0_tp = [100 99.9 99.7 82] of detecting targets with mi magnitudes in TP, at 1.3 s of exposure and 0 ”/s of angular rate;
- probabilities det_p1_tp = [100 99.9 90.3 58] of detecting targets with mi magnitudes in TP, at 1.3 s of exposure and 0.3 ”/s of angular rate;
- reference angular rates ref_rate = [0 0.3] [”/s] for the interpolation of p0 and p1 probabilities.

In order to simulate the effect of NEA and catalogue accuracy, in fact, the Cumulative Distribution Function is used. It gives the area under the PDF from minus infinity to x (figure 7.1). It means that, to each x of CDF, a probability corresponds.

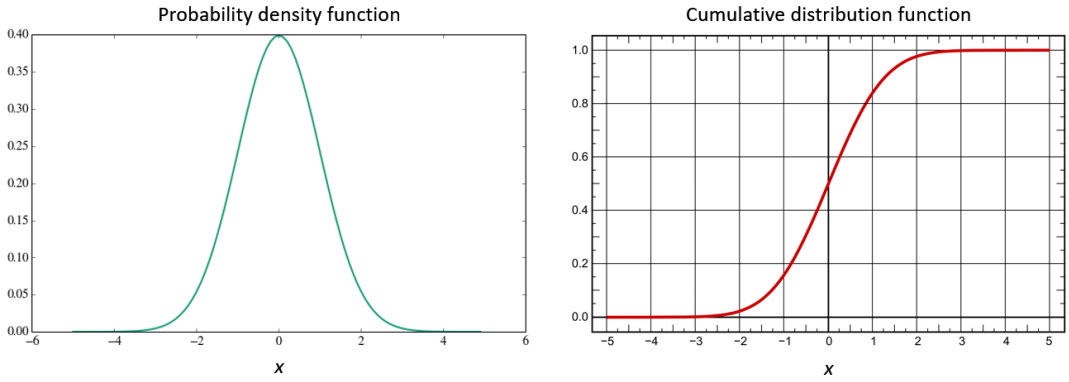


Figure 7.1: Functions for detection probability

So, since noise follows quite well a Poisson distribution, CDF is analyzed in order to determine its abscissa values that discretize a probability value (from 0 to 100) in 1000 elements. Then, since the Matlab function *randn* returns a normally distributed number with mean 0, if its result is lower then the x corresponding to a specific detection probability *det_p*, it will be a case happening *det_p* times out of 100.
Furthermore, to consider the effect of angular rate on detection, *det_p* is previously linearly interpolated on ref_rate. Hence, it is re-evaluated at each cycle.
The algorithms called in sequence are "linear_interpolation.m" and "detection.m".
Lastly, parameters regarding configuration of the test are listed:

- string "command" corresponding to "atm" or "rtm", depending on whether RME is computed on absolute or relative quaternions;

- number Nstart corresponding to the row of DynamicInputFile from which RTM/ATM starts;
- number Nend corresponding to the row of DynamicInputFile when RTM/ATM stops;
- vector "CCD" including active detectors (one at all or one per PEM);
- three-element vector "disappear" containing: ID of the star forced to disappear, starting cycle of the event, ending cycle of the same.

7.2 Main script

As already anticipated in section 6.2, simulator applies to ATM with AP-coarse mode or to RTM. Therefore, it is necessary to simulate two acquisition phases and a tracking one.
The first one will be run at the first available dynamic input after the command to the tracking mode. The second one, instead, simulates IC and it is performed 6 seconds after first AP. So, TP starts 4 seconds after and it is constituted by as many cycles as those foreseen by DynamicInputFile until 2 seconds miss from SBM.
Each phase is repeated for any active CCD as shown in the main script, called "Main.m".

Acquisition phase

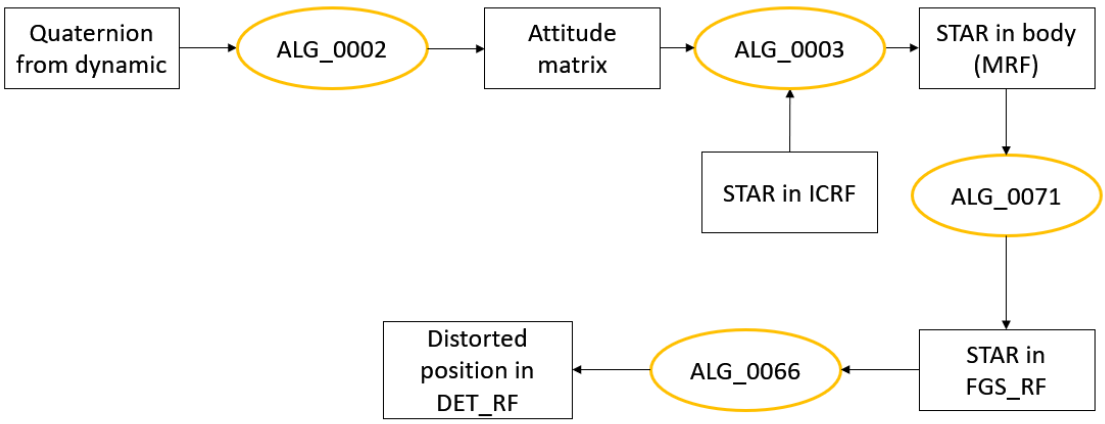


Figure 7.2: Star catalogue projection process - ALG_0220

After reading the three input files, a check is performed to verify that the catalogue contains stars. If so, the catalogue star projection takes place ("ALG_0220.m").

The process is the same of the CAF generation (resumed in figure 7.2, with reference to internal functions), except for the projection from FGS_{RF} to DET_{RF} that includes also the mounting matrix M:

$$\begin{bmatrix} x_{det} \\ y_{det} \end{bmatrix} = MA^T \begin{bmatrix} x'_{fgs} - x_{0_{det}} \\ y'_{fgs} - y_{0_{det}} \end{bmatrix} \tag{7.1}$$

So, the resulting coordinates take into account the optical distortion, the axis orientation matrix and the mounting matrix (all in "ALG_0066.m"). It can be, therefore, sensible to the error induced by telescope lenses and to the bias between VIS and FGS. The first depends on the geometry of the lenses and position of the targets inside focal plane; the second depends on thermoelastic or simply mechanical deformation during the entire life of the S/C.

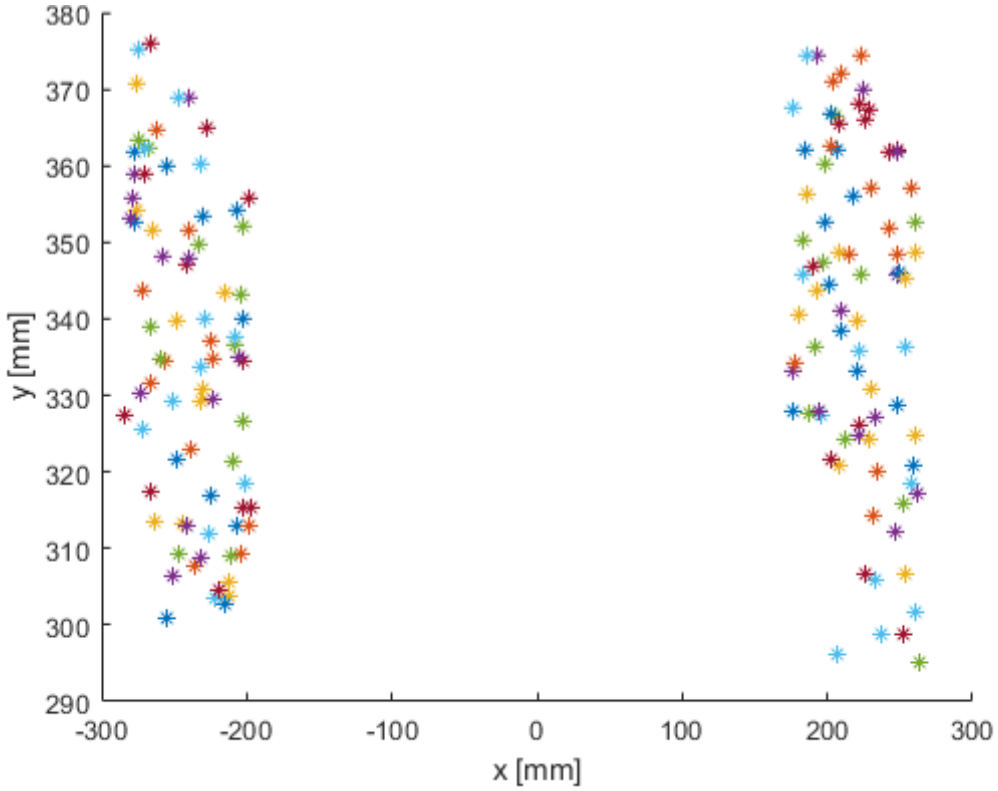


Figure 7.3: Example of star catalogue projection in FGS_RF

Until this point, the path of the photons towards CCDs is simulated. Now the ability in detecting such sources needs to be implemented.
Two main contributors to high frequency errors are chosen to be part of the model. They are the so called NEA and the catalogue accuracy. The first includes all the CCD related noise, as explained in chapter 5. It is therefore a random noise and

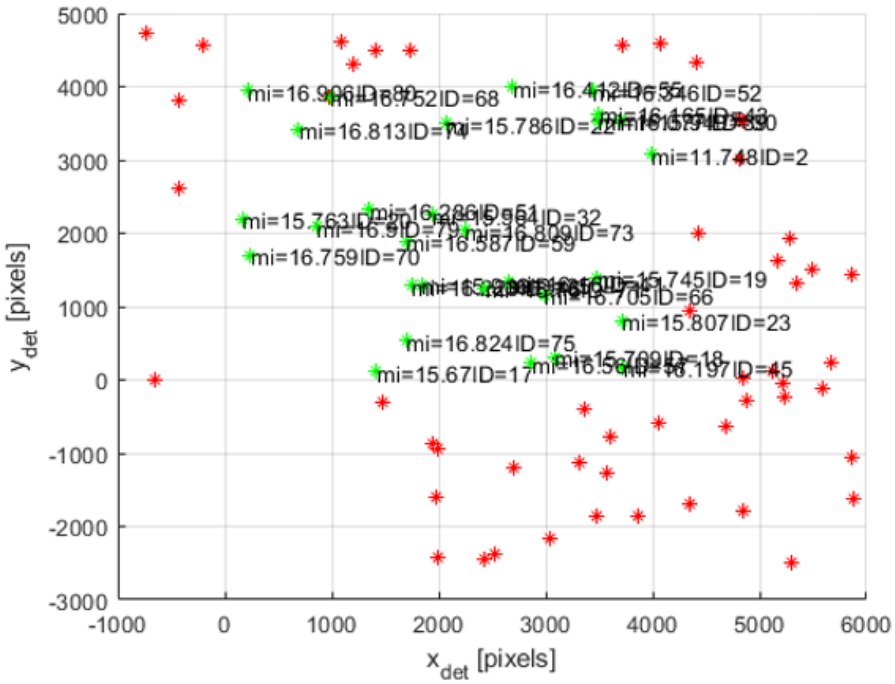


Figure 7.4: Example of perturbated and detected stars (green) in DET_RF

dependent from the star magnitude. The following polynomial expression is used to compute the variance of NEA:

$$\sigma_{NEA} = \eta_0 + \eta_1 m_i + \eta_2 m_i^2 + \eta_3 m_i^3 + \eta_4 m_i^4 + \eta_5 m_i^5 + \eta_6 m_i^6 \tag{7.2}$$

The accuracy of the catalogue is provided in terms of ascension-declination (α, δ) for each star. So, the variance of the same is:

$$\sigma_{cat} = \sqrt{\sigma_\alpha^2 + \sigma_\delta^2}\,\frac{180}{\pi}\,3600 \tag{7.3}$$

translating the result from [mrad] to [mas].
At this point, referring to what already said about the detection probability simulation, for each star a random number r is generated and if it is smaller than det_p, the star is considered detected and its coordinates modified as:

$$\begin{aligned} x'_{det} &= x_{det} + (r\sigma_{cat} + r\sigma_{NEA})0.01 \\ y'_{det} &= y_{det} + (r\sigma_{cat} + r\sigma_{NEA})0.01 \end{aligned} \tag{7.4}$$

where 0.01 is the conversion factor from [mas] to [pixels].
In this way, a set of targets is obtained without taking into account the FOV of the

detectors. These are 4096 x 4096 pixels and, having just computed the positions in terms of pixels, it is immediate to verify:

$$\begin{aligned} 0 < x'_{det} < 4096 \\ 0 < y'_{det} < 4096 \end{aligned} \tag{7.5}$$

Then, another check is performed, to verify that at least one star is still detected. If it is so, the objective is to report the results to the MRF. This because the attitude is determined starting from the new body vectors (measurement) and the original inertial vectors (known reference).
Inverting equation 7.1, it can be written:

$$\begin{bmatrix} x_{fgs_m} \\ y_{fgs_m} \end{bmatrix} = A\, M^T \begin{bmatrix} x'_{det} \\ y'_{det} \end{bmatrix} + \begin{bmatrix} x_{0_{det_b}} \\ y_{0_{det_b}} \end{bmatrix} \tag{7.6}$$

with x'_{det}, y'_{det} in [mm] and the eventual introduction of an offset bias $x_{0_{det_b}}$, $y_{0_{det_b}}$.
Instead, to pass from FGS_RF to MRF_RF, equation 6.9 is inverted:

$$\begin{Bmatrix} u_{x_{MRF_m}} \\ u_{y_{MRF_m}} \\ u_{z_{MRF_m}} \end{Bmatrix} = \begin{Bmatrix} -x_{fgs_m}/R \\ -y_{fgs_m}/R \\ -f_b/R \end{Bmatrix} \tag{7.7}$$

with the normalization distance $R = \sqrt{x^2_{fgs_m} + y^2_{fgs_m} + f_b^2}$. In fact, it is in this transformation that the bias can play a role, changing the focal length.
Before passing to the tracking phase, some selections need to be performed. The first regards the exclusion of the stars falling near external borders or inside the quadrants intersection. There, in fact, the charge transfer can heavily perturb the results and the eventual window can not belong to two different quadrants.
More in details, the internal border check is always done, while the external one only in case of more than Ntrack_max targets are detected. In fact, margin from external border is taken to minimize the risk of an exit of a star from the FOV in next cycles. Furthermore, it is not done in tracking phase.
After another check to understand if some stars are still present in the reduced FOV, the attitude can be computed through q-method. Following the steps described in section 4.2, the weight of each i target is defined and so the resulting *weighted matrix* W, for N_m number of measured targets:

$$\begin{aligned} w_i &= \frac{1}{N_m}, \ \forall i \\ W &= \sum_{i=1}^{N_m} w_i \begin{Bmatrix} u_{x_{MRF_{m_i}}} \\ u_{y_{MRF_{m_i}}} \\ u_{z_{MRF_{m_i}}} \end{Bmatrix} \begin{Bmatrix} u_{x_i} & u_{y_i} & u_{z_i} \end{Bmatrix} \end{aligned} \tag{7.8}$$

