Delft University of Technology

Development and Verification of the Electro-Optical and Software Modules of the Facet Nano Star Sensor

by

Emilio Fraile García

Student number: 4118545

A thesis submitted in partial fulfillment for the degree of Master of Science in the

Chair of Space Systems Engineering
Faculty of Aerospace Engineering

September 2014

Declaration of Authorship

I, Emilio Fraile García, declare that this thesis titled, 'Development and Verification of the Electro-Optical and Software Modules of the Facet Nano Star Sensor' and the work presented in it are my own. I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other Dutch or foreign examination board. The thesis work was conducted from November 2011 to July 2012 under the supervision of Hans Kuiper at TU Delft and Anselm Ho and Eddie van Breukelen at Innovative Solutions In Space.

In Delft,

Signed: Emilio Fraile García

Date: 25th September 2014

"If I have seen further it is by standing on ye sholders of Giants" [sic]

Isaac Newton in a letter to his rival Robert Hooke dated February 5, 1676

Delft University of Technology

Abstract

Faculty of Aerospace Engineering
Chair of Space Systems Engineering

Master of Science

by Emilio Fraile García

This Thesis report is written for the space community, specially for people interested in the field of nanosatellites, who know how star trackers work, but are unfamiliar with their operations and limitations in an earth observation mission. To introduce users to the topic, this document is primarily concerned with a single aperture star tracker in order to simplify the explanation.

The general mission parameters are described and the critical requirements are flowed down to the Attitude Control System and subsequently the star tracker. Various influences on star tracker performance and availability are described, and different subsystems are analyzed in terms of system performance: optics, baffle, image sensor and on-board software. In addition, the verification process was present from the very beginning, including a verification plan and the calibration and validation activities to be performed on ground in the lab and during the commissioning phase.

It is shown that a single aperture star tracker is well suited for a very limited set of operating conditions, namely a strictly nadir-pointing earth observation mission, however the flexibility and robustness of the multiple-aperture star tracker is generally required for missions with realistic operational requirements.

Acknowledgements

My sincere thanks to Eddie van Breukelen for offering me the opportunity to join ISIS and for his wise advice and managing. A special thanks to my thesis supervisor, J.M. Hans Kuiper, whose recommendations marked key points in the developing of my thesis. This work would not be possible without Anselm Ho and all my colleagues at ISIS, who were always willing to bear a hand. I am also grateful to my friends for our good moments that made me feel at home. A thousand thanks to my parents, who always encouraged me to make it one step further. Finally, thanks to Eva, whose unconditional support provided the warmest comfort even in the coldest and cloudiest days.

Contents

List of Figures

List of Tables

Abbreviations

ADC	Analog to Digital (A/D) Converter
AIT	Assembly, Integration and Test
AIV	Assembly, Integration and Verification
ADCS	Attitude Determination and Control System
APS	Active Pixel Sensor
CAL/VAL	Calibration and Validation
CCD	Charge-Coupled Device
CMOS	Complementary MetalOxide Semiconductor
COTS	Commercial Off The Shelf
DN	Digital Number
DSNU	Dark Signal Non-Uniformity
EOS	Electro-Optical System
FF	Fill Factor
FN	Facet Nano
FPGA	Field Programmable Gate Array
FPN	Fixed Pattern Noise
Hipparcos	High Precision Parallax Collecting Satellite
I2C	Inter Integrated Circuit
MTF	Modulation Transfer Function
MSc	Master of Science
ND	Neutral Density (Filter)
NEA	Noise Equivalent Angle
NIR	Near Infra-Red
OBC	On-Board Computer
OTF	Optical Transfer Function

PCB	Printed Circuit Board
PRNU	Photo Response Non-Uniformity
PSF	Point Spread Function
QE	Quantum Efficiency
RMS	Root Mean Square
RPY	Roll, Pitch and Yaw angles
SAA	South-Atlantic Anomaly
SEU	Single Event Upset
SNR	Signal-to-Noise Ratio
TBC	To Be Confirmed
TBD	To Be Determined
TDI	Time Delay Integration
TID	Total Ionizing Dose
USB	Universal Serial Bus
UUT	Unit Under Test
VNIR	Visible and Near Infra-Red

Physical Constants

Speed of Light	c	=	$2.997\ 924\ 58\ \times 10^{8}\ ms^{-1}$
Elemental Charge	e	=	$1.602\ 177\ 33(49)\ \times 10^{-19}\ C$
Planck's Constant	h	=	$6.626\ 176\ \times 10^{-34}\ Js$
Stephan-Boltmann's Constant	σ	=	$5.670\ 51(19)\ \times 10^{-8}\ Wm^{-2}K^{-4}$
Gravitation constant	G	=	$6.672\ 55\ \times 10^{-11}\ m^{3}kg^{-1}s{-}2$
Boltzmann constant	k	=	$8.617\ 385\ \times 10^{-5}\ eV/K$
Charge of electron	e	=	$1.60217657\ \times 10^{-19}\ C$
Permeability of vacuum	μ_0	=	$4\ \pi \times 10^{-7}\ Vs/(Am)$

Dedicated to my family.

Chapter 1

Facet Nano Project

1.1 Introduction

Small satellites in the range of Picosatellites and Nanosatellites are the cost effective solution to investigate certain applications. These satellites offer the platform to develop and test COTS based solutions. One of the most limiting factors of applications for nanosatellites is their relatively poor attitude determination and control capability: poor both in terms of accuracy and rate of success.

Facet Nano is a multi-aperture star tracker developed by ISIS, a company leader in nanosatellite technology, based in Delft. Facet Nano will contribute to achieve a better accuracy and rate of success than the common attitude determination systems used on nanosatellites without star trackers.

1.1.1 State of the Art

Current ADCS systems for nanosatellites include sun sensors, gyroscopes and magnetometers. Magnetometers measure the intensity and direction of the Earth's magnetic field. This sensor has a medium accuracy of 1 deg, however its main disadvantage is that it can work only at LEO (Low Earth Orbits) because the magnetic field weakens with increasing altitude [1].

Analog sun sensors are the most common used sensors; however, they are extremely inaccurate: 0.3deg of accuracy in FOV (Field Of View) of 30 deg. They need direct sunlight and so they are installed at the panels of the satellite instead of several solar cells. In our case this is a critical problem because fewer solar cells means less power

provided to the satellite [2]. Besides they do not work on eclipse, reducing their rate of success [3].

Star trackers provide numerous advantages over other attitude sensors because of their ability to provide full, three-axis orientation information with high accuracy and flexibility to operate independently from other navigation tools. However, current star trackers are optimized to maximize accuracy, at the exclusion of all else. Although this produces extremely capable systems, the excessive mass, power consumption, and cost that result are often contradictory to the requirements of smaller space vehicles.

Star trackers image a part of the sky and, by comparing a map of the sky to the picture, it can determine its orientation relatively to the stars. They have the higher accuracy of all AD systems but their size and weight are too large for nanosatellites [4]. Thus a question arises, is it possible to design smaller and lower cost star trackers that can provide adequate attitude and rate determination to small, highly maneuverable, low-cost spacecraft?

1.2 Research Framework: Facet Nano System Overview

Facet Nano is envisioned as a multi-aperture star tracker functioning as a high accuracy attitude sensor. This sensor is to be used on nano satellites and should have an accuracy of at least an order of magnitude better than any current nano satellite attitude sensors. It also has to be cost effective by utilizing COTS components. The superiority of a multiple aperture design is an increase in the availability of the system. This should allow a longer operating time for a payload as well an increase the degree of accuracy of the system. This leads to higher costs of a more complex system and slightly added hardware mass.

This section describes the state of the system development at the beginning of the MSc thesis in November 2011.

1.2.1 Optics

A custom-designed lens system was rebuilt with Commercial Off The Shelf (COTS) lenses, due to the high production costs of custom-designed lenses. The fundamental idea is to use lenses with the same aperture diameter and focal length as the custom-designed lenses. The difference in focal length of the COTS lens system is corrected with the distance between the lens surfaces for the VNIR wavelength range.

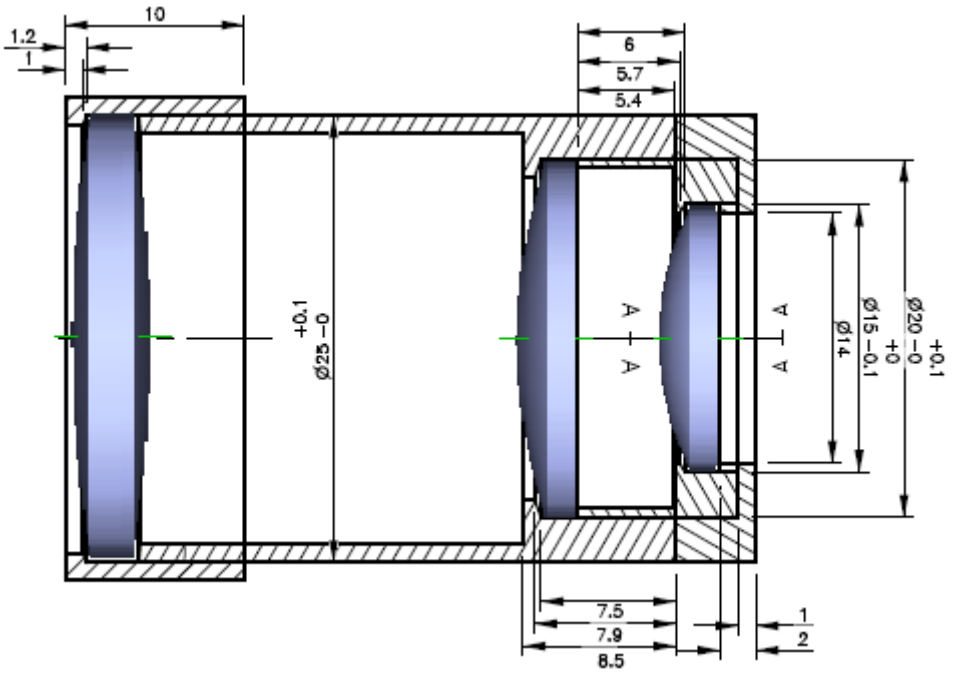

Figure 1.1: Sketch of lens mount

The current optical design for the image sensor gives the best performance possible with a 3 element design. For the development of a Flight Model, an iteration of the optical design with the final material is required. The current design is based on the radiation resistant BK7G18 glass which is expensive and has severe lead times. A proper trade-off and material research was performed to find a material class that would just satisfy the requirements. Fused Silica glass emerged as the best choice to go with. It has good environmental performance like the mirror and radiation resistance glass but is more readily available and cheaper. It completely eliminates the survival risk of the optical elements from the degradation effects of TID.