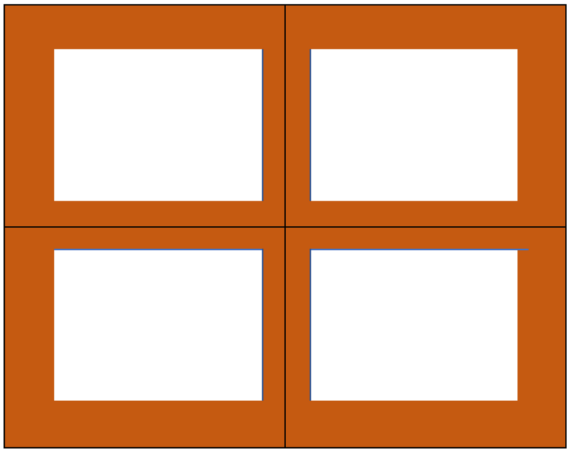


Figure 7.5: Border check exclusion areas (not drawn to scale)

So, the matrix Q is built, according to equation 4.15.
Actually, the function for attitude determination ("ALG_0010.m") works on Q^{-1}, since the eigenvalues/eigenvectors are the same[5]. Hence the quaternion in acquisition phase q_{ap} is found, looking for the eigenvector corresponding to the maximum eigenvalue.
Now the exposure time for next tracking cycles is computed. It can be demonstrated that optimal exposure rises exponentially with magnitude. In fact, in nominal conditions it is computed as:

$$\begin{aligned} T_{exp} &= T_{ref}\, 2.512^{\,m_{sat}} \\ T_{ref} &= \frac{\text{THsat}}{\text{S0 Speak}} \end{aligned} \tag{7.9}$$

where T_{ref} is the reference time, that is the time necessary to saturate the peak pixel, while m_{sat} is the minimum magnitude among all the targets.
However, there are limits to the result. In particular, it must be within an a priori defined range [Texp_min, Texp_max] and it depends on angular rate, since there is a maximum acceptable image displacement DeltaSmax.
If the maximum angular rate component between ω_x and ω_y, ω_m, multiplied by the T_{exp} (derived above) is bigger than DeltaSmax, then:

$$T_{exp} = \frac{\text{DeltaSmax}}{\omega_m} \tag{7.10}$$

rounding the result according to Texp_step.
Then, the last selection of the phase is based on a quality index. It allows to

[5]This because Q is an invertible matrix.

determine the set of stars that provides the lowest NEA for the next cycles. It has to be computed for at least 3 and maximum 10 brightest stars[6]:

$$\text{Qindex}_J = \sum_{i=1}^{J} \frac{\sigma^2_{NEA_i}}{J^2}, \quad \forall J \in \quad [\text{NSmin}, \text{NSmax}] \subset \mathbb{N} \tag{7.11}$$

So, the set J with the lowest quality index is saved for tracking. It is usually the set with most targets since it goes with J^2.
All the acquisition phase process is summarized in figure 7.6.

Tracking phase

TP can start if there are at least 3 stars after acquisition. In fact, only in this case, the pattern could be recognized with TRIAD elements.
Again, there is the projection from ICRF to FGS_{RF}, with the difference that now the current quaternion is the one corresponding to the current row of DynamicInputFile.
Then, detection probability follows with another randomly perturbation. Therefore the selection of the targets falling inside CCDs and the filtering of the ones not falling in the quadrant intersections. The external border check, instead, it is not done.
As in AP, the new positions are reported back to MRF and so the unit vectors are computed.
At this point a new algorithm is implemented to simulate the possibility that one star is not detected at one or more cycles. This can be due to different causes, for example to a transition of an undesired object between the S/C and the star. The function "missingstar" needs a declaration of the interested ID, of the starting and of the final cycle of the event.
After other warnings to verify that at least 3 stars are still detected, the absolute attitude measurement is performed. It includes the q-method already described and the angular rate computation. This is done through the evaluation of the attitude determined at the current cycle and the one at the previous one:

$$\dot{q} = \frac{q_n - q_{n-1}}{\text{Delta\_time}} \tag{7.12}$$

Then, there is a relation between the quaternion first derivative and angular rate, since it is the derivative of Euler angles. It can be demonstrated that:

$$\dot{q_n} = \frac{1}{2} Q(q_n) \cdot \Omega \tag{7.13}$$

[6]Qindex = 1 by default, i.e. if there are less than 3 detected stars. In that case the selection is not done.

where Q is a function of the quaternion and Ω a four components vector with the last three being the angular rates:

$$Q(q_n) = \begin{bmatrix} q_{n_x} & q_{n_r} & -q_{n_z} & q_{n_y} \\ q_{n_y} & q_{n_z} & q_{n_r} & -q_{n_x} \\ q_{n_z} & -q_{n_y} & q_{n_x} & q_{n_r} \\ q_{n_r} & -q_{n_x} & -q_{n_y} & -q_{n_z} \end{bmatrix} \tag{7.14}$$

$$\Omega = \begin{bmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

From here a key step starts. In fact, the ultimate scope of the simulator is to understand the compliance of FGS to AME and RME requirements. For this, an error in terms of Euler angles (ϕ, θ, ψ) is computed.
To do so the expected quaternion (dynamic input) and computed quaternion (from q-method) are evaluated. The simplest way to determine an error is to multiply them, inverting one of the two. It means that one has to represent an opposite rotation, i.e. the first three components with opposite sign:

$$q_{n_{err}} = q_n \cdot q^{-1} \tag{7.15}$$

In this way the error would be halved, since the quaternion components are defined in function of half the rotation. So the result becomes:

$$\begin{aligned} \phi &= 2q_{err}(1) \\ \theta &= 2q_{err}(2) \\ \psi &= 2q_{err}(3) \end{aligned} \tag{7.16}$$

checking that $q_{err}(4) > 0$. If it is not, the angles have to change sign.
Furthermore it is necessary to check if it is the first time the attitude is determined in TP, that means it is the first cycle in which at least 3 stars are detected and provided to the computation algorithm. If the answer is positive, q_n becomes the reference respect to which, at the next times, the relative attitude is derived.
In tracking two modes of determination are foreseen: single mode and fused mode. One uses 1 active CCD at a time, the other instead uses one CCD per PEM, it means 2 active CCDs. In the first case only the absolute attitude is obtained, while in the second one also the relative measurement is done.
In single mode maximum 10 targets per CCD are saved and an output per each detector is generated. In fused, 20 targets are considered and there is a single output. This last always stands for quaternion and related error in Euler angles.

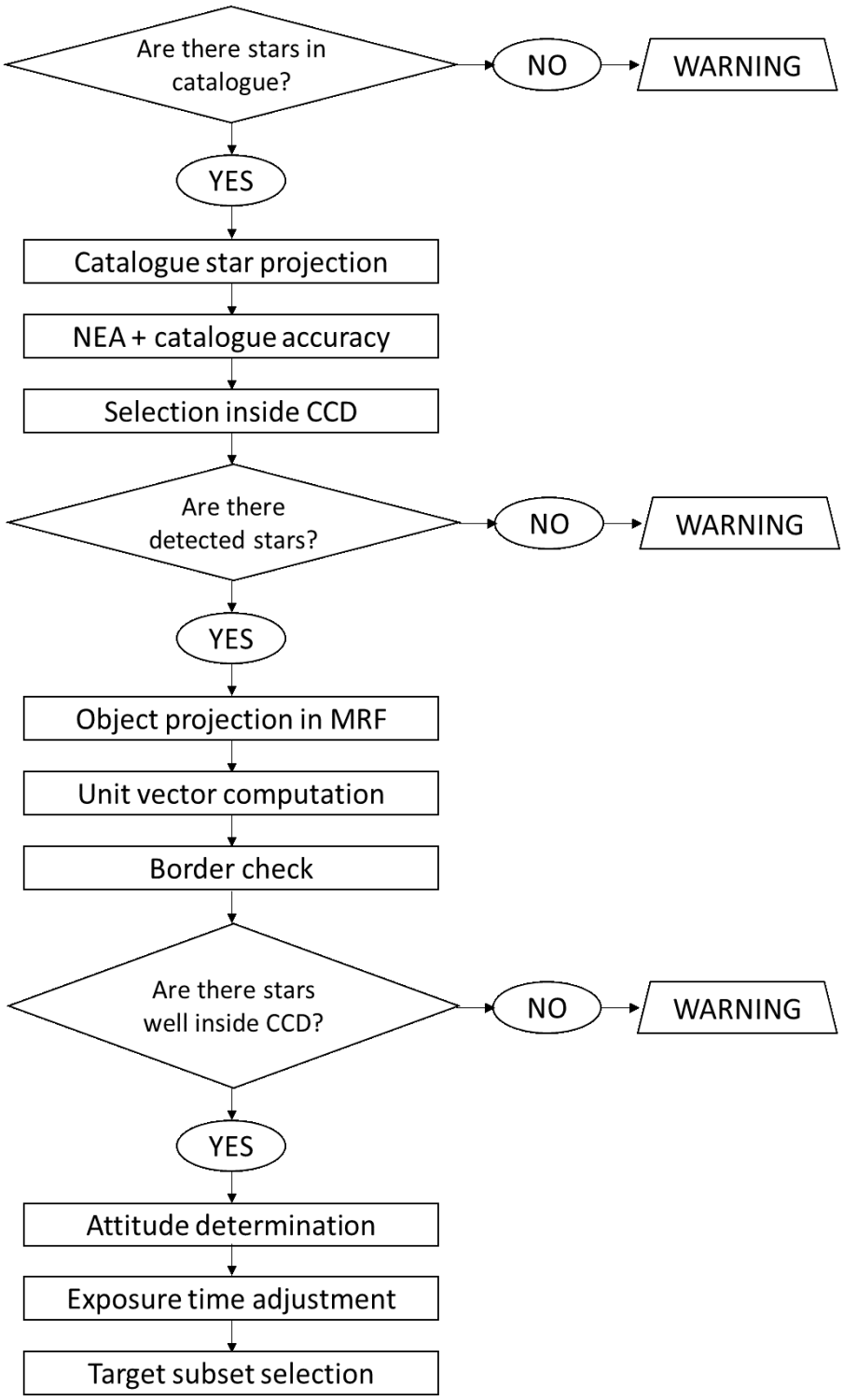


Figure 7.6: AP/IC of the simulator

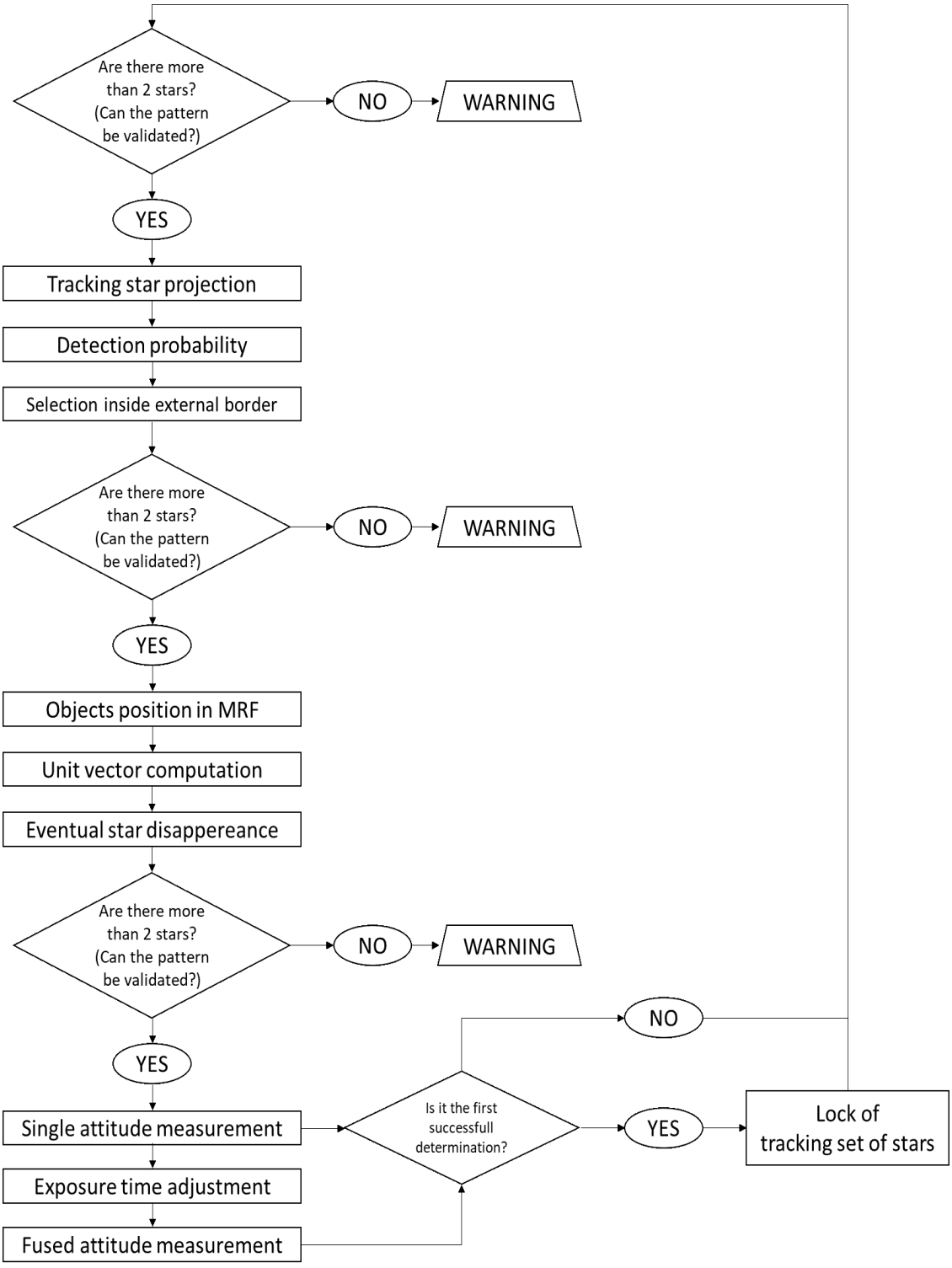


Figure 7.7: TP of the simulator

7.3 Output files and simulation results

The output of the simulator includes a series of variables and they are different between AP, IC and TP. They results in 4 ".mat" files: "AP_results.mat", "IC_results.mat", "TP_results.mat" and "performances.mat". For the first two the following are saved:

- computed quaternion and Euler angles;
- set of stars detected inside CCD, considering NEA and catalogue accuracy;
- set of trackable stars: maximum 10, excluding the ones falling near detectors borders;
- computed exposure time.

These phases are important mostly to predict which stars (and coordinates) the real FGS will detect.
Then, for each tracking cycle, the same are evaluated with the following additional information:

- angular rates (absolute and relative);
- the number and time of the cycle of locking (single and fused CCDs cases);

So, performances are derived through a portion of script that can be labeled as *post-processing* and that returns for each cycle:

- Euler angles;
- error on absolute attitude (single CCD);
- error on absolute attitude (fused CCDs);
- error on relative attitude (fused CCDs).

Error is defined between a measured and a reference quantity. In case of absolute computation, it results from the comparison between measured and dynamic input quaternion. Instead, the relative error is derived thorugh the evaluation of the mean AME computed over the multiple cycles between current and locking cycle. So, the overall RME is usually computed as 3 times the standard deviation of the absolute errors ϵ around their mean value μ:

$$S = 3\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}|\epsilon_i - \mu|^2} \tag{7.17}$$

meaning a validity of 99.7% and where N is the total number of tracking cycles:

$$N = \frac{\text{Nend} - \text{Nstart}}{\text{Delta\_time}} + 1 \tag{7.18}$$

Hence, errors are reported in terms of Euler angles and, for the requirements of FGS, in [as].
An alternative way of evaluating RME has to be done if the commanded mode is RTM. In this scenario, the only quaternions are the relative ones. It follows that the distribution of absolute errors is substituted by another one of relative errors, defined as (m=measured, r=reference):

$$\epsilon_i = [q_m(\text{t=t}_i) - q_m(\text{t=t}_{lock})] - [q_r(\text{t=t}_i) - q_r(\text{t=t}_{lock})] \tag{7.19}$$

where the subtraction of two quaternions means difference in Euler angles.
Anyway, taking as reference the example in figure **??**, simulator starts running at each highlighted time, until 2 or more seconds before the slew are missing.
Consequently, there is an output folder for each commanded tracking mode (see section 6.2).
As already anticipated, different scenarios foresee:

- only high frequency errors;
- high and low frequency errors.

Furthermore, for low frequency errors, there is the difference between self, cross and not calibrated. In the first case, considering two active detectors, error is applied on a single CCD while the other is the reference respect to which calibration occurs. In other cases, error is applied on both CCDs.
It is worth to specify that results described below are all referred to an observation around the pointing $(\alpha, \delta) = (10, 48)$ [deg]. Therefore OBCF and CAF contain only the targets selected respect to that reference.
The chosen configuration is the one with CCD 1 and 3 as active detectors.

Stars projection results

In order to verify the robustness of the projection algorithms, the same image is generated by simulator and ESG. In particular all the targets position are supposed to be consistent between the two parallel simulation tools.
The test consists, therefore, in the projection of all the catalogue stars in the CCDs frame, without affecting the measurement with high or low frequency errors. So, the two output images are overlapped.
The result for CCD 1 and at the first dither is shown in figure 7.8. It can be noticeable that there are objects not detected by simulator, because of their classification. In fact, simulator only works with star-like targets. On the other hand,

the detected ones (marked with *) fall on the same pixel pointed by ESG.
The same result can be repeated for the other CCDs and dithers, with the same accuracy, further validating the first part of the model.
The same figure, then, explicits the fact that simulator does not include the image reconstruction algorithms. It already assumes the target as a point light source located at the center of cluster.
Another consideration can be done for what concerns the influence of the S/C angular rate. It is not already taken into account, since the projection is considered to occur instantaneously and so the process would not be influenced by the exposure time. This also because the angular rate is then accounted for the phase of detection probability simulation, as previously described.

Tracking results with high frequency errors

The case of null bias is an ideal one. This would happen if there are no mounting imprecisions in S/C assembly phase or all the thermoelastic induced deformations would be perfectly compensated by calibration. This is not possible due to the limits of calibrating sensors and to the extreme temperature variations during the whole life of S/C.
However it is an interesting case, to entirely understand the effects of NEA and catalogue uncertaintes on (α, δ). These last are relatively small (order of μas), respect to noise contributions (order of mas).
For the 4 dithers of the observation cited above, AME (mean of $AME(t)$) and RME results are listed in table 7.2. Since both rely on a mean value, the result might change depending on whether they are computed over the whole dither or only over 700 s (as for budget). However, mean AME is negligibly impacted by the time of integration, since it is not a cumulative error.
AME values are relatively low, but not null. They represent the influence of the high frequency noise and catalogue accuracy on measurements.
RME, on the other hand, is bigger because it is the deviation from mean AME and so it does not depend on AME amplitude. Furthermore, if RME is computed over a reduced interval of time, it is not sure that it is lower. In fact, in fourth dither the opposite happens.
RME, as it will be demonstrated in the folllowing pages, is not influenced by bias[7], since it does not take an inertial vector as reference for attitude computation. So, the results obtained in this paragraph can already be compared with budget previsions.

[7]It changes, but only for randomness of noise.

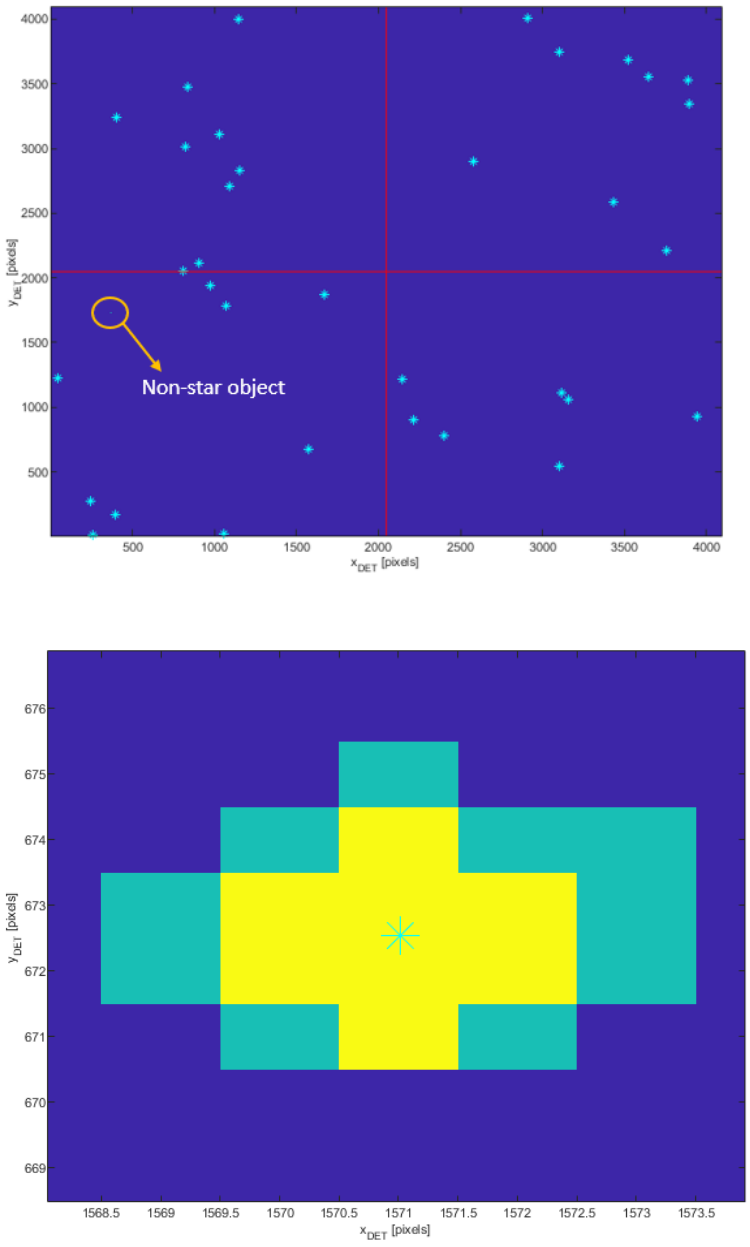


Figure 7.8: Simulator projected stars (*) over ESG image for CCD 1

The expected performances were resumed in the HF line of table 5.5:

$$\begin{aligned} RME_{\phi} &= 0.0020 \\ RME_{\theta} &= 0.0040 \\ RME_{\psi} &= 0.2171 \end{aligned} \tag{7.20}$$

	x axis	y axis	z axis
AME fused mode [as]	1.14E-5	7.28E-5	0.0026
RME fused mode @99.7% [as]	0.0048	0.0087	0.5113
RME fused mode @99.7% [as] (over 700 s)	0.0046	0.0085	0.5069

(a) Dither 1

	x axis	y axis	z axis
AME fused mode [as]	2.43E-6	7.10E-5	0.0030
RME fused mode @99.7% [as]	0.0051	0.0090	0.5140
RME fused mode @99.7% [as] (over 700 s)	0.0050	0.0087	0.4863

(b) Dither 2

	x axis	y axis	z axis
AME fused mode [as]	2.05E-5	3.28E-4	0.0164
RME fused mode @99.7% [as]	0.0049	0.0092	0.5002
RME fused mode @99.7% [as] (over 700 s)	0.0048	0.0090	0.4941

(c) Dither 3

	x axis	y axis	z axis
AME fused mode [as]	5.77E-5	7.68E-5	0.0106
RME fused mode @99.7% [as]	0.0044	0.0083	0.4545
RME fused mode @99.7% [as] (over 700 s)	0.0043	0.0087	0.4610

(d) Dither 4

Table 7.2: Measurement errors with null bias

So, simulator output replies the foreseen order of magnitude, even if it returns a bigger value. However, it is worth to specify that budget results were obtained from a Montecarlo analysis, i.e. from multiple simulations regarding multiple pointings. Therefore, final budget is a weighted value and it does not comprehend all possible scenarios. Anyway, results are all well below the requirements (table 5.1).
Another consideration can be done looking at single CCD and fused CCDs estimation (single CCD is applied only during absolute mode). For example, for first

dither AME becomes (for 4 CCDs):

$$\begin{aligned} AME_\phi &= [5.38\text{E-}4; 0; 0.0013; 0] \\ AME_\theta &= [6.76\text{E-}4; 0; 0.0021; 0] \\ AME_\psi &= [0.0539; 0; 0.1344; 0] \end{aligned} \tag{7.21}$$

that is a less accurate calculation. This is due to the fact that q-method has half the pair of directions than fused mode.
Both in fused and single mode, errors have zero (or similar) mean value. This because bias is null and there are not geometric (steady) uncertaintes. Oscillations on the two main attitude angles, ϕ and θ, over the first dither are reported in figures 7.9 and 7.10.
Relative errors, instead, have not null mean value, according to dynamic attitude of the S/C during the observation. As shown in figure 7.11, still focusing on first dither, Euler angles drift away from initial value, mainly due to telescope wheel assemblies adjustment. Accordingly to this, ϕ has a positive mean error value and θ has a negative one.

Tracking results with bias

To introduce low frequency errors, three relevant cases are considered:

- Not calibrated: maximum uncertainty;
- Modified frame: self-calibration, uncertainty applied on a single CCD;
- Required calibrated: cross-calibration.

Not calibrated case

As already anticipated, "not calibrated" stands for on ground calibrated and it is the worst case. The other two are gradually better, as listed.
Starting from the first of the three, the following uncertaintes are inserted modifying "InputFile.m" parameters f_b, $x_{0_{det_b}}$, $y_{0_{det_b}}$ and *tilt*:

- local focal length knowledge: 20 mm;
- CCD tilt knowledge: 143 as;
- CCD relative tilt knowledge: 286 as;
- CCD positioning knowledge: 0.1042 mm;
- CCD relative positioning knowledge: 0.2084 mm.

	x axis	y axis	z axis
AME fused mode [as]	2.4564	1.1823	80.3150
RME fused mode @99.7% [as]	0.0047	0.0088	0.4911
RME fused mode @99.7% [as] (over 700 s)	0.0046	0.0090	0.4894

(a) Dither 1

	x axis	y axis	z axis
AME fused mode [as]	2.4547	1.1765	79.7240
RME fused mode @99.7% [as]	0.0047	0.0093	0.5371
RME fused mode @99.7% [as] (over 700 s)	0.0048	0.0090	0.5303

(b) Dither 2

	x axis	y axis	z axis
AME fused mode [as]	2.4860	1.2058	78.7070
RME fused mode @99.7% [as]	0.0053	0.0088	0.4994
RME fused mode @99.7% [as] (over 700 s)	0.0052	0.0088	0.4844

(c) Dither 3

	x axis	y axis	z axis
AME fused mode [as]	2.5122	1.1801	78.7602
RME fused mode @99.7% [as]	0.0043	0.0082	0.4519
RME fused mode @99.7% [as] (over 700 s)	0.0041	0.0083	0.4543

(d) Dither 4

Table 7.3: Measurement errors with maximum bias

Results are reported in table 7.3. Then, referring to table 5.5, all AME values are under budget estimation. Instead, RME has the same order of magnitude and, as above mentioned, it does not change significantly with bias. For this reason, from here RME computations are not reported anymore.
Since this is a case of maximum bias, it is interesting to note that errors now have a mean value that is far from zero. Oscillations on the two main attitude angles over the first dither are reported in figures 7.12 and 7.13.