1.2.2 Image Sensor

LUPA 300 was the first CMOS image sensor selected for the Facet Nano star tracker. This selection was made based on the specifications found in the data sheet. Nevertheless, when assessing the real performance of the sensor, the expectations were not met and the LUPA 300 turned out to be a useless sensor for a star camera system: signal-to-noise ratio and detector sensitivity were below specifications.

A pre-selection of candidate sensors was done based on data-sheet specifications. Active Pixel Sensors (APS) are CMOS based image sensors that incorporate at each pixel an amplifier circuit. The amplifier act as a charge amplifier that converts the photo-generated charge to a voltage as to improve the image characteristics. The active pixel

sensors are replacing gradually CCD based Spacecraft sensors. CCDs have some shortcomings that are: high power consumption, expensive and bulky due to the need for three to eight extra peripheral chips. The larger a pixel is the more photons it can collect and the larger the lens will be. The APS requires 1% of the power of an equivalent CCD system and occupies 10% of the size of a CCD technology based system. This means a reduction in weight and cost compared to the CCD based technology for the same sensitivity. The APS is also less susceptible to radiation exposure; measurements performed on test structures at the University of Delft have shown that the APS imagers are very radiation tolerant [5] [6]. Therefore APS CMOS technology was preferred over CCD technology because of its lower power consumption and its radiation hardening.

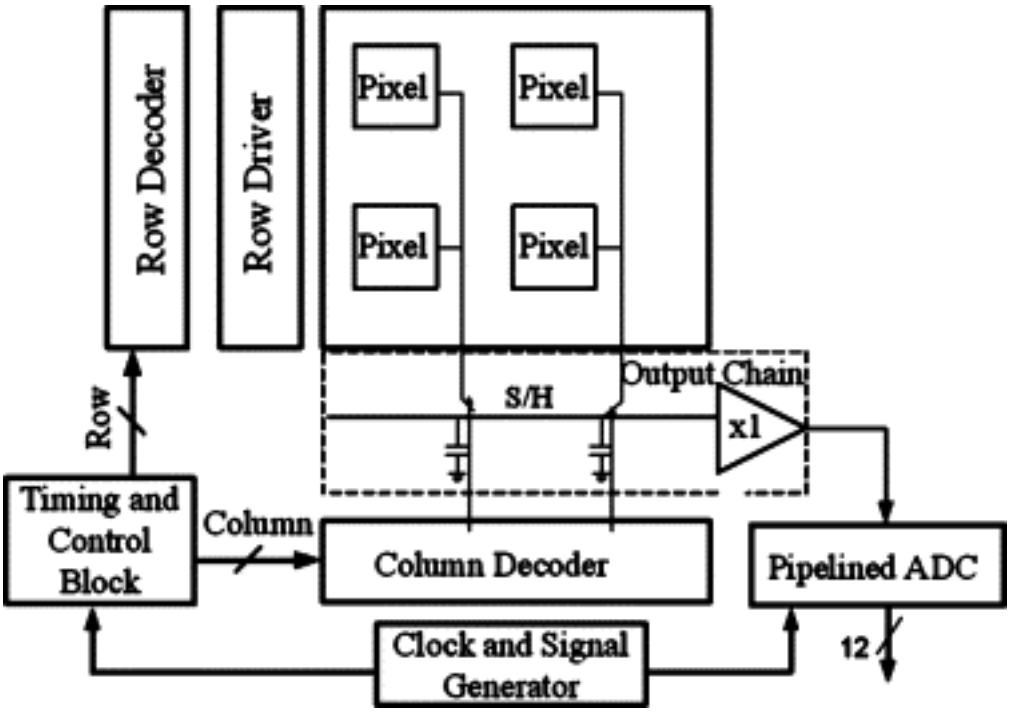

Figure 1.2: Active Pixel Sensor (architecture)[7]

Figure 1.2 illustrates the APS architecture. Each pixel of an APS can be addressed individually. So also certain areas can be addressed and read out. This means that the image of a faint star in a certain area and the image of a bright star in an other area in principle can be read out separately and at different sample rates. The result is that star trackers based on an APS can have a very high dynamic range of observation. Therefore a very bright star and a very faint star can be both detected and tracked at the same time. Besides, once an initial attitude is estimated, only the information from the pixels adjacent to the detected stars is processed in order to evaluate the movement of the stars over the APS sensor to estimate the new attitude and the angular speed. Hence, the computational time can be drastically reduced by analyzing a decimated number of pixels.

Each candidate sensor was evaluated under thermal and electro-optical tests. The purpose of the thermal tests was to characterize the thermal behaviour of the sensor and the dark current as the major noise source of the sensor. The electro-optical test was performed in order to measure the signal-to-noise ratio of the sensor and characterize other electro-optical parameters.

Table 1.1: Specifications of the preselected image sensor

Specification of the selected image sensor	
Shutter type	Global
Array size	838x640
Active array size	838x640
Pixel size (um)	5.8
Shortest side length (mm)	10
Operating temperature (Celc)	-30 to +65
Power consumption (mW)	80
ADC resolution (bit)	8
Supply voltage (V)	3.3 and 1.8
Full well capacity (kilo electrons)	10
Dynamic range (dB)	51
Responsivity (V/lux.sec)	17
Dark current (LSB/s)	8
PRNU (% rms)	0.8
DSNU (% rms)	6
SNR (dB)	40

1.2.3 Read-out Electronics

The latest electrical design update on Facet Nano was carried out in July 2009: a system architecture description was provided and a trade-off among components was done.

The system was composed of one Motherboard and 5 secondary boards (Camera board). Since the cameras had different orientations, it was necessary to have one rigid board per CMOS sensor which are linked to the Motherboard with a flexible link (a Flex PCB).

The star tracker communicated to the outside with an I2C bus. Some timing signals were also provided for having precise timing control and synchronization. The main components were:

- Cypress LUPA-300 CMOS imager
- Atmel AT91SAM9G20 microcontroller (32-bit ARM processor) as the processor
- FPGA is an Actel ProASIC3L, suitable for space applications

- 1.8V memories, in order to reduce power consumption
  - Mobile SDRAM volatile memory to store images
  - Numonix M58WR016K non-volatile memories to store the software and the star catalog.
- Different separated power supplies:
  - 2.5V digital for the cameras, the microcontroller and FPGA I/O bank.
  - 2.5V analog very low noise for the cameras (and the microcontroller ADC).
  - 3.3V analog very low noise for the cameras pixel reset.
  - (2.0V and 2.8V very low noise for the cameras pixel reset dual slope level if used)
  - 1.2V to 1.5V digital for the FPGA core
  - 1.0V digital for the microcontroller core.
  - 1.8V digital for the external memories and the microcontroller or FPGA memory interface I/O bank.

Although this architecture could be easily adapted for the selected sensor, two and a half years later it is likely outdated. A low cost alternative is to use the electrical board provided in the manufacturer's demo kit instead. It was used during previous thermal and optical tests. In these tests, it was demonstrated that the electronics did not add too much noise during the read-out process.

The criteria to opt for the demo-kit board are the project time-constraint, the unavailability of a pcb-layout engineer at the company and the positive performance of the board during the sensor tests. Besides, the software to run with the demo-kit is provided and it allows implementation of several filtering and noise reduction algorithms.

On the other hand, the use of the manufacturer board presents other limitations: maximum integration time of 100ms, difficulty to obtain the board data sheet and ignorance on the component performance used in the board. Moreover, even if the demo-kit meets the requirements for the final prototype, it is subjected to another company's intellectual property. Hence, designing an electrical interface will have to be addressed in a future phase of the project.

1.2.4 Software

Algorithms for the centroiding, image processing, star identification and attitude determination were already designed. The image processing and star detection algorithms

were developed inside the ISIS company and based on the most basic principles of their application. In addition many modules were designed to fit the old sensor.

The basic and limited design of the algorithms lead to a poor performance. Because of this, they fail to detect stars accurately and provide many false positives, i.e., are not robust against the system noise sources. Thus and update and enhancement is needed in order to adapt the algorithm to a new sensor and to meet the performance requirements.

The star identification algorithm was created by a 3rd party, Delta Utec. It makes use of the latest Hipparcos star catalog. An interface between the detection and the identification algorithms needs to be designed, coded and tested as well.

1.3 Research Questions, Aims and Objectives

This study aims at the development of the electrical and software component of a star sensor in order to provide small satellites with attitude data more frequently and more accurately than current state of the art attitude systems for nanosatellites.

The first research objective is therefore the characterization of the parameters involved in the performance of the Facet Nano system and evaluate their impact on the availability and the accuracy of the star tracker. The second research objective is to select an image sensor and to develop software algorithms capable of detecting stars to interface between the selected sensor and the attitude determination subsystem.

The main research questions that will be addressed during this project can be enunciated as:

Can the Facet Nano system provide attitude data with higher availability and accuracy than current attitude sensors for nanosatellites?

The search for an answer to this question divides in into several research sub questions:

i. *What is the algorithm to detect stars and cope with noise that maximizes system performance?*

ii. *Are the sensor and the algorithms tolerant to radiation effects?*

iii. *What are the limiting operational temperatures for the FN star camera?*

iv. *What is the optimal integration time for best performance of the FN star camera?*

v. *What are the noise sources that affect the system? Are there other distortion effects?*

vi. *How can the FN camera be calibrated?*

1.4 Methodology

The first part of this study comprehends the development of a star sensor. This will be first addressed by selecting an image sensor based on data sheet specifications, simulations and electro-optical testing. Secondly, a software algorithm will be designed and coded that uses the sensor output as raw data to detect stars.

The second part includes the verification and validation of the developed systems. Verification is the proof of compliance with design solution specifications and descriptive documents [8]. It is also the methodical development of evidence of system of interest compliance with requirements ("shalls"). It is accomplished at each level of the system architectural hierarchy. It is performed to ensure the product complies with requirements [9].

Validation is the Proof that the product accomplishes the intended purpose based on stakeholder Expectations [10]. It is performed for the benefit of the customers and users to ensure that the system functions in the expected manner when placed in the intended environment. This is achieved by examining the products of the system at every architectural level. It is also the proof, by examination of objective evidence, that the product accomplishes the intended purpose. Validation is performed to ensure that the product is ready for a particular use, function, or mission [9].

Starting from these definitions, a list of verification and validation requirements will be compiled in order to check against the objectives of this thesis. Firstly, all the system components and interfaces will be identified using an N2 chart to discover all the elements involved in the Facet Nano system. Then, tests will be designed and performed on each subcomponent and interface of the system to check whether they meet system requirements. Eventually, integration and validation will be done at the latest phase of the project when all the hardware components are physically available. It will be accomplished by designing an end-to-end test and comparing the performance against the validation requirements.

1.5 Experimental Set-up

The sensor selection will be assessed based on test results. These tests include:

a. Thermal tests in a dark thermal chamber to characterize noise behaviour versus temperature and obtain an estimate of the FPN of the sensor.

b. Electro-optical tests in a clean room to measure electro-optical parameters such as SNR and quantum efficiency.

c. Radiation test at ESA ESTEC facilities to study the effect of space environment conditions on the sensor and the changes on its properties during mission lifetime.

Apart from that, a simulation model of the sensor will be developed. It will be used both as a research and a validation tool. Research because it will help to estimate the system performance and validation because, if the model is correct, it will prove the design of the sensor, the optics and the software components.

1.6 Results, Outcome and Relevance

The starting point for this project has been already discussed in the *Facet Nano overview* section. The expected outcome is a camera able to capture night-sky pictures and detect stars and a computer model that simulates the behaviour and performance of this star camera.

The main deliverables of this project are:

i. A comprehensive report on the research performed stating the results of the work and a description of the taken steps.

ii. A performance budget of the system that computes availability and accuracy based on input parameters such as temperature, integration time and the sensor model.

iii. A diagram flow of the software, describing the different paths of the algorithms and the inputs and outputs of each block.

iv. Source code of the functional algorithms written in MATLAB.

v. An N2 Chart where all the subsystems and interfaces are identified.

vi. Requirements Discovery Tree and System Requirements documents, where the killing and driving requirements are inferred and described.

All these deliverables together will be used to decide upon the feasibility of the Facet Nano star camera as a system able to overcome the current attitude sensors for nanosatellites.

The work will reach a conclusion when a star camera capable of detecting sky stars with a precision defined by requirements is produced. The workload is estimated in nine months of full time dedication for a MSc student. The project planning is shown on a Gantt chart, attached to this document as an appendix.

1.7 Summary

The Facet Nano star camera represents a challenge in the space technology field. There is a trend of satellites becoming smaller and smaller. One reason is to reduce costs and development times. Besides, a group of small satellites could show capabilities in space yet to be discovered; the OLFAR project, a swarm of cubesats working as an antenna array is only one of the great applications we could see in the coming years.

The Facet Nano star tracker will improve the operations of lots of payloads flying on small satellites. They will be able to work during larger periods of time due to the increased availability of the attitude system. Therefore, FN will primarily improve the ADCS functionality and secondly the related payloads, improving the amount and the quality of scientific missions that will make use of small satellites to fly their payloads.

Chapter 2

Image Sensors and Star-Detection Algorithms for Star Trackers

2.1 Introduction

Current AOCS (Attitude and Orbit Control Systems) for nanosatellites include sun sensors, gyroscopes and magnetometers. Magnetometers measure the intensity and direction of the Earth's magnetic field. This sensor has a medium accuracy of 1 deg, however its main disadvantage is that it can work only at LEO (Low Earth Orbits) because the magnetic field weakens with increasing altitude [11].

Analog sun sensors are the most commonly used sensors; however, they are extremely inaccurate: 0.3 deg of accuracy in FOV (Field Of View) of 30 deg. They need direct sunlight and so they are installed at the panels of the satellite instead of several solar cells. In our case this is a critical problem because fewer solar cells means less power provided to the satellite [2]. Besides they do not work on eclipse, reducing their rate of success [3].

Star trackers provide numerous advantages over other attitude sensors because of their ability to provide full, three-axis orientation information with high accuracy and flexibility to operate independently from other navigation tools. However, current star trackers are optimized to maximize accuracy, at the exclusion of all else. Although this produces extremely capable systems, the excessive mass, power consumption, and cost that result are often contradictory to the requirements of smaller space vehicles.

Star trackers image a part of the sky and, by comparing a map of the sky to the picture, it can determine its orientation relatively to the stars. They have the higher accuracy of

all AD systems but their size and weight are too large for nanosatellites [4]. Thus a question arises, is it possible to design smaller and lower cost star trackers that can provide adequate attitude and rate determination to small, highly maneuverable, low-cost spacecraft? ISIS is developing Facet Nano, a star tracker prototype for nanosatellites that will provide a better accuracy and rate of success than the common attitude determination methods used on these satellites.

In a spacecraft, the star-tracker guidance system determines the attitude by matching an observed star field to a star catalog. The core components of a star tracker system are the Optical, Mechanical, Electrical and Software subsystems. The image sensor, included in the electrical subsystem, is the most important component of a star tracker, as the whole instrument is designed around it [12]. After the image sensor has acquired an image of the sky, software algorithms process it to detect stars and match them against a catalog like the HIPPARCOS [13] or the Tycho-2 [14].

The purpose of this document is to explain the guts of an image sensor and identify the different types of sensors and their properties in order to make a wise election depending on the final application, as well as to review the different software algorithms that enable star detection and identification by processing sky images.

2.2 State of the Art

2.2.1 Image Sensors

There are two main technologies in the field of image sensors: Charged-Coupled Device (CCD) and Complementary Metal Oxide Semiconductor (CMOS). The technologies and the markets that use them continue to mature, but the comparison has a strong dependency: both can meet the requirements depending on the application. Each one has unique strengths and weaknesses giving advantages in different fields [15] [16].

CCD sensors present higher quantum efficiency (QE, the number of electrons produced by the sensor for each photon hitting the pixel array [17]). On the other hand, they consume considerably more than CMOS sensors and they integrate incoming light as each column is being read out sequentially [18]. CMOS sensors used to have a lower performance until the appearance of micro-lenses materials and of more advanced semiconductor manufacturing technologies [19]. Lower CMOS quantum efficiency is compensated nowadays by micro-lenses that improve the fill factor (FF, the amount of light hitting and active part of the pixel array) up to 70% [20][21]. A more expensive technique to improve QE and FF is the backthining of a waffle of sensors [22].

In addition, the read-out circuitry and addressing also differs considerably comparing CCD and CMOS. For CCD the read-out is done sequentially between columns of pixels. Meanwhile for CMOS the read-out is done pixel by pixel, which makes it simpler to track individual stars moving throughout the FPA.

2.2.2 Star Detection Algorithms

The process of determining the position and the magnitude of a star from an image can be extremely difficult. A systematic approach to estimate the instrumental magnitude of each star is proposed by [23]:

- Find a star
- Center a window of N pixels around the star
- Add up the total number of electrons generated in the window
- Determine the average background in the image
- Determine the instrumental magnitude
- Determine the aperture correction
- Compute the aperture-corrected instrumental magnitude

Each of the former steps needs a deeper study to be carried. As well as estimating the instrumental magnitude of the star, finding its position with respect to a inertial reference frame is also required. A method to find stars on an image is via noise reduction and centroiding techniques [24]. This can be done via image thresholding, centroiding and then the formation os star clusters [25]. Nonetheless, noise reduction is not always efficient for a small satellite computing capability, nor effective, leading to many false positives, i.e., the detection of stars that were simply noise [26].

More complex centroiding algorithms include the detection of stars up to sub-pixel accuracy using numerical models of PSFs (Point Spread Function, the projection of a star on the image sensor) [27].

2.3 Results and Analysis

CMOS sensors are increasingly at the forefront of choices for implementation in imaging systems. This is due to a higher balanced performance compared to CCDs, smaller size, reduced complexity and cheaper cost [25].

Of the 2 types of CMOS sensors available (rolling and global shutter [28]), the rolling shutter design would theoretically ensure a sensitive sensor for the low light level application. This is based on the larger FF of the design architecture. Nevertheless, this would come at the sacrifice of an expanded electrical design and image processing software to correct for motion blur effect [29].

With respect to the software algorithms, the most promising performance is reached in combination between the modeling of PSF and the noise threshold. This could be achieved by correlating the acquired images with a modeled star-window. Further research is required in order to characterize the performance and the requirements of this technique.

2.4 Summary

In this chapter the characteristics of the key element in any optical instrument, the image sensor, are discussed. The two main technologies were presented: CCD and CMOS sensors. Although CCD devices are more sensitive to light and present a better quantum efficiency, CMOS devices have other characteristics that make them the ideal candidates for space applications, specially for small satellites.

For future work, it is recommended that CMOS sensors are preselected based on datasheet specifications. Radiation, thermal, optical and electrical tests shall be designed to check whether any of the preselected sensors meet the expected performance and whether they are able to survive the space environment.

In addition, state of the art algorithms for star detection and identification were reviewed. From the techniques analyzed in this chapter, it is concluded that several techniques could be merged to increase the performance and decrease the processing time. This is an innovative method that requires a more detailed analysis of the algorithms and their CPU requirements to check whether they could run on board a small spacecraft.

Chapter 3

Systems Engineering

Systems engineering is both an art and a science. We can compare it to an orchestra. Most people understand what music is, but not everyone can play an instrument. Each instrument requires different expertise and skill. Some musicians spend their entire careers mastering a single instrument, but sophisticated music involves many different instruments played in unison. Depending on how well they come together, they may produce beautiful music or an ugly cacophony.

Wiley J. Larson [30]

3.1 Systems Engineering Management Plan

This document describes the system engineering management plan(SEMP) for the Facet Nano project. The SEMP identifies and describes the SE team organization, roles and responsibilities, overall tasks, and engineering management planning required to control the design, development, fabrication, and tests associated with the Facet Nano project. This document shall apply to all participants of Facet Nano project.

Systems engineering has the responsibility for managing the technical aspects of the project. The systems engineer is responsible for technical oversight of the project and for coordinating the technical aspects of the project, particularly internal and external project interfaces. While one individual is assigned as the lead systems engineer, it must be recognized that systems engineering is a team effort, in which the entire technical community must participate, in order to achieve project success within the allotted constraints. The lead systems engineers role is to work with the project manager and the technical community to satisfy the customers needs.

3.1.1 Technical Design Verification

Technical design verification and validation will be accomplished through the use of one of the following methods: test, analysis, demonstration, similarity, inspection, simulation, validation of records, or not applicable, as described in the NASA Systems Engineering Handbook entitled SP- 6105 Design Requirements Verification Matrix.

The following section will list all requirements associated to the Facet Nano project. Besides, it identifies the verification method assigned to each requirement in the verification plan. Each procedure used for verification and validation testing will be shown in the verification plan. The tests will also be scheduled and resource loaded on the project Gantt chart.

3.1.2 Interface Control

The design team, along with the SE project manager, shall determine when formal interface control documents are required. Initially a system interface document, specifically an N2 chart, will be prepared and expanded with each phase of the project.

3.1.3 Technical Performance

The technical performance parameters selected for tracking shall be key indicators of project success. Each parameter identified shall be correlated within specific requirements. The list of performance parameters include:

- Attitude Determination Accuracy: defined as the difference between the attitude provided by the system and the error-free real spacecraft attitude.
- Attitude Availability: defined as the percentage of time that the star tracker is able to provide an attitude with an accuracy as good as the design target.
- System Environmental Performance: defined as the behaviour and degradation of the system over flight time. This parameter will likely limit the lifetime of the system.

3.1.4 Project Risk Analysis

Technical, safety, cost, schedule, environmental, and programmatic risks shall be addressed by the Facet-Nano Project by following the ISIS Risk Management Procedure.