Modified frame case

In self-calibration a CCD is taken as reference, i.e. having a null tilt and position bias respect to VIS. So, the following parameters are considered:

- local focal length knowledge: 3 mm;
- CCD tilt knowledge: - as;
- CCD relative tilt knowledge: 7 as;
- CCD positioning knowledge: - mm;
- CCD relative positioning knowledge: 0.025 mm.

	x axis	y axis	z axis
AME fused mode [as]	0.4741	0.2474	9.6417

(a) Dither 1

	x axis	y axis	z axis
AME fused mode [as]	0.4743	0.2492	9.6433

(b) Dither 2

	x axis	y axis	z axis
AME fused mode [as]	0.4790	0.2509	9.4362

(c) Dither 3

	x axis	y axis	z axis
AME fused mode [as]	0.4824	0.2468	9.4381

(d) Dither 4

Table 7.4: Measurement errors with self-calibration

Results are reported in table 7.4. Then, referring to table 5.7, AME values are below budget estimation, except for ϕ angle. It is around 50 μarcsec bigger than expected. However, as already said, it is a plausible result, being budget derived from a Montecarlo simulation. It is necessary to note that the same is anyway under requirement of 0.6".

Required calibrated case

In cross-calibration, VIS is the reference and its measurement is compared to that of FGS. The following parameters are foreseen:

- local focal length knowledge: 3 mm;
- CCD tilt knowledge: 7 as;
- CCD relative tilt knowledge: 14 as;
- CCD positioning knowledge: 0.01 mm;
- CCD relative positioning knowledge: 0.01 mm.

Results are reported in table 7.5. Furthermore, the same considerations of the self-calibration case on ϕ angle are still valid and here omitted.
In addition, it is worth to specify that AME requirement over z axis (8.7"), for this particular pointing, would be met only with cross-calibration. Anyway, the relative mode would garantuee the required pointing stability (RME), even if no calibration occurs.