The objective of the Facet-Nano Project Risk Management Procedure is to document the process in which the project team will identify and assess the risks in achieving project success, and to balance the mitigation of these risks against the acceptance and control of these risks.

3.1.5 Systems Engineering Process

This section of the SEMP shall describe the systems engineering process activities as they will be used to accomplish the project.

3.1.5.1 Requirements Analysis and Definition

System requirements will be generated to formalize the technical specifications of the projects. Requirements will allow to undertand the problem to be solved. They shall be:

- Atomic: only one sentece per requeriment, where it is easy to identify the subject, the result and the success criteria.
- Verifiable: in order to allow to prove that they have been met during the verification phase of the project.
- Unambiguous and measurable: to avoid interpretation issues.

3.1.5.2 Functional Analysis

A functional analysis shall be performed at the formulation phase of the project. This analysis will form the basis of the generation of system requirements. The functional analysis shall be an informal iterative process stemming from the top-level requirements. The functional analysis shall also provide an avenue to verify that all requirements have been identified.

3.1.5.3 Tradeoff Studies

Opportunities for tradeoff studies shall be identified and performed by the appropriate design teams. These studies shall take into account all relevant issues including technical, economic, and scheduling feasibility. All final decisions and the rationale for the decisions will be documented.

3.1.5.4 Documentation

The following documentation shall be produced for the Facet Nano Project:

- Systems Engineering Management Plan
- Requirements Document
- N2 Chart
- Functional Diagram
- House of Quality
- System Performance Budgets
- Verification Plan
- Software Documentation

3.1.5.5 Systems Engineering Tools

During the course of the project, various types of analysis will be performed. Due to the large market of tools available to aid in analysis efforts, the project team must come to consensus on certain types of tools that will be used. This uniformity will ensure compatibility between the files that would be shared among team members, thereby minimizing loss of productivity. Other tools shall be added as needed throughout the life cycle of this project. Software configuration management shall be addressed in computer software. Current tools to be applied to the Facet Nano Project are as follows:

Table 3.1: System Engineering Tools

Products	**SW Tools**
Schedule	Microsoft Project
Requirements	Microsoft Excel
Functional Analysis	Microsoft Visio
Decision Making	Microsoft Excel
Presentations	Microsoft PowerPoint
Plans and procedures	Microsoft Word
Software	MATLAB

3.1.5.6 Standarization

Design teams, to the maximum extent practical, shall make use of common parts, equipment, or supplies throughout all phases of the Facet Nano Project. Considerations for standardization include, but are not limited to:

- Reducing the number of different models
- Maximizing the use of common parts in different equipment
- Maximizing the use of commercial off-the-shelf (COTS) items and components
- Maximizing the use of interchangeable parts
- Maximizing the use of industry standards

3.2 System Requirements

The Facet-Nano star tracker instrument provides a better accuracy and rate of success than the common attitude determination methods used on nanosatellites. The star-tracker combines an accuracy of about 0.01 degrees with an inherent robustness against sun-blinding. It is designed to be compatible with the CubeSat standard in order to enable CubeSat missions with high pointing requirements.

The following system requirements are the result of several iterations and reviews carried out to achieve the goals stated in the former paragraph:

ID	Requirement	Rationale	Verif.
1	**Functional**		
1.1	Facet Nano shall function as an attitude sensor for the spacecraft	It can be the only sensor or part of a suite of attitude sensors	S, T
1.2	Facet Nano shall detect stars with a camera unit(s)	An image sensor is required to be able to detect the spectral wavelengths emitted by stars.	S, T
1.3	Facet Nano shall identify the detected stars	The true stars have to be identified from the noise background and false positive stars	S, T
1.4	Facet Nano shall determine the star position vectors in the inertial reference frame	The inertial reference frame is relative to the defined axis of Earth. The star coordinates stored in catalogs are based on this inertial frame	A, S, T

1.5	Facet Nano shall determine its absolute attitude vector	The FN attached to a spacecraft has its own reference frame which is relative to the inertial frame of space. The attitude vectors of the FN are then determined.	A, S, T
1.6	Facet Nano shall provide its attitude data to the S/C	The S/C OBC needs the attitude data	A, T
2	**Mission**		
2.1	Facet Nano shall autonomously determine its attitude	As its primary mode, the FN has to be able to measure its attitude upon request. Secondary modes using previous attitude estimation or other attitude sensor data is only considered when its highest accuracy performance cannot be guaranteed.	A, S, T
2.2	Facet Nano should be able to track stars	This mode can be implemented to support the FN when the success of its primary mode cannot be guaranteed. This mode relies on using previous attitude data to attempt to identify previously detected stars that may appear in current time step. Its accuracy is highly dependent on the number of stars that can be tracked.	S, T
2.3	The Facet Nano optics shall transmit the light energy	The optical subsystem has to transmit the visible spectrum of the star energy. This can be achieved by means of diffractive optics or mirrors.	A
2.4	The Facet Nano sensor shall detect the transmitted energy & measure the strength of the signal	The detection is affected by the spectral response of the sensor, noise sources as well as the optics transmission. The sensor has to convert the light energy from an analogue signal to digital signal that can be interpreted by the subsystem.	S
2.5	The Facet Nano shall maintain a good SNR during its operation	The SNR has to be maintained above the minimum level to ensure continued performance over its operational lifetime. To achieve this the sensor performance has to be characterized for its environment (thermal & radiation effects, expected light sources). This requires good knowledge of the signal threshold.	A

2.6	The Facet Nano shall generate a raw image based on its FOV	The captured image has to be processed before attitude determination	T
2.7	The Facet Nano shall implement image processing algorithms to the raw star image	As the raw star contains effects of noise sources, the algorithm will filter this out as best as possible.	A
2.8	The Facet Nano shall implement star identification algorithms to identify stars observed in the image	The basic function identifies stars by finding matches in a stored catalog. The catalog only contains stars that can be detected by the system	A
2.9	The Facet Nano shall implement an attitude determination algorithm	This algorithm has to measure the attitude using the output from the star identification algorithm	A
2.10	The Facet Nano should be able to measure the angular rotation rate	This is an additional attitude data that can be provided to the spacecraft if other attitude sensors cannot measure this accurately	T
2.11	The Facet Nano shall operate when commanded by the S/C	The FN only operates when required as part of its duty cycle	T
2.12	The Facet Nano shall transmit the attitude data to the S/C when requested	Data transfer should only happen when commanded	T
2.13	The Facet Nano structure shall provide mechanical support to the optical subsystem	It aligns the whole optical subsystem together and also to the sensor	I
2.14	The Facet Nano structure shall provide mechanical support to the sensor	This will ensure that the sensor is aligned properly with the optical subsystem to get the desired performance	I
3	**Interfaces**		
3.1	Facet Nano structure shall be mounted to the S/C		I
3.2	Facet Nano structure shall mount the optical subsystem		I
3.3	Facet Nano structure shall mount the sensor		I
3.4	Facet Nano structure shall mount the other electrical hardware		I
3.5	Facet Nano optics shall transmit the light energy on the sensor surface	This data transmission is in the form of a digital signal. An image is generated from the data.	A

3.7	Facet Nano processor shall interface with the image processing algorithms	The algorithms are stored in the processor	A
3.8	Facet Nano processor shall interface with the star identification algorithm	The algorithm is stored in the processor	A
3.9	The star identification algorithm shall interface with the star catalog	The catalog stores the positions of detectable stars as well as the all the possible star patterns. The number of patterns is dependent on the camera configuration (the FOV and number of apertures)	A
3.10	Facet Nano processor shall interface with the attitude determination algorithm	The algorithm is stored in the processor	A
3.11	Facet Nano processor shall provide the attitude data to the S/C ADCS or OBC	This depends on the design of the S/C if the OBC handles all processes or if there is a separate ADCS controller from the OBC	T
3.12	Facet Nano shall have mounting interfaces to the MGSE	Since the FN is a delicate equipment it needs a proper MGSE to avoid damage during handling	I
3.13	Facet Nano shall have data interfaces to the EGSE	The FN has to be calibrated during its assembly & integration phase, so EGSE is required to assist.	I, T
4	**Environmental**		
4.1	Facet Nano shall have a 1st mode of natural frequency, for all 3-axes, above 90 [Hz]	ISIS CubeSat Qualification Requirement	A, T
4.2	Facet Nano shall survive the maximum Static & Dynamic Launch loads that can be experienced	ISIS CubeSat Qualification Requirement Longitudinal Min -6,9 [g] Longitudinal Max +10,8 [g] Lateral Min -7,5 [g] Lateral Max +7,5 [g]	A, T
4.3	Facet Nano shall survive the maximum Random Vibration load	ISIS CubeSat Qualification Requirement 20 to 60 [Hz] = 0,030 [g$\hat{2}$/Hz] 60 to 150 [Hz] = +6 [dB] 150 to 700 [Hz] = 0,0747 [g$\hat{2}$/Hz] 700 to 2.000 [Hz] = -3 [dB] GRMS 11,15 [g] // 120 sec/axis	A, T
4.4	Facet Nano shall survive the maximum Static Vibration load	ISIS CubeSat Qualification Requirement 5 -10 [Hz] = 10 [mm] (0 to P) 10 to 100 [Hz] = 4 [g] 1 upsweep // 2 oct/min	A, T
4.5	Facet Nano shall be designed to function at altitudes at LEO (100 [km] to 1000 [km])	Nanosatellites have only been used at LEO	A

4.6	Facet Nano shall be able to survive its mission lifetime (3 [years])	Based on average lifetime expectancy of current Nanosatellites	A
4.7	Facet Nano sensor shall survive in a radiation environment TBD TID for 3 year mission at 650km altitude	The FN design can be constrained for a minimum shielding thickness and the sensor can be verified to meet that TID.	A
4.8	Facet Nano optical elements shall survive in a radiation environment TBD TID, for a 3 year mission at 650km altitude	The material should have an EOL transmission that ensures continued camera performance over its mission lifetime. It also has a secondary function of radiation shield	A
4.9	Facet Nano shall be designed to survive in a radiation environment TBD TID, for a 3 year mission at 650km altitude	Based on the expected mission requirements and design requirements of the optics and sensor, the TID limit can be refined to meet the system survival and operational requirements.	A
4.10	Facet Nano shall limit the effects of ATOX on the structure	At LEO, there is still the presence of ATOX that can degrade the FN structure and components.	A
4.11	Facet Nano shall be designed to operate in an electromagnetic & plasmic environment	The system has to be designed to avoid any static discharges and short circuits	A
4.12	Facet Nano optics shall survive the maximum pre-load acceleration (100 [g])	Since the optical material is brittle, it has to withstand the pre-load applied during its lifetime. The worst case acceleration is taken during the launch phase.	A, T
4.13	Facet Nano optics shall have a survival temperature range	-50 [C] to +100 [C] It should withstand the temperature that could be experienced in space. The biggest failure point is the effect of thermal shock from high thermal gradients. This can lead to fracture of the brittle material.	A, T
4.14	Facet Nano optics shall have an operating temperature range	-40 [C] to +70 [C] The optical material has to tolerate this temperature range without large negative impact on the image quality. The optics and the opto-mechanic structure should have the smallest thermal gradient possible.	A, T

4.15	Facet Nano optics shall have a storage temperature range	-50 [C] to +100 [C] It should withstand the temperature range without suffering permanent long term damage & shortening of the life cycle.	A, T
4.16	Facet Nano electronics shall have a survival temperature range	-40 [C] to +90 [C] Typical survival range of COTS electronics	A, T
4.17	Facet Nano electronics shall have an operating temperature range	-30 [C] to +40 [C] Operating range is tighter due to the desired lower noise effects for the low light level application. The sensor will be the determinant factor of this range value due to its sensitivity.	A, T
4.18	Facet Nano electronics shall have a storage temperature range	-40 [C] to +90 [C] It should withstand the temperature range without suffering permanent long term damage & shortening of the life cycle.	A, T
4.19	Facet Nano shall have a survival temperature range	-40 [C] to +80 [C] This takes into consideration the survival temperature range of the optics and electronics of which the electronics has a more strict requirement.	A, T
4.20	Facet Nano shall have an operating temperature range	-30 [C] to +40 [C] This takes into consideration the operating temperature of the optics and electronics of which the electronics has a more strict requirement.	A,T
4.21	Facet Nano shall have a storage temperature range	-40 [C] to +80 [C] Typical room conditions that may be encountered in cold & hot weather (including safety factor)	A, T
4.22	Facet Nano EMC shall be identified and analyzed	-	A, T
4.23	Facet Nano shall function in ambient pressure	100 kPa	T
4.24	Facet Nano shall function in near vacuum pressure	100 Pa	T
4.25	Facet Nano shall have a clean-room cleanliness requirement	Reduce contamination risk on optics & sensor surface in the final integration.	A, T
5	**Physical**		
5.1	Facet Nano shall fit within a CubeSat cross sectional area	100 x 100 [mm2] This is the maximum cross sectional area of a CubeSat and the system should fit within a conventional POD	I, T

5.2	Facet Nano volume should be equal to or less than the volume of a 1-Unit CubeSat	100 x 100 x 100 [mm3] Ideally the Facet Nano should fit inside a 1-Unit volume as a size any larger would just not be economical. Baffles should be taken into consideration, unless they are extendable or there is a unique mission requirement.	I, T
5.3	Facet Nano shall have a maximum weight	1 [kg] In most microsat range and higher, a star tracker should only be a small mass fraction of the S/C. With a CubeSat, this fraction will be higher. It would only be feasible for CubeSats larger than a 3-Unit. The 1 [kg] is therefore restricted by the previous 1-Unit volume limit.	I, T
5.4	Facet Nano shall use optical material appropriate for the radiation environment	Typical glass or plastic materials would deteriorate from gamma radiation exposure. An appropriate material choice needs to have sufficient EOL performance.	A
5.5	Facet Nano electrical hardware shall store all the software internally	To function autonomously & efficiently, the FN should store all the working code & software algorithms in its processor or memory.	A
5.6	Facet Nano shall house the optical subsystem in the internal structure	To provide mechanical support & protection from the environment. Tolerances have to be specified for the optical elements and mechanical structure for compatibility.	I
5.7	Facet Nano shall house the electronic subsystem in the internal structure	To provide mechanical support & protection from the environment	I
5.8	Facet Nano structure shall provide minimum radiation shielding to all hardware	specify min wall thickness The shield thickness will depend on the mechanical design and also the environmental requirements of the TID limit	A
6	**Operational**		
6.1	Facet Nano shall operate primarily in a ”lost in space” mode	Since it is usually the sensor with the highest accuracy on-board the S/C, a star tracker should not rely on other sensors. It has to have a fast response time	A

6.2	Facet Nano should have a secondary "tracking" mode	The tracking mode is mean to supplement the attitude determination by predicting stars that may appear in the next time step. It should also attempt to provide a rough attitude in case there are not enough detected stars for an accurate attitude determination.	A
6.3	Facet Nano shall have an on-board controller to execute all processes		A
6.4	Facet Nano sensor shall have a minimum data sampling rate	1 [Hz] A good update rate is required to ensure that FN can provide continuous attitude data for the continued operation of the S/C & its payloads.	A
6.5	Facet Nano controller shall have a minimum sampling rate to the sensor	2 [Hz] The update rate should be higher than the sensor data sampling rate to ensure that no information is loss.	A
6.6	Facet Nano shall tolerate a maximum rotation rate	1 [deg/s] FN should still be able to determine the attitude if a S/C is rotating. The success rate is inversely proportional to the rotation rate but the robustness can be increased with the algorithm design.	A, T
6.7	Facet Nano shall have a minimum success rate for attitude determination at the maximum rotation rate	80% The 80% is an initial estimate to provide a good system performance	A
6.8	Facet Nano should have a minimum success rate for angular rotation rate determination at the maximum rotation rate	90% Results of the success rate for different use cases will be a requirement by customers.	A
6.9	Facet Nano shall have a minimum pointing accuracy	$<$108 arcsec [3σ] pitch axis The minimum value is based on the 1s accuracy of 36arcsec requirement for FN $<$108 arcsec [3σ] yaw axis $<$1080 arcsec [3σ] roll axis The roll axis accuracy can vary up to 10x the pitch & yaw axis. It will be verified in the design phase.	A, S

6.10	Facet Nano sensor shall have a maximum integration time	TBD This is a rough value based on previous analysis of a sensor. The integration time is dependent on the performance of the selected sensor. It involves a study of the sensitivity, noise performance and thermal performance. The other modes for tracking and rotation rate tolerance also affect this choice.	A, T
6.11	Facet Nano camera unit shall transmit wavelengths in the visible light spectrum 300 [nm] to 1100 [nm]	This is the typical sensitive range of COTS sensors. Most COTS image sensors function in the visible spectrum and has less requirements than an IR or UV sensitive sensor.	A, T
6.12	Facet Nano star identification algorithm shall identify stars of a minimum apparent visual magnitude	Mv = +5 Initial performance analysis has determined this to be a good threshold allowing a variety of configurations	A, T
6.13	Facet Nano camera unit shall detect stars of a minimum apparent visual magnitude	Mv = +5.5 This detection threshold should be higher than that of the star identification algorithm. This ensures that the star identification will always have a high probability of identifying all the observed stars. Current competitors have a detection threshold of Mv +6.	A, T
6.14	Facet Nano shall determine the instrumental magnitude of the system	Due to reddening of stars by stellar absorption, the magnitude of observed stars by the system is not the same as the actual visual magnitude. This correction is done once in the design phase.	A
6.15	Facet Nano shall have a star catalog stored in the system	The star catalog is required as it stores information of all the detectable stars by the system. The main information is the stars position and patterns which are necessary for attitude determination.	I
6.16	Facet Nano shall have a minimum SNR	Studies on previous star trackers on small to large satellites have had an average 10 SNR. FN could tolerate a slightly lower SNR as we have a lower star detection threshold & lower pointing accuracy. But this is relative to the sensor performance.	A, T

6.17	Facet Nano shall have a maximum power requirement	<1W This will depend on the study of S/C use cases, payloads & available sensors. But it logically and practically cannot have a huge power demand. At least <15% of power budget.	A, T
6.18	Facet Nano shall have a minimum availability	>95% This value reflects the duty cycle when the FN is available to function.	A
6.19	Facet Nano sensor shall not be permanently damaged from direct sun exposure	Since this is likely to happen in CubeSat due to limited configuration choices & less accurate attitude control system, the sensor could get direct exposure and should not get burned out.	A
6.20	Facet Nano shall have a minimum full Field of View	10 A larger FOV would allow more stars to be observed. Initial performance analysis has confirmed this value as the minimum value when based on other criteria of success rate and initial sensor accuracy.	A, T
6.21	Facet Nano shall have a minimum Sun Exclusion Angle	10 This initial value has been confirmed by initial performance analysis though it does not include any ray tracing analysis. The SEA should be equal to or larger than the full FOV as a safety precaution to ensure that blinding by the Sun is minimized.	A, T
6.22	Facet Nano sensor shall maintain the detection threshold requirement up to EOL	Even with degradation of the sensor over time, it should still be able to meet the minimum detection threshold until EOL	A
7	**Human Factor**		
7.1	Facet Nano opto-mechanical assembly shall be assembled in a clean-room	This requires clean-room standards to be met by personnel during the assembly process. Appropriately skilled personnel required for the opto-mechanical assembly.	I
7.2	Facet Nano electronics hardware shall be assembled in an ESD protected room	Personnel must obey ESD precaution when handling the electronic components.	I
7.3	Facet Nano camera unit shall be assembled in a clean-room	This requires clean-room standards to be met by personnel during the assembly process.	I
7.4	Facet Nano camera unit shall be calibrated by qualified personnel	Calibration can only be done if the personnel & equipment are available to perform this.	I

7.5	Facet Nano shall be integrated in a clean-room	Final integration of the finished product has to be in a clean-room to avoid any contamination of the optics & sensor.	I
8	**Logistic Support**		
8.1	Facet Nano shall be transported in a sealed package	It should be vacuum sealed or airtight to reduce the risk of contamination.	I
8.2	Facet Nano shall be transported in a durable case with adequate protection	To ensure a low risk of damage to the product.	I
8.3	Facet Nano shall have a user manual included with the transport case	Current SOP of ISIS for all shipped products	I
8.4	Facet Nano shall have a handling and storage guide included with the transport case	Current SOP of ISIS for all shipped products	I
8.5	Facet Nano shall have a product history record included with the transport case	Current SOP of ISIS for all shipped products	I
8.6	Facet Nano shall be stored in a clean-room	Avoid prolonged contamination	I
8.7	Facet Nano shall be updated with a new star catalog if kept in long term storage	Stars do drift over time and each star drifts at different speeds.	I
8.8	Facet Nano shall have replaceable parts & components		I
9	**Product Assurance**		
9.1	Facet Nano shall have documentation created for each development test	This should show a record of the progress that has been made. It will allow easy verification of the test method & results.	I
9.2	Facet Nano shall have documentation created for each simulation test	This should show a record of the progress that has been made. It will allow easy verification of the test method & results.	I
9.3	Facet Nano performance budget shall be created based on the operational requirements	The performance budget will used as a measure to determine if the overall system meets the final requirements. It will be continuously updated as the requirements are refined from testing results.	I
9.4	Facet Nano optics shall be designed for optimal performance with the specified sensor	Due to different characteristics between sensors (e.g. pixel size), a custom optical design is required to give the best performance possible for a selected sensor.	A
9.5	Facet Nano opto-mechanics shall have its tolerances specified	This will make sure the optical & mechanical design are compatible from the onset of design.	I