	x axis	y axis	z axis
AME fused mode [as]	0.4113	0.0995	3.8395

(a) Dither 1

	x axis	y axis	z axis
AME fused mode [as]	0.4117	0.1014	3.8580

(b) Dither 2

	x axis	y axis	z axis
AME fused mode [as]	0.4169	0.1035	3.7697

(c) Dither 3

	x axis	y axis	z axis
AME fused mode [as]	0.4193	0.0990	3.7722

(d) Dither 4

Table 7.5: Measurement errors with cross-calibration

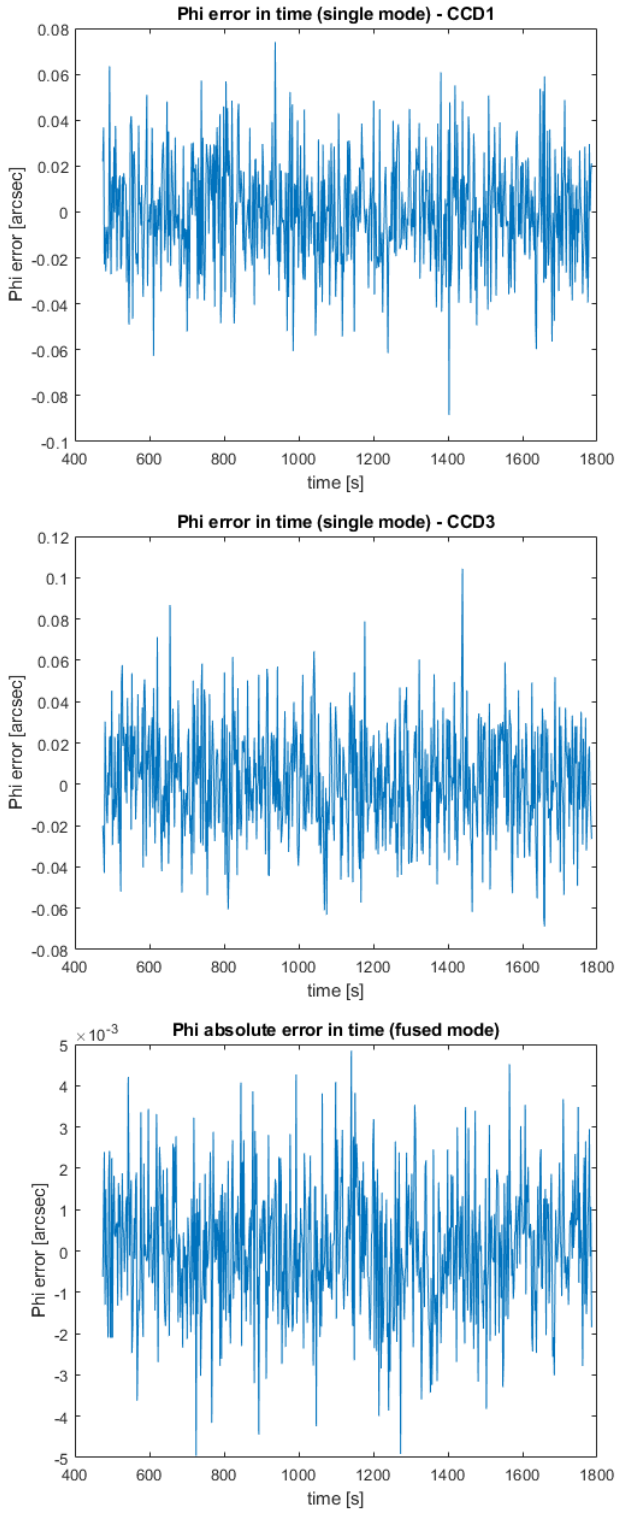


Figure 7.9: ϕ error over time without bias

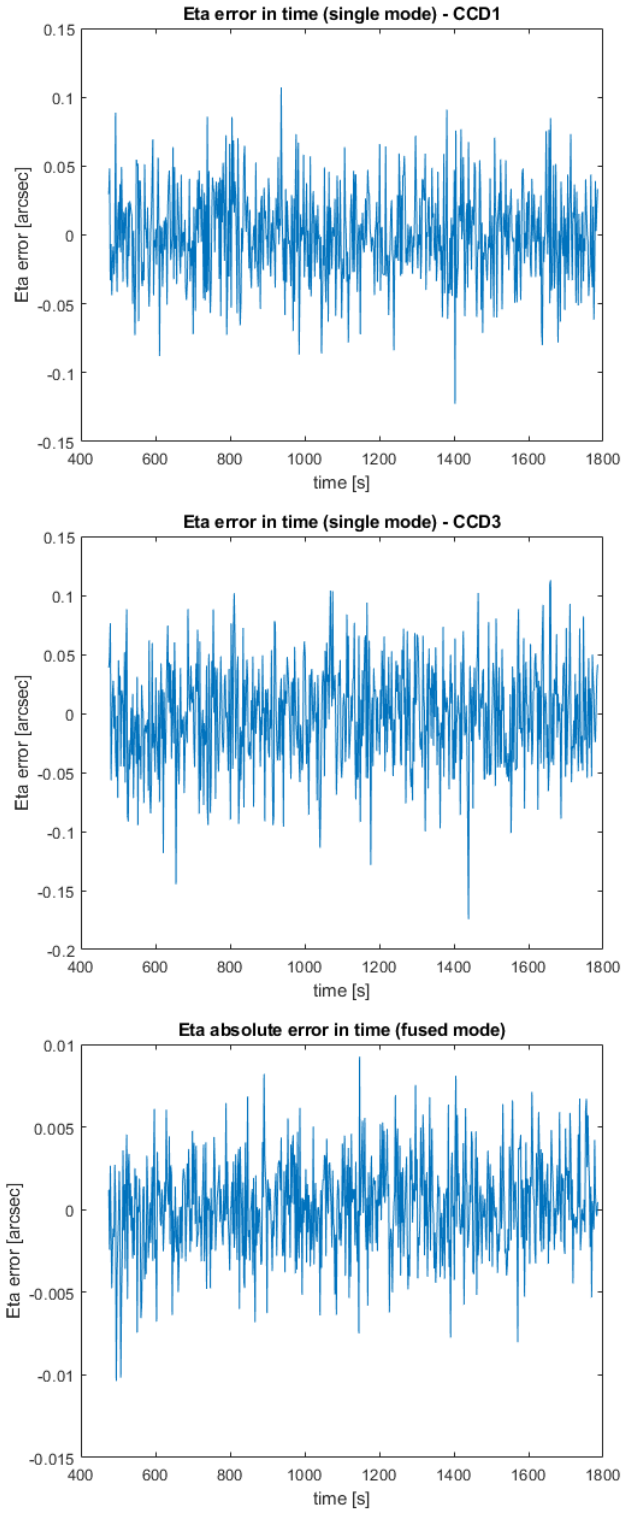


Figure 7.10: θ error over time without bias

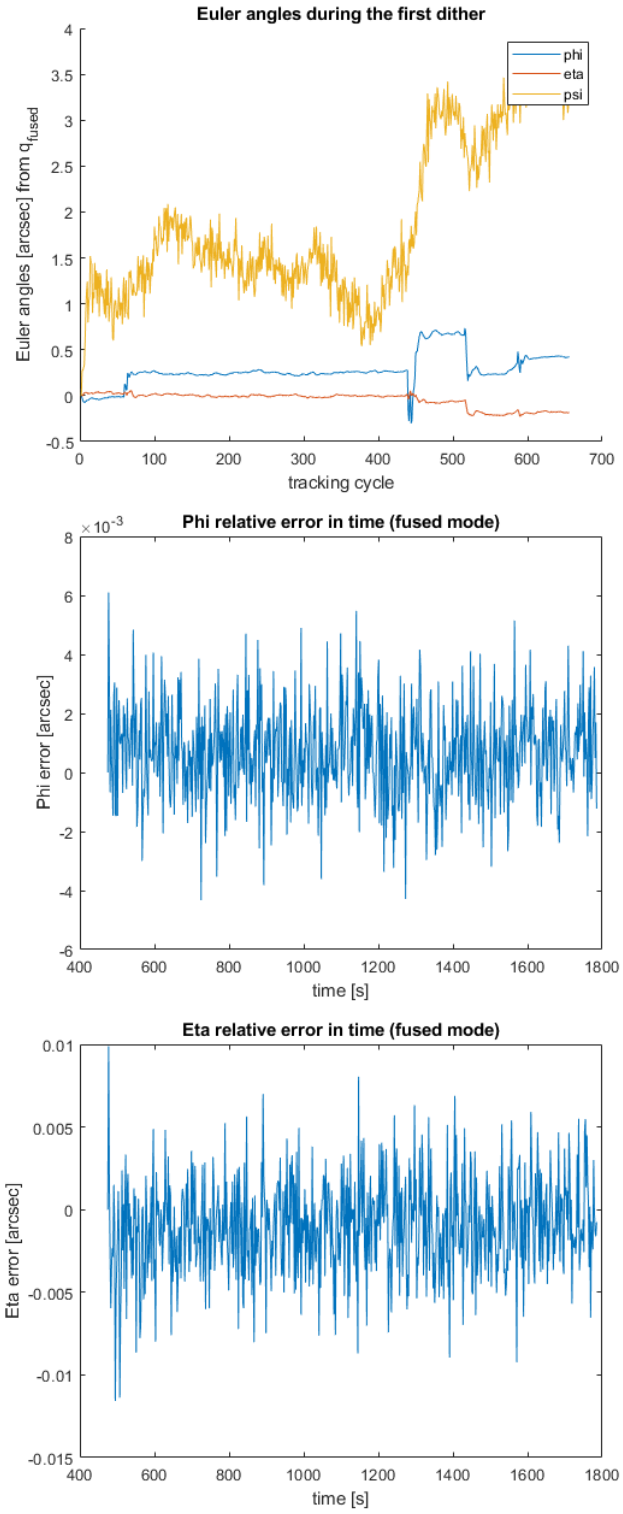


Figure 7.11: Relative errors over time

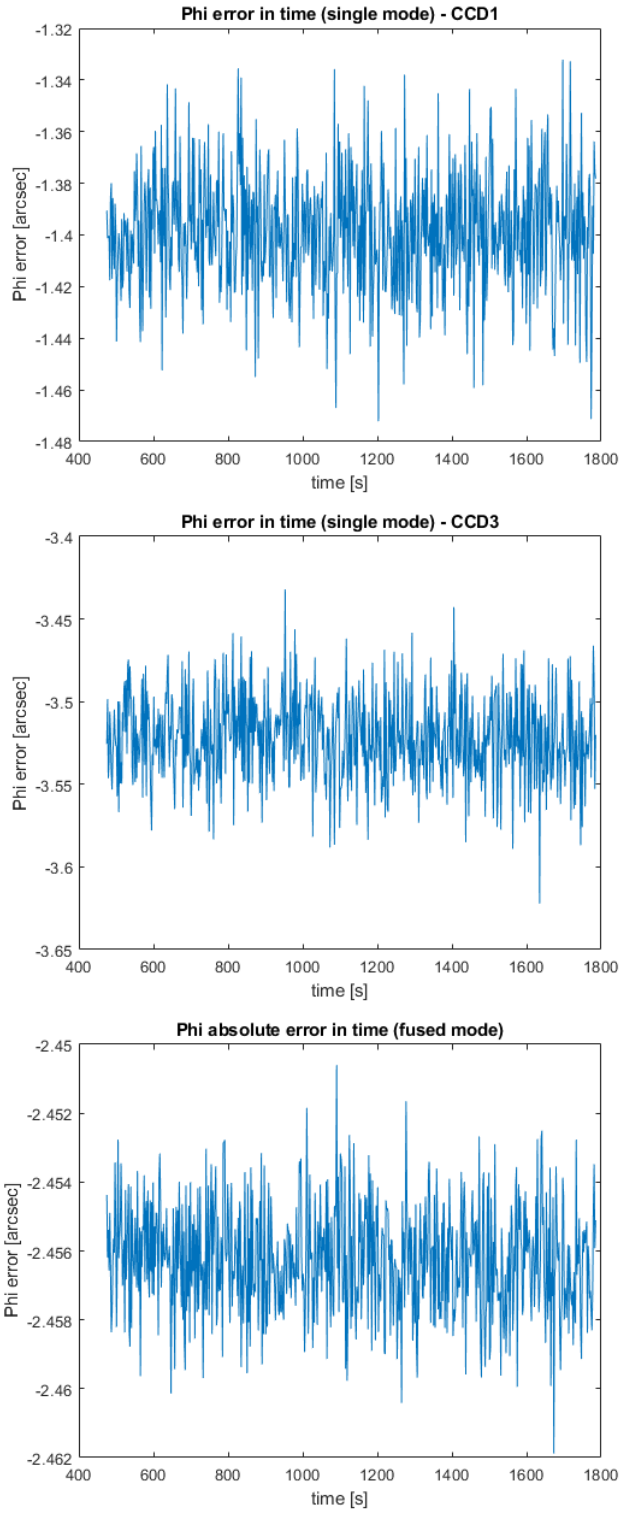


Figure 7.12: ϕ error over time with maximum bias

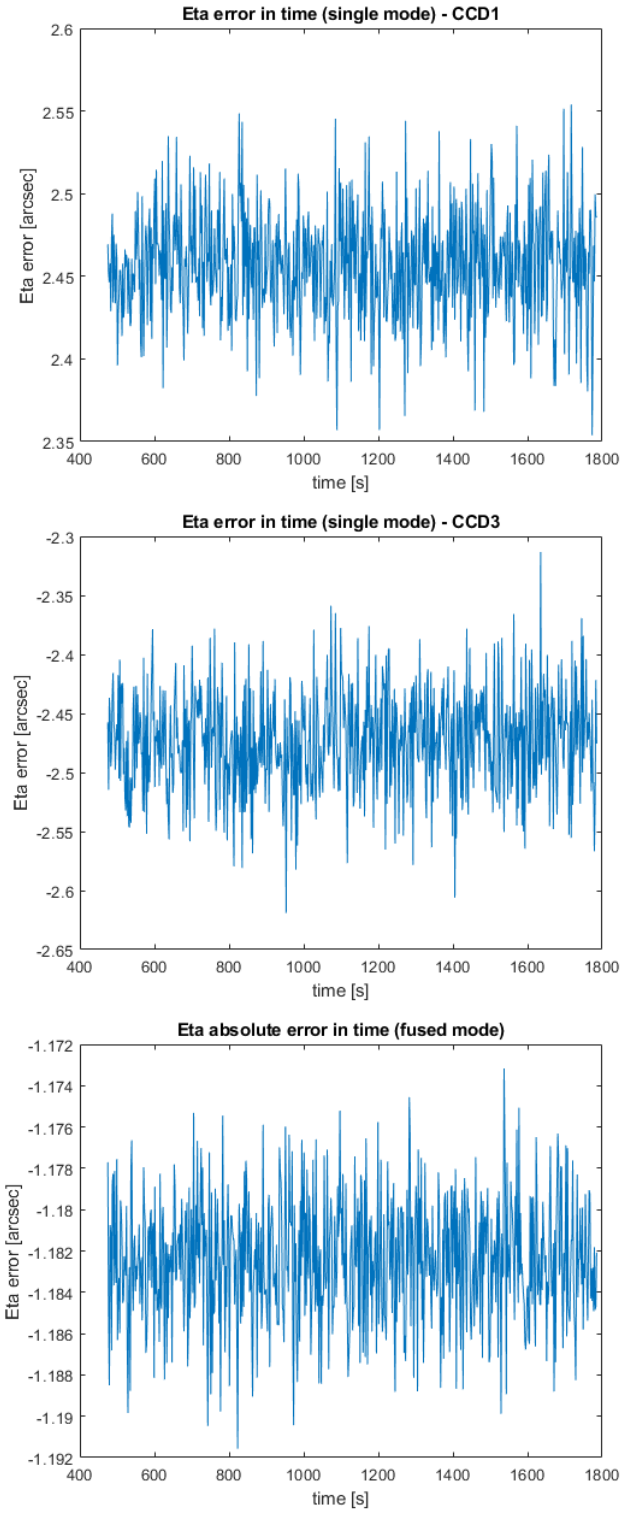


Figure 7.13: θ error over time with maximum bias

Chapter 8

Conclusions and way forward

The ESA Euclid mission has been presented with particular focus on the attitude control of the spacecraft during the science operations which are very challenging in terms of absolute pointing accuracy and pointing stability.
The Fine Guidance Sensor is the key sensor to meet the pointing requirements, so its design and main functionalities have been described. The tracking modes have been analyzed in details and a simulator with a reduced subset of algorithms has been built to estimate the FGS Absolute and Relative Measurement Error.
As a demonstration case, the FGS tracking operations during an entire Euclid observation have been simulated, in particular its performances in terms of AME and RME have been derived. Starting from the uncalibrated conditions (worst case) up to the self-calibrated and cross-calibrated conditions, measurement accuracy is obtained.
Actually, performances contributors included in the model are:

- high frequency: NEA;
- field dependent: star catalogue accuracy;
- low frequency: bias on focal length, CCD position and tilt.

Results, in terms of errors on Euler angles, are in line with the expected budgets. Taking as reference the first dither the following AME values are obtained:

	x axis [as]	y axis [as]	z axis [as]
AME not calibrated	2.4564	1.1823	80.3150
AME self-calibrated	0.4741	0.2474	9.6417
AME cross-calibrated	0.4113	0.0995	3.8395

Table 8.1: FGS simulated absolute measurement errors

RME, instead, is not influenced by low frequency errors and it can be considered around the output of the simulation without bias:

	x axis [as]	y axis [as]	z axis [as]
RME (over 700 s)	0.0048	0.0087	0.5113

Table 8.2: FGS simulated relative measurement errors

In general the comparison of a simulation result with budget expected performances can not be a full demonstration of the consistency of simulator in replying real FGS response. First of all, to rise its confidence level, multiple runs on multiple pointings/observations need to be executed and, then, tests on real sensor are required to better validate the model. With this objective, all the inputs needed to run the same test on the real FGS unit have been prepared. It is going to be performed by FGS provider in next months and, so, it is a short term activity already in place.
A near future utilization of the tool is intended to be in support of ESG stimulation on EQM. In particular, tests are going to concern the behaviour of FGS under RTM commands and, after those, also ATM ones. For the first mode, RME will be derived from relative quaternions and Euler angles, while for the second one, the same together with AME starting from absolute results. In such procedures, simulator will help to easily predict FGS behaviour or verify test results.
In addition to these applications, then, in case of unexpected mission scenarios or anomalies, investigation on possible causes can have a fast support. Moreover, new error contributions could be introduced in the model, like for instance optical distortion coefficients or a temporary disappearance of a detected target.
Further development could be made in order to account for window mode acquisitions. It would mean that in each phase of tracking mode, an implementation of attitude propagation and future windows selection for the next cycle is needed.
In a wider context, many future high pointing precision S/Cs will include a Fine Guidance Sensor, sometimes sharing its FOV with the scientific instrument, like foreseen by Euclid AOCS architecture. Therefore, algorithms developed for Euclid FGS and included in simulator can represent a basis for studying and early prototyping new FGS applications; the implemented features, in fact, allow to explore other geometries, star magnitudes and CCD properties.

Appendices

Appendix A

HEALPIX conversion

"HEALPIX_conversion.c" main script

```
/* ----------------------------------------------------------------

HEALPIX_conversion.c

----------------------------------------------------------------*/

/* Script that reads 48 Healpix files and translates them
in 192 files. It also converts them from ASCII to binary
or viceversa (modify constant "format" definition!).
Pay attention: it does not overwrite, so it needs
an input and an output file path! */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <stdint.h>
#include <stdbool.h>
#include <libgen.h>
#include "HEALPIX_conversion.h"

static void util_fail_ (const char *file, int line,
const char *func, const char *msg)
  {
  fprintf(stderr,"%s, %i (%s):\n%s\n",file,line,func,msg);
  exit(1);
  }
```

```
#if defined (__GNUC__)
#define UTIL_FUNC_NAME__ __func__
#else
#define UTIL_FUNC_NAME__ "unknown"
#endif
#define UTIL_ASSERT(cond,msg) \
  if(!(cond)) util_fail_(__FILE__,__LINE__,UTIL_FUNC_NAME__,msg)
#define UTIL_FAIL(msg) \
  util_fail_(__FILE__,__LINE__,UTIL_FUNC_NAME__,msg)

static const short utab[]={
#define Z(a) 0x##a##0, 0x##a##1, 0x##a##4, 0x##a##5
#define Y(a) Z(a##0), Z(a##1), Z(a##4), Z(a##5)
#define X(a) Y(a##0), Y(a##1), Y(a##4), Y(a##5)
X(0),X(1),X(4),X(5)
#undef X
#undef Y
#undef Z
};

static const double twothird = 2.0 / 3.0;
static const double pi = 3.14159265358979323846264338327950288419 7;
static const double twopi = 6.28318530717958647692528676655900576839 4;
static const double inv_halfpi = 0.63661977236758134307553505349005 74;


/* START OF MAIN FUNCTION */
int main()
{

   // Initializations
   long nside; /* Number of divisions on base-res pixel side*/
   int64_t ipix;                      /* HEALPIX pixel index */
   int64_t level = 2;        /* level of HEALPIX resolution */
   int64_t n1 = 12;                             /* constant */
   int64_t n2 = 2;                              /* constant */
   int64_t n3 = 35;                             /* constant */
   int64_t n4 = 4;                              /* constant */
   double theta;        /* HEALPIX conventional declination */
   double phi;            /* HEALPIX conventional ascension */
   int i;                  /* index for HEALPIX input files */
```

```
/* name of one of 48 ASCII original files */
char fileIN;

/* name of one of 192 converted files (ASCII or binary) */
char fileOUT;

/* 0 if the desired conversion is asci2bin, 2 if it is bin2asci ,
1 if it is requested to pass from 48 to 192 files */
int format = 0;

if (format==0) {

   /* to make the conversion ascii/binary */
  for(i=0;i<192;i++){
       sprintf(&fileIN, "../ISC/ISC_v2_192files/healpix%
          03d.csv", i);
          convert_text(&fileOUT, format, i);
   }

}
else if (format==2) {

   /* to make the conversion binary/ascii */
   for(i=0;i<192;i++){
       sprintf(&fileIN, "../ISC/ISC_v2_192files/healpix%
          03d.bin", i);
          convert_text(&fileOUT, format, i);
   }

}
else if (format==1) {

for (i=0;i<48;i++)
{
/* Input HEALPIX.csv file (48 files) */
if (i<10){
 sprintf(&fileIN, "../ISC/ISC_v2/healpix0%d.csv",
           i);
}
else{
 sprintf(&fileIN, "../ISC/ISC_v2/healpix%d.csv",
```

```
            i);
}

FILE *fp_in;

printf("Input file %d = %s\n", i, &fileIN);

/* Output HEALPIX.csv file (192 files) */
FILE *fp_out;

/* To open the original file */
fp_in = fopen(&fileIN, "r");

/* nside = 2 for 48 files */
/* nside = 4 for 192 files */
nside = pow(2, level);

/* To scan and save in 192 .csv files */
printf("Start scanning...\n");
for (;;)
{

   int64_t SourceId;
   double Alpha;
   double Delta;
   double AlphaError;
   double DeltaError;
   double MuAlpha;
   double MuDelta;
   double MuAlphaError;
   double MuDeltaError;
   double Mag;
   double MagError;
   int Classification;
   int Neighbor;
   int Variability;
   int BrightNeighbor;
   double NearestNeighborDist;
   double Parallax;
   double ParallaxError;

   const int stringa_len = 1024;
```

```
      char stringa[stringa_len];
      int nread = fscanf(fp_in, "%llu, %[^\n]s",
                 &SourceId, stringa);

     if (nread < 2)
     {
     if (nread > 0)
      {
fprintf(stderr, "Misalignment in inputfile");
exit(1);
      }
     break;
     }

      nread = sscanf(stringa,"%lf,%lf,%lf,%lf,%lf,%
                 lf,%lf,%lf,%lf,%lf,%d,%d,%d,%d,%lf,%lf,%lf",
                 &Alpha, &Delta, &AlphaError, &DeltaError,
                 &MuAlpha, &MuDelta,&MuAlphaError, &MuDeltaError,
                 &Mag, &MagError, &Classification, &Neighbor,
                 &Variability, &BrightNeighbor,
                 &NearestNeighborDist,&Parallax, &ParallaxError);

      if (nread < 16)
      {
  fprintf(stderr, "Misalignement in inputfile");
  exit(1);
      }


      /* to pass from IAU to HEALPIX convention */
      theta = - Delta * pi / 180 + pi/2;
      phi = Alpha * pi / 180;

      /* to compute the HEALPIX index from alpha and
                 theta */
      ang2pix_nest64(nside, theta, phi, &ipix);

      /* to save in 192 files */
      sprintf(&fileOUT, "../ISC/ISC_v2_
                 192files/healpix%03d.csv", ipix);
      fp_out = fopen(&fileOUT, "a");
```

```
      fprintf(fp_out, "%19llu,%14.10lf,%14.10lf,%
                  11.4e,%11.4e,%12.4lf,%12.4lf,%11.4e,%11.4e,%
                  7.4lf,%11.4e,%1d,%1d,%1d\n",
      SourceId, Alpha, Delta, AlphaError,DeltaError,
                  MuAlpha, MuDelta,MuAlphaError,MuDeltaError,
                  Mag, MagError,Classification,Neighbor,
                  Variability);

      fclose(fp_out);
   }

   fclose(fp_in);
   }

   }
   else{

   fprintf(stderr, "Not valid format input value");
   exit(1);

   }

   return 0;
}
/* END OF MAIN FUNCTION */
```