9.6	Facet Nano opto-mechanics shall be manufactured to the tolerances specified	If the assembly is not compatible & does not meet the performance requirements, it has to be rejected.	A
9.7	Facet Nano optics shall have specified coating requirements	The manufacturer can provide a certificate of conformity or tests can be done internally to verify the requirement.	A, I
9.8	Facet Nano sensor shall be subjected to radiation tests	The tests will quantify the performance characteristics of the sensor. This is especially necessary for a COTS sensor that is not originally intended for space application. This test should be done as early as possible in the design phase due to its high risk factor.	T
9.9	Facet Nano sensor shall be subjected to thermal tests	The tests will quantify the performance characteristics of the sensor.	T
9.10	Facet Nano sensor shall be replaced if production run ends	If the sensor is no longer produced, a new sensor choice must be made.	A
9.11	Facet Nano shall be redesigned if there are major component changes	A new sensor would require new optics & therefore a whole new design revision.	A
10	**Configuration**		
10.1	Facet Nano controller shall consist of the processor & memory	Minimum equipment required for functionality of the hardware.	I
10.2	Facet Nano algorithms shall be executed only by the controller	The controller should control every function of the system.	A
10.3	Facet Nano camera unit(s) shall consist of the optical subsystem & image sensor		I
10.4	Facet Nano structure shall provide a base reference frame for integrated camera unit(s)	Since the structure forms the physical support of the system, it should be used a reference frame for aligning the camera unit & to the S/C.	I
10.5	Facet Nano shall be mounted to the S/C away from any noise sources	This will avoid additional noise sources that affect performance.	I
10.6	Facet Nano shall be mounted to the S/C away from any thermal sources	Since a low light level application needs very low noise levels (low temperature), negative contribution from thermal sources has to be avoided.	I
10.7	Facet Nano optics shall be aligned to the active area of the image sensor	The faint light sources need to be focused to increase the probability of detection.	I

10.8	Facet Nano electrical hardware shall be surrounded by the structure so as to provide maximum radiation shielding	The structure should not only provide mechanical support but also act as the primary radiation shield.	I
10.9	Facet Nano shall have a thermal control design	The thermal control is required for the FN to perform within its operating conditions.	I
11	**Design**		
11.1	Facet Nano structure shall have a coating over all exposed surfaces to minimize light reflection	By having black color coatings on surfaces exposed to light, this reduces the risk of stray light falling on the sensor active area.	A
11.2	Facet Nano shall implement baffles to minimize stray light reflection	The baffle helps with reduction of stray light & blinding by celestial bodies like the Sun, Earth & Moon.	A, S, T
11.3	Facet Nano optics shall have coatings to transmit the desired spectral range	The optical element coatings act as a filter for transmitting the desired wavelength. It also reduces reflection and stray light.	A, S
11.4	Facet Nano camera unit shall have air vents	Since the FN has to operate in space vacuum, air has to be allowed to vent during launch. Otherwise the applied pressure on the optics would lead to negative effects. The air vents will also allow access to clean the surfaces before launch.	I
11.5	Facet Nano structure shall have a minimum manufacturing tolerance	0.01 [mm] The opto-mechanical interfaces should have the same tolerances for best compatibility & operating performance	I
11.6	Facet Nano optics shall have a minimum manufacturing tolerance	0.01 [mm] The opto-mechanical interfaces should have the same tolerances for best compatibility & operating performance	I
12	**Verification**		
12.1	Facet Nano shall have a performance budget compiled	The performance budget is required to show the overall system performance. This will help determine how well the system is functioning when taking into account the contribution of each subsystem.	I
12.2	Facet Nano shall have a conceptual design review	The concept design will show how the design requirements can be fulfilled. The risks & design choices are the highlights.	I

12.3	Facet Nano shall have a detailed design review	This later phase will have a complete design architecture. Some development & simulation tests should be completed already for the high risk items. The solutions should be presented.	I
12.4	Facet Nano shall have a critical design review	This final design should be ready for production. All changes identified previously should have been implemented & the risks also addressed & solved. The final verification of the design is therefore necessary.	I
12.5	Each document shall be reviewed by another team member	By having team members review each other's work, this will give the team complete understanding of the system. This is necessary since the subsystem are intertwined together.	
12.6	Facet Nano shall have a design review for the mechanical subsystem	Since there are a lot of co-dependencies between each subsystem, each subsystem should be individually reviewed & the impacts to other subsystems analyzed	I
12.7	Facet Nano shall have a design review for the optical subsystem	Since there are a lot of co-dependencies between each subsystem, each subsystem should be individually reviewed & the impacts to other subsystems analyzed	I
12.8	Facet Nano shall have a design review for the electronics subsystem	Since there are a lot of co-dependencies between each subsystem, each subsystem should be individually reviewed & the impacts to other subsystems analyzed	I
12.9	Facet Nano shall have a design review for the software algorithm subsystem	Since there are a lot of co-dependencies between each subsystem, each subsystem should be individually reviewed & the impacts to other subsystems analyzed	I
12.10	Facet Nano sensor shall be subjected to TID testing	A COTS sensor made for terrestrial applications has to be verified that it will survive the projected mission lifetime while maintaining satisfactory performance levels.	T
12.11	Facet Nano sensor should be subjected to SEE testing	This testing is necessary as SEE can affect the availability & accuracy of the system	T

12.12	Facet Nano sensor shall be subjected to thermal tests in a controlled thermal chamber	This will allow testing to be performed over a wide range of temperature & temperature gradients that may be experienced during mission lifetime	T
12.13	Facet Nano shall be subjected to thermal tests in a thermal vacuum chamber	This is an important test as the thermal conditions in vacuum are not the same in terrestrial atmosphere. A vacuum chamber is needed to simulate this environment as closely as possible.	T
12.14	Facet Nano optics shall have a tolerance analysis performed	Due to the multiple variables present in the optical elements, a Monte Carlo simulation would determine how robust the design is. It would also verify the tolerance requirements.	A
12.15	Facet Nano shall have mission analysis performed	This will verify the functional & operational requirements. This will also help in verifying the performance budget.	A
12.16	Facet Nano optics shall have its production requirements verified	The manufacturer or end user shall verify that the finished product is within range of tolerances specified & meet the other production requirements.	A
12.17	Facet Nano mechanical structure shall have its production requirements verified	The mechanical parts have to meet the specified tolerances to ensure compatibility with the rest of the system & that the operational requirements are met.	A
12.18	Facet Nano camera unit shall be calibrated with proper equipment & personnel	The calibration is necessary to verify the assembly process & to provide the subsystem correction from unavoidable tolerance errors.	T
12.19	Facet Nano shall be calibrated with proper equipment & personnel	The calibration is necessary to verify the final integration & to provide the system correction from unavoidable tolerance errors.	T
12.20	Facet Nano shall have its operational requirements verified	This will test the whole system & verify the accuracy of the operational requirements. Comparison with the performance budgets will help to find room for improvement.	A

13	**Software**		
13.2	The software shall be able to track stars	This is in addition to the 'lost-in-space' mode. Tracking mode will be less cpu-demanding.	T
13.3	The software should be able to calculate the rotational rate around all three axes in its reference frame	Star-tracker reference frame	A, T
13.4	The software should be able to provide raw star data	In order to be processed by the star-tracker controller.	A, T
13.5	The software shall be able to autonomously calculate the attitude	As it is the main ACS sensor, it shall be able to operate independently.	T
13.6	The software should be able to autonomously calculate the rotational rate	As there might be no other sensors.	A, T
13.7	The software shall be able to autonomously determine its calculation time	In order to make performance checks.	A, T
13.8	The software shall start its operation when commanded by the spacecraft	It shall respond to OBC commands.	T
13.9	The software shall provide the attitude, rotational rate and calculation time to the spacecraft when commanded	Same as 13.8	S, T
13.10	The software shall ensure that the acquisition period shall be maximally 20 sec	In order to assure software optimization	T
13.11	The software shall be able to update the information at a rate of 1Hz or faster	1Hz is usual update frequency	S, T
13.12	The software shall be able to determine its calculation time with an accuracy of 0.001 sec or better	In order to beat other sensor accuracies	S, T
13.13	The software shall be able to improve the optical resolution with a factor 10 for a spacecraft with a rotational rate of maximal 0.1 deg/s		S, T
13.14	The software shall be able to determine its attitude from the recognized stars with an an accuracy of at least 5 arcsec		S, t
13.15	The software shall be able to improve the total system accuracy with a factor 2 by performing a statistical analysis		T

13.16	The software shall be able to determine the attitude of the facet nano for a spacecraft with a rotational speed of maximal 1 deg/s	Larger rotational speeds would decrease SNR and centroid accuracy	T
13.17	The reliability of the attitude determination algorithm shall be at least 0.998	To be verified by analysis and simulation	A
13.18	The rate of success of the attitude and rotational rate determination shall be at least 0.95	To be determined by the number of stars in the FoV	A
13.19	The software shall be able operate in LEO with an altitude ranging between 400-1000 km	As it is designed for cubesats operating in LEO	A
13.20	The lifetime of the software in the space environment shall be at least 3 year	Usual CubeSat lifetime	A
13.21	The reliability of the software during its lifetime shall be at least 0.98	RAMS analysis	A
13.22	The software shall be compatible with a I2C data interface	Standard for ISIS cubesats	A, T

Table 3.2: Facet Nano Requirements

Note: Verification methods include analysis (A), inspection (I), Simulation (S) and Test (T).

3.3 Functional Analysis

Figure 3.1 shows the functional analysis of the Facet Nano star tracker. It shows an end-to-end approach, identifying all the different functional blocks that integrate the system. Between the different blocks we can identify the internal an external interfaces of the system. These interfaces will be addressed by the N2 Chart.

3.4 Interfaces

3.4.1 N2 Chart

The N-squared (or N2) chart can be used to help define system interfaces. The Facet Nano N2 Chart is illustrated in figure 3.2. System components or functions are placed

on the diagonal; the remainder of the linkages represent various interface inputs and outputs. The N2 Chart complements the Functional Analysis and highlights the data flows as inputs and outputs of system functions.

Where a blank appears, there is no interface between the respective functions. The most populated area belong to the motherboard

3.5 House of Quality

The House of Quality is shown in figure 3.3. It is used for defining the relationship between customer desires and the program capabilities. The basic structure is a table with "Whats" as the labels on the left and "Hows" across the top. The roof is a diagonal matrix of "Hows vs. Hows" and the body of the house is a matrix of "Whats vs. Hows". Both of these matrices are filled with indicators of whether the interaction of the specific item is a strong positive, a strong negative, or somewhere in between.

The House of Quality functions as a living document and a source of ready reference for related products and future upgrades. Its purpose is to serve as a vehicle for dialogue to strengthen vertical and horizontal communications. Through customer needs and competitive analysis, the House of Quality helps to identify the critical technical components that require change.

3.6 Performance Budget

See figure 3.4.

As a result of the budget analysis, some TBD in the requirements specification were cleared:

- SNR margin (dB) must be larger than 3dB
- NEA (Noise Equivalent Angle) must not exceed 36 arcseconds.

NEA magnitude is directly related to attitude accuracy and is inversely proportional to SNR. Therefore, either the signal needs to be increased or the noise needs to be decreased in order to reduce the NEA. This implies that the effect of the noise is small if the signal is relatively larger than the noise and vice versa. Notes:

0. Circular orbit assumed

1. Circular FOV assumed
2. FOV is fixed to 14deg by design
3. Sky coverage is determined by simulation
4. Noise is extrapolated from experimental data
5. Maximum operational temperature is 39.3 Celsius

3.7 System Verification

The verification process is one of the main task of a system engineer. This process used by the system's owner and by other stakeholders to show that the as-built system, sub-system, and components meet all of their requirements and design. It is used by the system's owner and other stakeholders to accept the system products from the development team. There are many definitions for the meaning of system verification, but all of them agree on the same objective:

- System Verification is a testing process which ensures that the selected solution meets specified requirements and properly integrates with interfacing products.
- Verification is the process that proves the system [or sub-system or component] meets its requirements and matches the design.

In a nutshell, verification is the confirmation that the model does what it was designed to do [31]. Since verification is based on requirements and design, one of the keys to successful and effective verification is well-written and complete requirements and design documents. These requirements and design elements are developed, reviewed, and approved earlier in the project timeline before the system is developed or procured. Planning for the verification activities starts with the System Verification Plan. This plan is best written at the same time the requirements of the system, sub-system, or component are developed.

This is done to show that the requirements, as written, can be verified. At the end of the detailed design effort, verification procedures can be written. These procedures are the detailed steps to be taken to verify each requirement and design element. There must be a clear trace from each requirement, through the Verification Plan, down to a detailed step in the verification procedure. Verification is performed iteratively. It starts with the integration activities at the component level. It progresses through the sub-system development to the verification of the entire system.

We will see in chapter 6 a snapshot of the verification plan for the Facet Nano project, together with the verification activities that were carried out in order to assess the requirement compliance of the system.

3.8 Summary

In this chapter we presented the systematic approach to the Facet Nano project. Starting from the requirements, a functional block diagram was derived. Following, a N2 chart was obtained in order to identify all the interfaces within the system. Besides, the house of quality was used as a tool to identify the correlations between the customer needs and the design parameters.

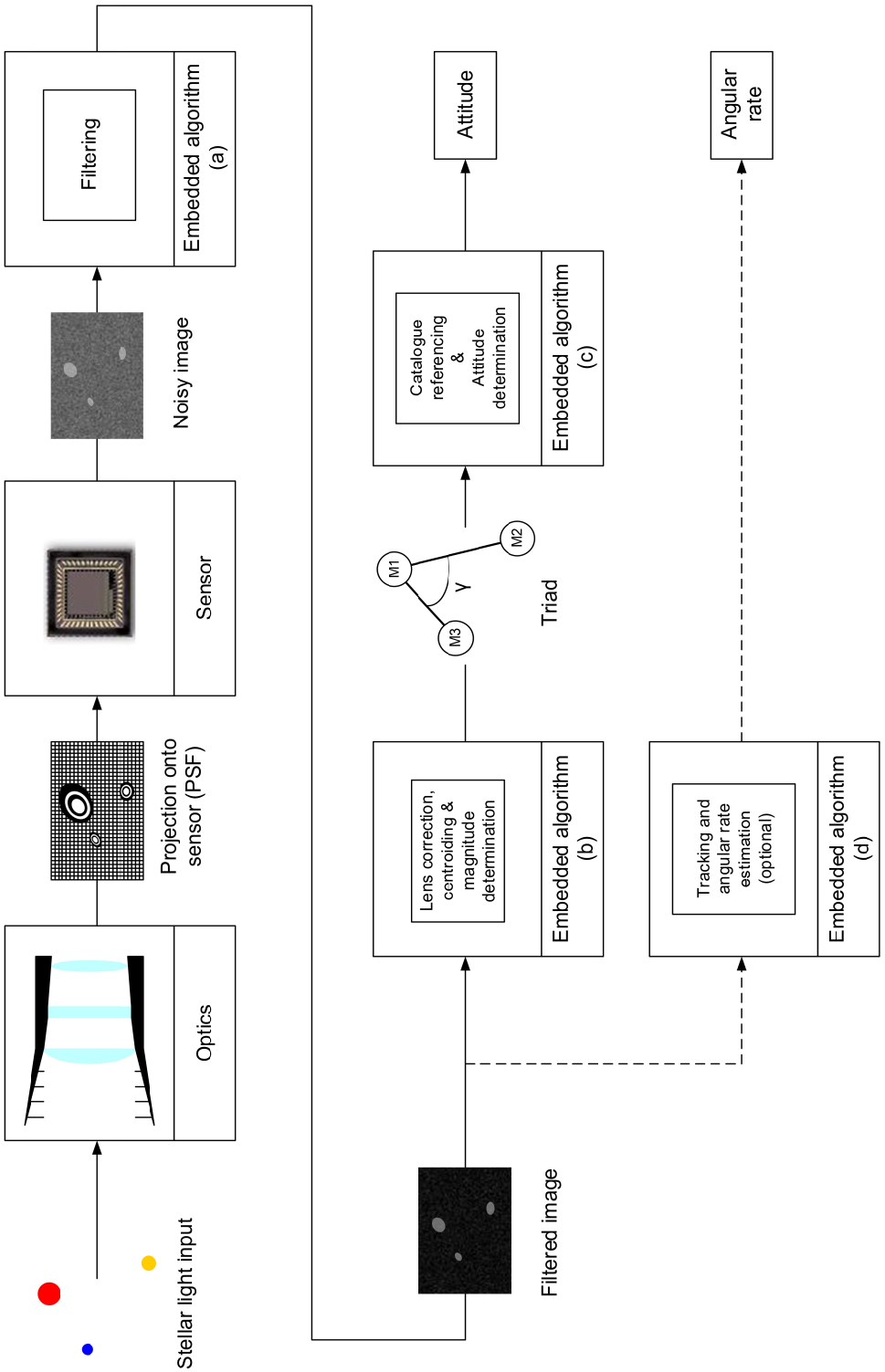

Figure 3.1: Functional Block Diagram

Space environment, attitude, dynamic state	**Star light** rays enter the aperture. Possibly other (unwanted) light sources as well	Temperature of the detector, 'smearing' upon the sensor. Radiation implies degradation	*NONE*	*NONE*	*NONE*	Improper star catalogue can severely disturb functioning		*INTERFACE NOT RELEVANT*
INTERFACE NOT CONSIDERED	**Optics**	Projected image onto the sensor: PSF, aberrations	*Influence of optics on noise is not considered*	Incorrect embedded modelling increases inaccuracies	-	-		*INTERFACE NOT CONSIDERED*
INTERFACE NOT CONSIDERED	-	**Sensor**	Digital information with added noise is sent to the microcontroller	-	-	-		*INTERFACE NOT CONSIDERED*
INTERFACE NOT CONSIDERED	-	*Change on integration time*	**Motherboard and processor**	Digital image with noise. Integration time for FPN cancelation.	-	-		*INTERFACE NOT CONSIDERED*
INTERFACE NOT CONSIDERED	-	-	*Change on integration time*	**Image Processing (filtering and noise threshold)**	Digital image w/o noise	false triads disturb attitude determination		*INTERFACE NOT CONSIDERED*
INTERFACE NOT CONSIDERED	-	-	*Number and position of detected stars*	TBC Future: tracking	**Star Detection (centroiding)**	Star positions at subpixel level		*INTERFACE NOT CONSIDERED*
INTERFACE NOT CONSIDERED	-	-	TBC	TBC Future: tracking	TBC Future: tracking	**Star Identification (star catalogue)**	Star vectors	INTERFACE IS NOT CONSIDERED
INTERFACE NOT CONSIDERED	-	-	Attitude data	-	-	-	**Attitude Determination**	**Attitude data** is passed on
INTERFACE NOT RELEVANT	Thermal	Thermal	*INTERFACE NOT CONSIDERED*	*Temperature for noise threshold and FPN cancelation*	*INTERFACE NOT CONSIDERED*	*INTERFACE NOT CONSIDERED*	*INTERFACE NOT CONSIDERED*	Satellite