Appendix B

CAF generation

"CAFgeneration.m" main script

```
%CAFgeneration.m
%Main script for the generation of CAF starting from ISCF
format long
close all
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%VARIABLES DEFINITION
%star = structure for the stars present in ISCF, fields :
%          ra [rad], decl[rad], mag, flag, ux, uy, uz
%q = quaternions from OSIF : 4 dithers x 4 components
%ra0, decl0 = reference ascension and declination(1st quaternion)
%reference = unitary vector deriving from (ra0, decl0)
%radius_ref = radius for the reference attitude [rad]
%coor_det = position in MRF [mm]
%x_fgs = x coordinate in MRF [mm]
%y_fgs = y coordinate in MRF [mm]
%x_det = x coordinate in DET_RF [pixels]
%y_det = y coordinate in DET_RF [pixels]
%selected_temp = struct for filtered stars according to FOV:
%          ra [rad], decl[rad], mag, flag, ux, uy, uz
%selected = struct for total found stars to avoid double output:
%          ra [rad], decl[rad], mag, flag, ux, uy, uz
%n_CCD = index for CCD
%dither = index for dither
%target = index for targets in plots generation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PARAMETERS INITIALIZATION
iscf = '0000000008.iscf';                          %input ISCF file
osif = '0000000008.osif';                          %input OSIF file
fid = fopen('0000000008_2.caf','w');               %output CAF file
Px0 = [0 0 0 0];              %offset for each CCD (x coordinate)
Px1 = [1 1 1 1];  %multiplying factor for each CCD (x coordinate)
Py0 = [0 0 0 0];              %offset for each CCD (y coordinate)
Py1 = [1 1 1 1];  %multiplying factor for each CCD (y coordinate)
PIXsize=12*10^(-3);                            %pixel dimension [mm]
focal=24500;                                      %focal length [mm]
x0det=[206 256 -206 -256];                     %DET_RF x offset [mm]
y0det=[303.931 413.931 413.931 303.931];    %DET_RF y offset [mm]
alfa=[0 1 0 0 0 0 0 0]; %poly coeff. for opt. distorption along x
beta=[0 1 0 0 0 0 0 0]; %poly coeff. for opt. distorption along y
radius_ref = 3*pi/180;                                    %3 degrees
%axis orientation matrix
A(:,:,1)=[0 1;1 0];
A(:,:,2)=[0 -1;-1 0];
A(:,:,3)=[0 -1;-1 0];
A(:,:,4)=[0 1;1 0];
%mounting parameter DET to FGS
tilt=[0 0 0 0];                                    %tilt angle [rad]
tilt_mat(:,:,1)=[cos(tilt(1)) sin(tilt(1));...
    -sin(tilt(1)) cos(tilt(1))];
tilt_mat(:,:,2)=[cos(tilt(2)) sin(tilt(2));...
    -sin(tilt(2)) cos(tilt(2))];
tilt_mat(:,:,3)=[cos(tilt(3)) sin(tilt(3));...
    -sin(tilt(3)) cos(tilt(3))];
tilt_mat(:,:,4)=[cos(tilt(4)) sin(tilt(4));...
    -sin(tilt(4)) cos(tilt(4))];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%to read the ISCF
star = readISCF(iscf);

%to read the OSIF
q = readOSIF(osif);

%reference attitude
[ra0,decl0] = q2radec(q(1,:));
[reference.ux,reference.uy,reference.uz] = radec2vec(ra0,decl0);
```

```
%to pass from RA dec to vector
[star.ux,star.uy,star.uz] = radec2vec(star.ra,star.decl);

for n_CCD = 1:4
    for dither = 1:4
        if dither == 1
            start = 1;
        else
            start = 0;
        end
        %to make the projection from IRF to MRF and DET_RF
        [x_fgs,y_fgs,coor_det,x_det,y_det]=...
          ALG_0220(q(dither,:),star,Px0,Px1,Py0,Py1,PIXsize,...
          focal,x0det,y0det,alfa,beta,A,tilt_mat,n_CCD);

        figure(1)
        plot(x_fgs,y_fgs,'*')
        title('Star projection from ISCF in FGS_{RF}')
        xlabel('x_{fgs} [mm]')
        ylabel('y_{fgs} [mm]')
        hold on

        %filtering of targets for the 4 dithers
        [selected_temp] = selection(star,x_det,y_det,dither);

        %to make again the projection to MRF and DET_RF
        [x_fgs_sel,y_fgs_sel,coor_det_sel,x_det_sel,y_det_sel]=...
             ALG_0220(q(dither,:),selected_temp,Px0,Px1,Py0,Py1,...
             PIXsize,focal,x0det,y0det,alfa,beta,A,tilt_mat,n_CCD);

        figure()
        hold on
        title(strcat('Selection projection in DET_{RF} - CCD',...
            num2str(n_CCD),'dither',num2str(dither)))
        xlabel('x_{det} [pixels]')
        ylabel('y_{det} [pixels]')
        for target = 1:length(selected_temp.id)
            plot(x_det_sel(target),y_det_sel(target),'*')
            txt = ['ID=',num2str(selected_temp.id(target))];
            text(x_det_sel(target),y_det_sel(target),txt)
        end
```

```
        %to update selected set without double the targets
        if start == 1 && n_CCD == 1
            selected = selected_temp;
        else
            [selected] = undouble(selected,selected_temp);
        end
    end

    %to make again the projection to MRF and DET_RF
    [x_fgs_sel,y_fgs_sel,coor_det_sel,x_det_sel,y_det_sel]=...
        ALG_0220(q(dither,:),selected,Px0,Px1,Py0,Py1,PIXsize,...
        focal,x0det,y0det,alfa,beta,A,tilt_mat,n_CCD);

    if n_CCD == 4
        figure()
        hold on
        title('Selected stars projection in MRF')
        xlabel('x_{fgs} [mm]')
        ylabel('y_{fgs} [mm]')
        for target = 1:length(selected.id)
            plot(x_fgs_sel(target),y_fgs_sel(target),'*')
            txt = ['ID=',num2str(selected.id(target))];
            text(x_fgs_sel(target),y_fgs_sel(target),txt)
        end
    end

end

%to generate CAF file in output
fprintf(fid,'%15.14f  %15.14f %15.14f\n',...
    reference.ux,reference.uy,reference.uz);
fprintf(fid,'%3.2f\n\n',radius_ref);

    for row = 1:length(selected.ux)
        fprintf(fid,'%15.14f  %15.14f %15.14f %6.4f %d\n',...
            selected.ux(row),selected.uy(row),selected.uz(row),...
            selected.mag(row),selected.flag(row));

    end
fclose(fid);
```

"q2radec.m" function

```
function [RA,DEC] = q2radec(quat)
%function to pass from quaternion to right asc, decl [rad]

%inertial pole
vec_body=[0 0 -1];

%rotation matrix
RotMat = ALG_0002(quat(1), quat(2), quat(3), quat(4));

%vector in inertial reference frame
vec_irf = RotMat'*vec_body';
vec_irf = vec_irf./norm(vec_irf);

%right ascension, declination [rad]
RA= atan2(vec_irf(2),vec_irf(1));
DEC= asin(vec_irf(3));
```

"radec2vec.m" function

```
function [ux, uy, uz] = radec2vec(ra, decl)
%function to pass from (ra,dec) [rad]
%to unitary vector (ux,uy,uz) pointing

ux = cos(ra).*cos(decl);
uy = sin(ra).*cos(decl);
uz = sin(decl);
```

"selection.m" function

```
function [selected] = selection(star,x_det,y_det,dither)
%selection of targets for the 4 dithers

accuracy1 = 90/0.1;       %accuracy for the first dither [pixels]
accuracy2 = 5/0.1;       %accuracy for the second dither [pixels]
side = 4096;                                %side of CCD [pixels]
s = 0;                                   %index for selected targets
selected = struct();                        %selected set of targets

%filtering process
```

```
for i = 1:length(star.ra)
    if (dither ==1)
        if (-accuracy1<=x_det(i) && x_det(i)<=side+accuracy1) &&...
                (-accuracy1<=y_det(i) && y_det(i)<=side+accuracy1)
            s = s+1;
            selected.id(s) = star.id(i);
            selected.ra(s) = star.ra(i);
            selected.decl(s) = star.decl(i);
            selected.mag(s) = star.mag(i);
            selected.flag(s) = star.flag(i);
            selected.ux(s) = star.ux(i);
            selected.uy(s) = star.uy(i);
            selected.uz(s) = star.uz(i);
        end
    else
        if (-accuracy2<=x_det(i) && x_det(i)<=side+accuracy2) &&...
                (-accuracy2<=y_det(i) && y_det(i)<=side+accuracy2)
            s = s+1;
            selected.id(s) = star.id(i);
            selected.ra(s) = star.ra(i);
            selected.decl(s) = star.decl(i);
            selected.mag(s) = star.mag(i);
            selected.flag(s) = star.flag(i);
            selected.ux(s) = star.ux(i);
            selected.uy(s) = star.uy(i);
            selected.uz(s) = star.uz(i);
        end
    end
end
```

"undouble.m" function

```
function [selected] = undouble(selected,selected_temp)
%function to update selected set without double the targets

new = struct();                          %structure for new targets
n = 0;                                       %index for new targets

%to find new targets
for i = 1:length(selected_temp.ra)
    found = 0;                %flag becomes 1 if there is a match
```

```
    for j = 1:length(selected.ra)
        if (selected_temp.ra(i) == selected.ra(j)) &&...
                (selected_temp.decl(i) == selected.decl(j))
            found = 1;
        end
    end
    if (found == 0)  %if the selected target is not already found
        n = n+1;
        new.id(n) = selected_temp.id(i);
        new.ra(n) = selected_temp.ra(i);
        new.decl(n) = selected_temp.decl(i);
        new.mag(n) = selected_temp.mag(i);
        new.flag(n) = selected_temp.flag(i);
        new.ux(n) = selected_temp.ux(i);
        new.uy(n) = selected_temp.uy(i);
        new.uz(n) = selected_temp.uz(i);
    end
end

if n > 0
    %to append new targets to selected set
    selected.id((length(selected.id)+1):...
        (length(selected.id)+length(new.id))) = new.id;
    selected.ra((length(selected.ra)+1):...
        (length(selected.ra)+length(new.ra))) = new.ra;
    selected.decl((length(selected.decl)+1):...
        (length(selected.decl)+length(new.decl))) = new.decl;
    selected.mag((length(selected.mag)+1):...
        (length(selected.mag)+length(new.mag))) = new.mag;
    selected.flag((length(selected.flag)+1):...
        (length(selected.flag)+length(new.flag))) = new.flag;
    selected.ux((length(selected.ux)+1):...
        (length(selected.ux)+length(new.ux))) = new.ux;
    selected.uy((length(selected.uy)+1):...
        (length(selected.uy)+length(new.uy))) = new.uy;
    selected.uz((length(selected.uz)+1):...
        (length(selected.uz)+length(new.uz))) = new.uz;
end
```

Appendix C

FGS simulator

"Main.m" main script

```
%Main.m
%FGS Simulator main script

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%VARIABLES DEFINITION

%n_target=number of targets from OBCF

%star=object "star" from OBCF:
%         CCD,ID,Mi,ux,uy,uz,rasc_acc [mrad],decl_acc [mrad],flag

%q=quaternions from dynamic file
%ang_rate=expected angular rate [rad/s]
%time=time when dynamic input is provided [s]
%time_t=time of each tracking cycle [s]
%N=index for dynamic input row
%ap=index for the two acquisition phases (AP+IC)
%tc=index for tracking cycles
%T=orientation matrix from q
%coor_det=distorted position in MRF [mm]
%x_fgs=x coordinate in MRF [mm]
%y_fgs=y coordinate in MRF [mm]
%x_det=x coordinate in DET_RF [pixels]
%y_det=y coordinate in DET_RF [pixels]

%detected=object "detected" stars :
%         CCD,ID,Mi,ux,uy,uz,rasc_acc,decl_acc,
```

```
%             detected.x_det_true_p,detected.y_det_true_p,ubody_m

%..._det_true_p=detected position inside detector [pixels]
%ubody=projected unit vector in body coordinates (MRF)
%ubody_m=measured unit vector in body coordinates (MRF)
%detectedt=detected object in each tracking cycle

%..._temp=variable that changes dimensions every "for" cycle,
%          useful because there can be changes after
%                                     detection and sub-selection

%x_fgs_m,y_fgs_m=measured positions in MRF [mm]
%qabs_ap=computed quaternion in acquisition phase

%track=object stars after border check:
%         CCD,ID,Mi,ux,uy,uz,rasc_acc,decl_acc,
%                   track.x_det_true_p,track.y_det_true_p,ubody_m

%Texp=exposure time [s] computed in AP
%Texp_tp=exposure time [s] computed at each tracking cycle
%det_p=detection probability at each current cycle,
%                                            for mag 16,17,18,19

%TRACKset=object stars to track, after subset selection:
%         CCD,ID,Mi,ux,uy,uz,rasc_acc,decl_acc,
%             TRACKset.x_det_true_p,TRACKset.y_det_true_p,ubody_m

%Qindex=quality index for TRACKset stars subset
%qabs=computed absolute quaternion
%ang_rate_abs=computed absolute angular rate [rad/s]
%qabs_fused=computed fused absolute quaternion
%ang_rate_abs_fused=computed fused absolute angular rate [rad/s]
%qrel_fused=computed fused relative quaternion
%ang_rate_rel_fused=computed fused relative angular rate [rad/s]
%valid_tracking=1 if an attitude is locked in single mode
%fused=counter for how many fused computations happen
%Nlock=number of tracking cycle of attitude locking (single)
%Nlock_fused=number of tracking cycle of attitude locking (fused)
%phi=first Euler angle resulting from fused determination
%eta=second Euler angle resulting from fused determination
%psi=third Euler angle resulting from fused determination
%errphi_fused,erreta_fused,errpsi_fused=euler angles error
```

```
%                   between measured and reference quaternion
%                                in fused absolute computation [rad]
%errphi_single,erreta_single,errpsi_single=euler angles error
%                      between measured and reference quaternion
%                             in single absolute computation [rad]
%errphi_rel,erreta_rel,errpsi_rel=euler angles error
%                 between current and locked quaternion
%                                in fused relative computation [rad]
%errphi_rel_ref,erreta_rel_ref,errpsi_rel_ref=
%         relative pointing error foreseen by dynamic input [rad]
%Euler_ref=[errphi_rel_ref,erreta_rel_ref,errpsi_rel_ref];
%sigma=standard deviation from the mean value of a vector
%AME_phi_single=Absolute Measurement Error on phi (single)   [as]
%AME_eta_single=Absolute Measurement Error on eta (single)   [as]
%AME_psi_single=Absolute Measurement Error on psi (single)   [as]
%AME_phi_fused=Absolute Measurement Error on phi (fused)     [as]
%AME_eta_fused=Absolute Measurement Error on eta (fused)     [as]
%AME_psi_fused=Absolute Measurement Error on psi (fused)     [as]
%RME_phi=Relative Measurement Error on phi (fused)           [as]
%RME_eta=Relative Measurement Error on eta (fused)           [as]
%RME_psi=Relative Measurement Error on psi (fused)           [as]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%PRE-PROCESSING

clear all
close all
format long

%Parameters acquisition from InputFile
InputFile

%to read OBCF
[n_target,star]=readOBCF(OBCF_filename);

%to read DynamicInputFile
[q,ang_rate,time]=readDynamic(Dynamic_filename);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%VARIABLES INITIALIZATION

%for all the phases
N=Nstart;
```

```
%for acquisition
x_fgs=zeros(4,80);
y_fgs=zeros(4,80);
coor_det=zeros(4,80,2);
x_det=zeros(4,80);
y_det=zeros(4,80);
x_fgs_m=zeros(4,80);
y_fgs_m=zeros(4,80);

%for tracking
qabs=zeros(4,4,(Nend-N-10)/Delta_time+1);
ang_rate_abs=zeros(4,3,(Nend-N-10)/Delta_time+1);
qabs_fused=zeros((Nend-N-10)/Delta_time+1,4);
ang_rate_abs_fused=zeros((Nend-N-10)/Delta_time+1,3);
qrel_fused=zeros((Nend-N-10)/Delta_time+1,4);
ang_rate_rel_fused=zeros((Nend-N-10)/Delta_time+1,3);
valid_tracking=[0 0 0 0];
fused=0;
Nlock=[0 0 0 0];
Nlock_fused =0;
time_t=zeros((Nend-N-10)/Delta_time+1,1);

%for post-processing
phi = zeros(1,length(qrel_fused));
eta = zeros(1,length(qrel_fused));
psi = zeros(1,length(qrel_fused));
errphi_single=zeros(4,(Nend-N-10)/Delta_time+1);
erreta_single=zeros(4,(Nend-N-10)/Delta_time+1);
errpsi_single=zeros(4,(Nend-N-10)/Delta_time+1);
errphi_fused=zeros(1,(Nend-N-10)/Delta_time+1);
erreta_fused=zeros(1,(Nend-N-10)/Delta_time+1);
errpsi_fused=zeros(1,(Nend-N-10)/Delta_time+1);
errphi_rel=zeros(1,(Nend-N-10)/Delta_time+1);
erreta_rel=zeros(1,(Nend-N-10)/Delta_time+1);
errpsi_rel=zeros(1,(Nend-N-10)/Delta_time+1);
AME_phi_single = zeros(4,1);
AME_eta_single = zeros(4,1);
AME_psi_single = zeros(4,1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%START OF ACQUISITION PHASES
```

```
for ap = 1:2

    if ap == 1
        fprintf('\n ACQUISITION PHASE\n\n');
    else
        fprintf('\n INTERMEDIATE CYCLE\n\n');
    end

    %to compute the detection probability according to
    %angular rate at the present cycle: linear interpolation
    %                        based on "det_p0_ap" and "det_p1_ap"
    [det_p] = linear_interpolation...
        (det_p0_ap,det_p1_ap,ref_rate,ang_rate(N,:));

    for n_CCD=CCD
        if n_target(n_CCD)==0
            fprintf(strcat('WARNING: for CCD ', num2str(n_CCD)...
                ,' no stars are present in the catalogue!\n'));

        else
            %catalogue star projection
            [x_fgs(n_CCD,1:size(star(n_CCD,1).ux,2)),...
            y_fgs(n_CCD,1:size(star(n_CCD,1).ux,2)),...
            coor_det(n_CCD,1:size(star(n_CCD,1).ux,2),:),...
            x_det(n_CCD,1:size(star(n_CCD,1).ux,2)),...
            y_det(n_CCD,1:size(star(n_CCD,1).ux,2))]=ALG_0220...
            (q(N,:),star(n_CCD,1),Px0,Px1,Py0,Py1,PIXsize,...
            focal,x0det,y0det,alfa,beta,A,tilt_mat,n_CCD);

             %to include the detection probability:
             %NEA + Catalogue accuracy
             %p is for "probability"
             [detected_temp]=detection...
                 (x_det,y_det,mi,det_p,etaNEA,...
                 star(n_CCD,1),abscissa,n_CCD);

             %to exclude targets outside CCD FOV
             [detected_inCCD_temp]=inCCD(detected_temp);

             %to allocate an array of structures with a-priori
             %                        unknown non-zero dimensions
             detected(n_CCD,1).ccd(1,1:length...
```

```
    (detected_inCCD_temp.id))=detected_inCCD_temp.ccd;
 detected(n_CCD,1).Mi(1,1:length...
     (detected_inCCD_temp.id))=detected_inCCD_temp.Mi;
 detected(n_CCD,1).ux(1,1:length...
     (detected_inCCD_temp.id))=detected_inCCD_temp.ux;
 detected(n_CCD,1).uy(1,1:length...
     (detected_inCCD_temp.id))=detected_inCCD_temp.uy;
 detected(n_CCD,1).uz(1,1:length...
     (detected_inCCD_temp.id))=detected_inCCD_temp.uz;
 detected(n_CCD,1).rasc_acc...
     (1,1:length(detected_inCCD_temp.id))=...
     detected_inCCD_temp.rasc_acc;
 detected(n_CCD,1).decl_acc...
     (1,1:length(detected_inCCD_temp.id))=...
     detected_inCCD_temp.decl_acc;
 detected(n_CCD,1).id...
     (1,1:length(detected_inCCD_temp.id))=...
     detected_inCCD_temp.id;
 detected(n_CCD,1).x_det_true_p...
     (1,1:length(detected_inCCD_temp.id))=...
     detected_inCCD_temp.x_det_true_p;
 detected(n_CCD,1).y_det_true_p...
     (1,1:length(detected_inCCD_temp.id))=...
     detected_inCCD_temp.y_det_true_p;

 if size(find(detected_temp.ccd),2)==0
    fprintf(strcat('WARNING: for CCD ', num2str...
    (n_CCD),' no stars are detected due to NEA,',...
' catalogue accuracy and detection probability!\n'));
 elseif size(find(detected(n_CCD,1).ccd),2)==0
    fprintf(strcat('WARNING: for CCD ', num2str...
    (n_CCD),' no stars are detected because',...
    'they fall outside CCD FOV!\n'));

 elseif size(find(detected_temp.ccd),2)>0 && ...
         size(find(detected(n_CCD,1).ccd),2)>0

    %m is for "measurement"
    %detected targets back to MRF
    [x_fgs_m(n_CCD,1:length...
        (detected(n_CCD,1).x_det_true_p)),...
        y_fgs_m(n_CCD,1:length...
```

```
    (detected(n_CCD,1).x_det_true_p))]=...
ALG_0065(detected(n_CCD,1).x_det_true_p,...
detected(n_CCD,1).y_det_true_p,PIXsize,A_b,...
tilt_mat_b,x0det_b,y0det_b,n_CCD);

%unit vector computation
[detected(n_CCD,1).ubody_m(1,1:length...
    (detected(n_CCD,1).x_det_true_p),1),...
detected(n_CCD,1).ubody_m(1,1:length...
    (detected(n_CCD,1).x_det_true_p),2),...
detected(n_CCD,1).ubody_m(1,1:length...
    (detected(n_CCD,1).x_det_true_p),3)]=...
    ALG_0070(x_fgs_m,y_fgs_m,focal_b,n_CCD);

%ATM-AP target selection
[track_temp,ubody_m_temp]=ALG_4020(Ntrack_max,...
    detected,detected(n_CCD,1).ubody_m,...
    border,qborder,n_CCD);

%to allocate an array of structures
%with a-priori unknown non-zero dimensions
track(n_CCD,1).ccd(1,1:length(track_temp.id))=...
    track_temp.ccd;
track(n_CCD,1).Mi(1,1:length(track_temp.id))=...
    track_temp.Mi;
track(n_CCD,1).ux(1,1:length(track_temp.id))=...
    track_temp.ux;
track(n_CCD,1).uy(1,1:length(track_temp.id))=...
    track_temp.uy;
track(n_CCD,1).uz(1,1:length(track_temp.id))=...
    track_temp.uz;
track(n_CCD,1).rasc_acc(1,1:length...
    (track_temp.id))=track_temp.rasc_acc;
track(n_CCD,1).decl_acc(1,1:length...
    (track_temp.id))=track_temp.decl_acc;
track(n_CCD,1).id(1,1:length(track_temp.id))=...
    track_temp.id;
track(n_CCD,1).x_det_true_p(1,1:length...
    (track_temp.id))=track_temp.x_det_true_p;
track(n_CCD,1).y_det_true_p(1,1:length...
    (track_temp.id))=track_temp.y_det_true_p;
```

```
%to allocate a 3-D vector
%with a-priori unknown non-zero dimensions
track(n_CCD,1).ubody_m(1,1:length...
    (ubody_m_temp(:,1)),:)=ubody_m_temp;

if size(find(track(n_CCD,1).ccd,2))==0
    fprintf=strcat('WARNING: for CCD ', ...
        num2str(n_CCD),'no stars are present',...
        ' after border check\n');
else
    %attitude determination
    [qabs_ap(n_CCD,:)]=...
        ALG_0010(track(n_CCD,1).ubody_m,...
        track(n_CCD,1).ux,track(n_CCD,1).uy,...
        track(n_CCD,1).uz,q(N,:));

    %exposure time adjustment
    [Texp(n_CCD)]=ALG_0075(Texp_max,Texp_min,...
        Texp_step,S0,Speak,track(n_CCD,1).Mi,...
        THsat,THexp_min,DeltaSmax,ang_rate(N,:));

    %target subset selection
    [TRACKset_temp,Qindex(n_CCD),ubody_m_temp]...
        =ALG_0025(Texp,etaNEA,NSmin,NSmax,...
        track,track(n_CCD,1).ubody_m,n_CCD);

    %to allocate an array of structures
    %with a-priori unknown non-zero dimensions
    TRACKset(n_CCD,1).ccd(1,1:length...
        (TRACKset_temp.id))=TRACKset_temp.ccd;
    TRACKset(n_CCD,1).Mi(1,1:length...
        (TRACKset_temp.id))=TRACKset_temp.Mi;
    TRACKset(n_CCD,1).ux(1,1:length...
        (TRACKset_temp.id))=TRACKset_temp.ux;
    TRACKset(n_CCD,1).uy(1,1:length...
        (TRACKset_temp.id))=TRACKset_temp.uy;
    TRACKset(n_CCD,1).uz(1,1:length...
        (TRACKset_temp.id))=TRACKset_temp.uz;
    TRACKset(n_CCD,1).rasc_acc(1,1:length...
        (TRACKset_temp.id))=...
        TRACKset_temp.rasc_acc;
    TRACKset(n_CCD,1).decl_acc(1,1:length...
```

```
                    (TRACKset_temp.id))=...
                    TRACKset_temp.decl_acc;
                TRACKset(n_CCD,1).id(1,1:length...
                    (TRACKset_temp.id))=TRACKset_temp.id;
                TRACKset(n_CCD,1).x_det_true_p(1,...
                    1:length(TRACKset_temp.id))=...
                    TRACKset_temp.x_det_true_p;
                TRACKset(n_CCD,1).y_det_true_p(1,...
                    1:length(TRACKset_temp.id))=...
                    TRACKset_temp.y_det_true_p;

                %to allocate a 3-D vector
                %with a-priori unknown non-zero dimensions
                TRACKset(n_CCD,1).ubody_m(1,1:length...
                    (ubody_m_temp(1,:,1)),:)=ubody_m_temp;

                if size(find(TRACKset(n_CCD,1).ccd),2)==0
                    fprintf(strcat('WARNING: In CCD ',...
                        num2str(n_CCD),...
                        ' no stars are acquired!\n'));
                elseif size(find...
                        (TRACKset(n_CCD,1).ccd),2)<NSmin
                    fprintf(strcat('WARNING: In CCD ',...
                        num2str(n_CCD),' less than ',...
                        num2str(NSmin) ,...
                        ' stars are acquired!\n'));
                end
            end
        end
    end
end

if ap == 1
    N = N+6;                        %6 seconds for AP duration
    save(strcat(output_folder,'\AP_results'),...
        'qabs_ap','TRACKset','detected','Texp');
else
    N = N+4;                        %4 seconds for IC duration
    save(strcat(output_folder,'\IC_results'),...
        'qabs_ap','TRACKset','detected','Texp');
end
```

```
end

%END OF ACQUISITION PHASES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%START OF TRACKING PHASE

fprintf('\n TRACKING PHASE\n\n');

%to verify that at least 1 tracking cycle can be performed
%before SBM
if (Nend-N)<0
    tc_end = 0;
    fprintf('Tracking phase is not performed,',...
        ' since FGS is commanded to SBM!\n');
else
    tc_end = (Nend-N)/Delta_time+1;

    for tc=1:tc_end                              %index for cycles

        fprintf(strcat('tracking cycle=',num2str(tc),'\n'));

        %time of current tracking cycle
        time_t(tc)=time(tc.*Delta_time-1+Nstart+9);

        %to compute the detection probability according to
        %angular rate at the present cycle: linear interpolation
        %based on "det_p0_tp" and "det_p1_tp"
        [det_p] = linear_interpolation...
            (det_p0_tp,det_p1_tp,ref_rate,ang_rate(N,:));

        %initialization at each new cycle
        x_fgs_t=zeros(4,10);
        y_fgs_t=zeros(4,10);
        coor_det_t=zeros(4,10,2);
        x_det_t=zeros(4,10);
        y_det_t=zeros(4,10);
        x_fgs_m_t=zeros(4,10);
        y_fgs_m_t=zeros(4,10);

        for n_CCD=CCD
            if size(find(TRACKset(n_CCD,1).ccd),2)==0 && tc==1
```

```
    fprintf(strcat('WARNING: for CCD ', ...
        num2str(n_CCD),...
        ' no tracking stars are present!\n'));

elseif size(find(TRACKset(n_CCD,1).ccd),2)<NSmin ...
        && tc==1
    fprintf(strcat('WARNING: for CCD ', ...
    num2str(n_CCD),' there are less than 3',...
  ' stars and the pattern cannot be validated!\n'));

elseif size(find(TRACKset(n_CCD,...
        1).ubody_m(1,:,1)),2)>=NSmin  && tc>=1
    %tracking star projection
    [x_fgs_t(n_CCD,1:length...
    (TRACKset(n_CCD,1).ux(1,:))),...
    y_fgs_t(n_CCD,1:length...
    (TRACKset(n_CCD,1).ux(1,:))),...
    coor_det_t(n_CCD,1:length...
    (TRACKset(n_CCD,1).ux(1,:)),:),x_det_t(n_CCD,...
    1:length(TRACKset(n_CCD,1).ux(1,:))),...
    y_det_t(n_CCD,1:length...
    (TRACKset(n_CCD,1).ux(1,:)))]=ALG_0220(q(N,:)...
    ,TRACKset(n_CCD,1),Px0,Px1,Py0,Py1,PIXsize,...
    focal,x0det,y0det,alfa,beta,A,tilt_mat,n_CCD);

    %to include the detection probability
    %NEA + Catalogue accuracy
    %p is for "probability"
    [detected_temp]=detection(x_det_t,y_det_t,mi,...
    det_p,etaNEA,TRACKset(n_CCD,1),abscissa,n_CCD);

    %to exclude targets outside CCD FOV
    [detected_inCCD_temp]=inCCD(detected_temp);

    %to exclude targets falling
    %in quadrant intersections
    [detected_inCCD_temp]=ALG_4020_tracking...
    (detected_inCCD_temp,qborder);

    %to allocate an array of structures
    %with a-priori unknown non-zero dimensions
    detectedt(n_CCD,1,tc).ccd(1,...
```

```
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.ccd;
detectedt(n_CCD,1,tc).Mi(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.Mi;
detectedt(n_CCD,1,tc).ux(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.ux;
detectedt(n_CCD,1,tc).uy(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.uy;
detectedt(n_CCD,1,tc).uz(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.uz;
detectedt(n_CCD,1,tc).rasc_acc(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.rasc_acc;
detectedt(n_CCD,1,tc).decl_acc(1,...
    1:length(detected_inCCD_temp.id))=...
detected_inCCD_temp.decl_acc;
detectedt(n_CCD,1,tc).id(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.id;
detectedt(n_CCD,1,tc).x_det_true_p(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.x_det_true_p;
detectedt(n_CCD,1,tc).y_det_true_p(1,...
    1:length(detected_inCCD_temp.id))=...
    detected_inCCD_temp.y_det_true_p;

%m is for "measurement"
%detected targets back to MRF
[x_fgs_m_t(n_CCD,1:length...
(detectedt(n_CCD,1,tc).x_det_true_p)),...
y_fgs_m_t(n_CCD,1:length...
(detectedt(n_CCD,1,tc).x_det_true_p))]=...
ALG_0065(detectedt(n_CCD,1,tc).x_det_true_p,...
detectedt(n_CCD,1,tc).y_det_true_p,PIXsize,...
A_b,tilt_mat_b,x0det_b,y0det_b,n_CCD);

%unit vector computation
[detectedt(n_CCD,1,tc).ubody_m(1,1:length...
```

```
    (detectedt(n_CCD,1,tc).x_det_true_p),1),...
detectedt(n_CCD,1,tc).ubody_m(1,1:length...
    (detectedt(n_CCD,1,tc).x_det_true_p),2),...
detectedt(n_CCD,1,tc).ubody_m(1,1:length...
(detectedt(n_CCD,1,tc).x_det_true_p),3)]=...
ALG_0070(x_fgs_m_t,y_fgs_m_t,focal_b,n_CCD);

%to not detect a specified star
%from cycle0 to cyclef
[detectedt(n_CCD,1,tc)]=missingstar...
    (disappear(1),disappear(2),disappear(3),...
    tc,detectedt(n_CCD,1,tc));

if size(find(detected_temp.ccd),2)==0
   fprintf(strcat('WARNING: for CCD ', ...
   num2str(n_CCD),...
   ' no stars are detected because of NEA,',...
   ' catalogue accuracy and',...
   ' detection probability!',...
   ' - Tracking cycle=',num2str(tc),'\n'));
elseif size(find(detected_temp.ccd),2)<NSmin
   fprintf(strcat('WARNING: for CCD ',...
   num2str(n_CCD),' less than 3 stars',...
   ' are detected because of NEA,',...
   ' catalogue accuracy and',...
   ' detection probability!',...
   ' - Tracking cycle=',num2str(tc),'\n'));
elseif size(find(detected_inCCD_temp.ccd),2)==0
   fprintf(strcat('WARNING: for CCD ',...
   num2str(n_CCD),' no stars are detected',...
   ' because they fall outside CCD FOV!',...
   '- Tracking cycle=',num2str(tc),'\n'));
elseif size(find...
        (detected_inCCD_temp.ccd),2)<NSmin
   fprintf(strcat('WARNING: for CCD ', ...
   num2str(n_CCD),' less than 3 stars are',...
   ' detected because they fall ',...
   'outside CCD FOV!','- Tracking cycle=',...
   num2str(tc),'\n'));
elseif size(find...
        (detectedt(n_CCD,1,tc).ccd),2)<NSmin
   fprintf(strcat('WARNING: for CCD ', ...
```

```
            num2str(n_CCD),' there are less than',...
            ' 3 stars and the pattern cannot be ',...
            'validated!\n',' A target is ',...
            'disappeared in Tracking cycle=',...
            num2str(tc),'\n\n'));

        elseif size(find...
                (detectedt(n_CCD,1,tc).ccd),2)>=NSmin
            %%%%SINGLE CCD COMPUTATION%%%%%%%%%%%%%%%%%%
            if valid_tracking(n_CCD)==0
              %ATM absolute attitude measurement (single)
              [qabs(n_CCD,1:4,tc),...
              ang_rate_abs(n_CCD,1:3,tc)]=...
              ALG_0041(detectedt(n_CCD,1,tc),...
              detectedt(n_CCD,1,tc).ubody_m,qabs...
              (n_CCD,1:4,tc),Delta_time);

              %ATTITUDE LOCKING (single)
              valid_tracking(n_CCD)=1;
              TRACKset(n_CCD,1)=detectedt(n_CCD,1,tc);
              Nlock(n_CCD)=tc;
            else
              %ATM absolute attitude measurement (single)
              [qabs(n_CCD,1:4,tc),...
              ang_rate_abs(n_CCD,1:3,tc)]=...
              ALG_0041(detectedt(n_CCD,1,tc),...
              detectedt(n_CCD,1,tc).ubody_m,qabs...
              (n_CCD,1:4,tc),Delta_time);
            end
        end
    end

    %exposure time adjustment
    [Texp_tp(n_CCD,tc)]=ALG_0075(Texp_max,Texp_min,...
        Texp_step,S0,Speak,TRACKset(n_CCD,1).Mi,...
        THsat,THexp_min,DeltaSmax,ang_rate(N,:));
end

%%%%FUSED CCDs COMPUTATION%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%to fuse attitude sets of two CCDs
[FUSEDset(1,tc)]=fuse(detectedt,CCD,tc);
```

```
    if size(find(FUSEDset(1,tc).ccd),2)==0
              fprintf(strcat('WARNING: for CCDs: ', ...
              num2str(CCD),' there are no targets',...
              ' and fused computation cannot be performed'));
    elseif size(find(FUSEDset(1,tc).ccd),2)<NSmin
              fprintf(strcat('WARNING: for CCDs: ', ...
              num2str(CCD),' there are less than',...
              ' 3 targets and fused computation',...
              ' cannot be performed'));
    elseif size(find(FUSEDset(1,tc).ccd),2)>=NSmin
        %ATTITUDE LOCKING (fused)
        fused = fused + 1;
        if fused == 1
            Nlock_fused = tc;
            %if single mode did not succeed
            if valid_tracking == [0 0 0 0]
                for n_CCD = CCD
                   TRACKset(n_CCD,1) = detectedt(n_CCD,1,tc);
                end
            end
        end

        %ATM absolute attitude measurement (fused)
        [qabs_fused(tc,1:4),...
        ang_rate_abs_fused(tc,1:3)]=...
        ALG_0041(FUSEDset(1,tc),FUSEDset(1,tc).ubody_m,...
        qabs_fused(tc,1:4),Delta_time);

        if tc>=2
            %ATM relative attitude measurement (fused)
            [qrel_fused(tc,1:4),...
            ang_rate_rel_fused(tc,1:3)]=ALG_0040...
            (FUSEDset(1,Nlock_fused),FUSEDset(1,tc),...
            qrel_fused(tc-1,1:4),Delta_time);
        end
    end

    %next dynamic input (Delta_time seconds later)
    N = N+Delta_time;
    fprintf('\n')
end
```

```
    save(strcat(output_folder,'\TP_results'),'qabs',...
    'ang_rate_abs','qabs_fused','ang_rate_abs_fused',...
    'qrel_fused','ang_rate_rel_fused','time_t','detectedt',...
    'TRACKset','Nlock', 'Nlock_fused','Texp_tp');

end

%END OF TRACKING PHASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%POST-PROCESSING

%If TP is executed
if tc_end~=0
    %Euler angles from fused relative quaternion
    for i=max(Nlock_fused,2):length(qrel_fused(:,1))
          [phi(i),eta(i),psi(i)]=...
              quat2euler(qrel_fused(i,:));                %[rad]
          phi(i) = phi(i)*180/pi*3600;                     %[as]
          eta(i) = eta(i)*180/pi*3600;                     %[as]
          psi(i) = psi(i)*180/pi*3600;                     %[as]
    end

    %to plot Euler angles
    figure()
    title('Euler angles during the whole simulation')
    hold on
    plot(phi)
    plot(eta)
    plot(psi)
    legend('phi','eta','psi')
    xlabel('tracking cycle')
    ylabel('Euler angles [arcsec] from q_{fused}')

    %to compute measurement errors during the whole simulation
    for tc=1:tc_end
        N=(Delta_time*tc-1)+(Nstart+9);

        for n_CCD=CCD
            %to compute error in Euler angles between measured
            %quaternion and dynamic input (single mode)
           [errphi_single(n_CCD,tc),erreta_single(n_CCD,tc),...
```

```
        errpsi_single(n_CCD,tc)]=att_err_comp(squeeze...
        (qabs(n_CCD,1:4,tc)),q(N,:));
    end

    %to compute error in Euler angles between measured
    %quaternion and dynamic input (fused mode)
    [errphi_fused(tc),erreta_fused(tc),errpsi_fused(tc)]=...
    att_err_comp(qabs_fused(tc,:),q(N,:));

    %to compute the relative pointing error between
    %dynamic input at current cycle and at time of locking
    [errphi_rel_ref(tc),erreta_rel_ref(tc),...
    errpsi_rel_ref(tc)] = att_err_comp(q(N,:),...
    q(Nlock_fused+Nstart+9,:));

    Euler_ref(tc,:)=[errphi_rel_ref(tc),...
    erreta_rel_ref(tc),errpsi_rel_ref(tc)];

    %to pass from Euler angles to quaternion
    [qrel_fused_ref(tc,:)]=Euler_Quat(Euler_ref(tc,:));

    %to compute measurement error in Euler angles between
    %current and locking quaternion (fused mode)
    [errphi_rel(tc),erreta_rel(tc),errpsi_rel(tc)]=...
    att_err_comp(qrel_fused(tc,:),qrel_fused_ref(tc,:));
end

%for RTM simulation
if (command == 'rtm') == logical([1 1 1])
    %RME on phi
    sigma = std(errphi_rel(3:end).*180/pi*3600);
    RME_phi = 3*sigma;                                    %[as]

    %RME on eta
    sigma = std(erreta_rel(3:end).*180/pi*3600);
    RME_eta = 3*sigma;                                    %[as]

    %RME on psi
    sigma = std(errpsi_rel(3:end).*180/pi*3600);
    RME_psi = 3*sigma;                                    %[as]

%for ATM simulation
```

```
    else if (command == 'atm') == logical([1 1 1])
        %RME on phi
        sigma = std(errphi_fused.*180/pi*3600);
        RME_phi = 3*sigma;                                      %[as]

        %RME on eta
        sigma = std(erreta_fused.*180/pi*3600);
        RME_eta = 3*sigma;                                      %[as]

        %RME on psi
        sigma = std(errpsi_fused.*180/pi*3600);
        RME_psi = 3*sigma;                                      %[as]

        end
    end

    %AME (fused) on phi
    AME_phi_fused = abs(sum(errphi_fused.*180/pi*3600)...
        /length(errphi_fused));                                 %[as]

    %AME (fused) on eta
    AME_eta_fused = abs(sum(erreta_fused.*180/pi*3600)...
        /length(erreta_fused));                                 %[as]

    %AME (fused) on psi
    AME_psi_fused = abs(sum(errpsi_fused.*180/pi*3600)...
        /length(errpsi_fused));                                 %[as]

    for n_CCD = 1:4
        %AME (single) on phi
        AME_phi_single(n_CCD) = abs(sum(errphi_single(n_CCD,:)...
            .*180/pi*3600)/length(errphi_single(n_CCD,:)));  %[as]

        %AME (single) on eta
        AME_eta_single(n_CCD) = abs(sum(erreta_single(n_CCD,:)...
            .*180/pi*3600)/length(erreta_single(n_CCD,:)));  %[as]

        %AME (single) on psi
        AME_psi_single(n_CCD) = abs(sum(errpsi_single(n_CCD,:)...
            .*180/pi*3600)/length(errpsi_single(n_CCD,:)));  %[as]
    end
end
```

```
save(strcat(output_folder,'\performances'),...
    'phi','eta','psi',...
    'errphi_single','erreta_single','errpsi_single',...
    'errphi_fused','erreta_fused','errpsi_fused',...
    'errphi_rel','erreta_rel','errpsi_rel',...
    'RME_phi','RME_eta','RME_psi',...
    'AME_phi_single','AME_eta_single','AME_psi_single',...
    'AME_phi_fused','AME_eta_fused','AME_psi_fused');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Bibliography

[1] ESA/SRE(2011)12, Euclid: Mapping the geometry of the dark Universe - Definition Study Report, ESA/ESTEC, September 2011 (Revision 1);

[2] Euclid Space Mission: building the sky survey, Statistical Challenges in 21st century Cosmology Proceedings IAU Symposium No. 306, 2014 A.C. Editor, B.D. Editor & C.E. Editor, eds.;

[3] The Euclid mission design - G. Racca, L. Stagnaro, R. J. Laureijs, L.M.G. Venancio and others, October 2016;

[4] Euclid pointing performance: operations for the Fine Guidance Sensor reference star catalogue - A. Bosco, A. Bacchetta, M. Saponara, G. Saavedra Criado, SpaceOps Marseille conference, June 2018;

[5] Spacecraft dynamics and control - E. Canuto, C. Novara, L. Massotti, D. Carlucci, C. P. Montenegro, Elsevier, 2018;

[6] ECSS-E-ST-60-20C-Rev.1, Space engineering: stars sensors terminology and performance specification, ESA/ESTEC, November 2008;

[7] Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere - K.M. Górski, Hivon and others, The Astrophysical Journal, 2005;

[8] Stars search inside FOV, Master Thesis in Thales Alenia Space - B. Catarino, April 2017;

[9] Confocal microscopy - H. Zhang, J. Liu, W. Wang, J. Tan, Morgan & Claypool publishers, December 2016;

[10] http://sci.esa.int;

[11] https://resonance.is/new-flaw-dark-energy-theory;

[12] http://www.hatiandskoll.com;

[13] http://www.mssl.ucl.ac.uk.