Figure 3.2: Facet Nano N2 Chart

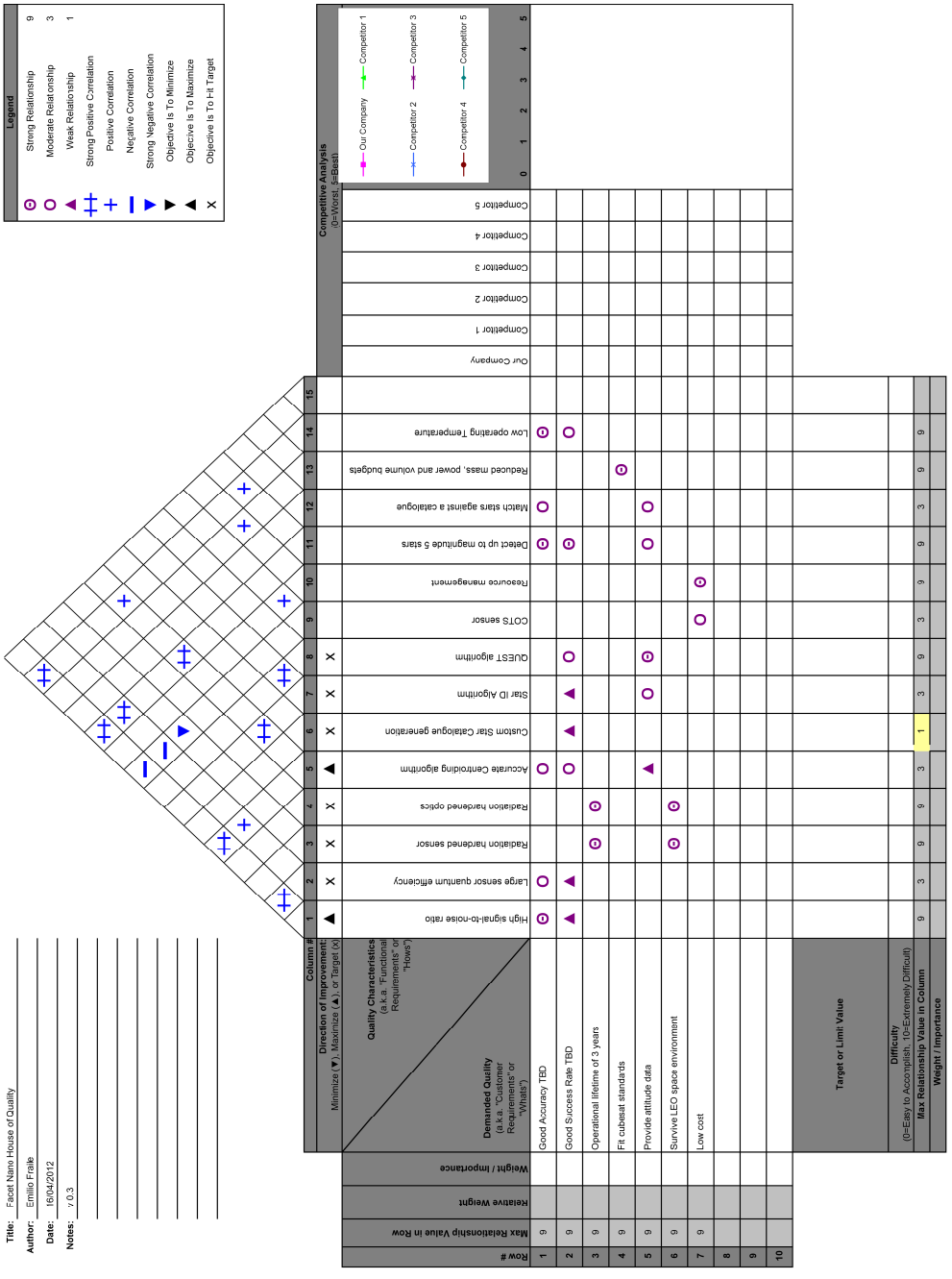

Figure 3.3: Facet Nano House of Quality

Facet Nano Performance Budget - 2 apertures	
Temperature [deg C]	25.0
Optics	
FOV[deg]	14
F#	1.3
Star	
Fintest detectable star magnitude	5.0
Intensity[W/m2]	3.373E-10
Photon flux [photons/(m2/s)]	8.474E+08
Fraction of the sky covered by the FOV[%]	0.37
Total number of stars available on the sky	1454.6
Average # stars in the FOV	10.8
Detector	
QExFF [%]	50
Pixel Size [m]	5.3E-06
Hor Resolution	640
Ver Resolution	512
Detector Area [m2]	3.68E-05
Exposure/integration time [s]	0.083
Dark&readout noise [electrons/s]	467
Centroiding Accuracy [pixels]	0.4089
Results	
Avalilability [%]	**81.61**
SNR (minimum) [dB]	**15.23**
NEA cross-boresight (pitch and yaw) [arcsecond]	**12.22**
NEA roll [arcsecond]	**12.22**

Figure 3.4: Facet Nano Performance Budget

Chapter 4

Image Sensor Selection

The biggest challenge proved to be the image sensor selection. It is the core of the star tracker and therefore the most critical part of the system. The space environment and the requirements of a star tracker restrict the choice of sensors that fit the picture. Furthermore, a custom developed sensor would prove too expensive and beyond the scope of the project of a cost effective product.

Table 4.1: Base requirements on sensor selection

Sensor Requirements	
Temperature	Survival range +40C to -30C Maximum operating temperature <+20C
Sensitivity	Detect a magnitude +5 star
Noise	The noise should be low to allow a SNR >10
Spectral response	Good spectral response in the visible spectrum (400-1000nm)
Power	Low power consumption (<400mW)

An update of the selected image sensor was released right after the optical and thermal tests completed on the 0.5MPix sensor. The company claims that the new 1.3MPix sensor exceeds the performance of the former one in terms of quantum efficiency while physical and electrical properties remain the same. Thus, a comparison between the two candidates was executed to choose the sensor that offered the best performance.

This chapter describes the characterization of the two candidate sensors, explains the criteria of the trade-off and shows the result of final selection.

4.1 Sensor Specifications

Table 4.2 shows the sensor specifications as stated in their data sheets. The main differences between sensors are:

The resolution is higher in the 1.3MPix sensor: 1.3Mpixels versus 0.5Mpixels. This generally means more detail in the images. Although very detailed images are not a key requirement in a star tracker, a larger number of pixels allows a spread of the incoming signal over a larger number of pixels, reducing the pixel signal-to-noise ratio but improving the centroiding accuracy. The 1.3MPix sensor uses a 10-bit ADC while 0.5MPix uses 8-bit ADC, thus quantization noise is a factor 4 lower in 1.3MPix than in the 0.5MPix. Signal-To-Noise ratio is similar in both sensors. Nonetheless, the 1.3MPix sensor includes a built-in binning capability that allows increased sensitivity and reduced temporal noise, resulting in an increase of the final SNR performance. Power consumption is doubled in the 1.3Mpix sensor to 200mW. This order of magnitude still remains within the power budget of the system although it shall be taken into account when considering a multiple camera design.

Table 4.2: Specifications of the preselected image sensor

Sensor characteristics	**0.5MPix**	**1.3MPix**	**Units**
Technology	CMOS	CMOS	-
Shutter Type	global	global	-
Resolution	838(H)x640(V)	1280(H)x1024(V)	pixels
Pixel size	5.8x5.8	5.3x5.3	μm
Pixel performance			
Bit depth	8	10	bits
Dynamic range	>51	>63	dB
Qsat (Full Well)	10	8.4	ke$^-$
SNR Max (25 deg)	40	39	dB
MTF at Nyquist, $\lambda = 550$ nm	50	50	%
Dark signal (25 deg)	88bits	2410bits	$LSB_{8/10}$/s
DSNU (25 deg)	6	6	$LSB_{8/10}/(nJ/cm^2)$
PRNU(RMS)	0.8	<1	%
Responsivity	178bits	8510bits	$LSB_{8/10}$/s
QE*FF	>70	>70	%
Pixel Binning Capability	No	Yes, 2x2	Pixels
Electrical Interfaces			
Power consumption:			
Functional	80	<200	mW
Standby	90	180	μW
Power Supplies	3.3 and 1.8	3.3 and 1.8	V
Operating Temperature	[-30, +65]	[-30, +65]	deg C

If the new sensor is larger due to the higher number of pixels, then a change in the optical design is expected. Optical engineer shall account for a larger FPA in order to distribute all the light coming from the stars on a larger sensor.

4.2 Noise Model

Ideally, the only photons passed through the imager are those from the stellar signal. However, all imagers contain uncertainties caused by the injection of various types of noise that are digitized along with the actual image.

The most common noise sources are signal shot, dark current and background noises. Additionally, there are pseudo-noise sources introduced during the process of digitizing the stellar signal, which include readout and quantization noises. A complete research on CMOS noise sources was made by Kara M. Huffman [4].

4.2.1 Signal Shot Noise

Signal shot noise results from the stellar signal falling on the imager and is due to the uncertainty in signal measurement. Huffman computes this term as the root square of the number of electrons that are generated on the imager. Nevertheless, this number of electrons depends on several optical magnitudes. As the optical design is not yet available, signal shot noise will be computed using a Poisson statistical distribution.

4.2.2 Dark Noise

Dark current noise is a result of thermally generated electrons, which are present even in the absence of light. The majority of these electrons are created in the forbidden energy gap between the valence and conduction band, where photons excite electrons between interface energy levels.

The number of dark current electrons at a given temperature is provided by the manufacturer in the data sheet. Using this value, Arrhenius law can be used to estimate the number of dark current electrons at any temperature. This is a first approach and the dark current noise must be measured for different temperatures in a thermal chamber.

4.2.2.1 Dark current

Dark current is caused by leakage of electrons into the charge storage region of the detector. These accumulate over the duration of the image integration. If we know the exact rate that these electrons leak into the charge storage region, we would just subtract them from the image and they would have no effect on our image at all.

However the arrival time of dark current electrons into the charge storage region is a statistical process. As with many time-integrating thermal processes, the arrival time has a Poisson distribution. As such we can measure the mean fairly accurately, and we can be confident that if we subtract the mean then the noise will be:

$$N = \sqrt{Q_D} \tag{4.1}$$

Where

$$Q_D = I_D \cdot t \tag{4.2}$$

For t = time and I_D = mean dark current.

4.2.2.2 Fixed Pattern Noise

Also the Dark Current is not the same for every pixel. Some pixels are hotter than others. This dark current pattern is fixed, and we can assume that we can subtract different dark current for each pixel.

However we need to understand the effect on noise. First consider a chip without a fixed pattern. If we were to choose pixels at random from dark image to generate a distribution of dark current, all pixels having the same statistical mean dark current, we would end up with a Gaussian distribution. Then consider that we choose pixels at random to generate the same distribution, but this time with a fixed pattern of dark current, then the second distribution will be more spread out. In other words the noise has increased.

Mathematically, the FPN can be considered as a multiplier on the dark current, such that the dark current I_{Dq} in pixel q is

$$I_{Dq} = (I_D \pm \sigma_D^2) \cdot (FPN \pm \sigma_{FPN}) \tag{4.3}$$

Therefore the errors will be:

$$N = I_D \cdot FPN \cdot \sqrt{\frac{\sigma_D^2}{I_D^2} + \frac{\sigma_{FPN}^2}{FPN^2}} \tag{4.4}$$

The mean of the FPN multiplier is 1, yielding

$$N = \sqrt{\sigma_D^2 + (I_D \cdot \sigma_{FPN})^2} \tag{4.5}$$

4.2.3 Background Noise

There are numerous types of background noise, but each is generally very specific to application and, therefore, often difficult to model. Background noise originating from the surrounding environment is usually quite small when compared to the lower magnitude stars generally used for navigation; it would certainly be dominated by other potential sources, such as glint from the sun as it reflects off of other surfaces of the spacecraft.

Background noise will be neglected in the simulation.

4.2.4 Quantization Noise

Quantization noise is a result of uncertainty in the true value of the electron count, as described above, caused in the ADC. The standard deviation of the quantization noise is:

$$\sigma_{quantization} = \frac{\Delta}{\sqrt{12}} \tag{4.6}$$

Where

$$\Delta = \frac{fullWell}{2^N} \tag{4.7}$$

N is the number of bits for the A/D converter, and fullWell represents the full well capacity and the maximum number of electrons that a single pixel can hold.

4.2.5 Readout Noise

Readout noise is associated with the inexact conversion of electrons from the signal falling on the imager to the readout of the imager. It is mainly caused by the on-chip output amplifier, where electrons are converted into analog voltage. The standard deviation of readout noise is given by:

$$\sigma_{readout} = \frac{R}{\Delta} \tag{4.8}$$

Where R is the number of electrons caused by the readout noise and Δ is the gain in electrons/ADU, as used above for quantization noise.

4.2.6 Reset / Thermal / KTC noise

This noise was not initially considered by Huffman. However it is important to consider it as it is a notorious noise source, especially at low temperatures where the dark current signal is not so dominant.

4.2.7 Addition of noise sources

The total amount of noise falling on the imager consists of several sources, each with a distinct statistical dependence. By statistical definition, the variance is the mean square of the difference in actual value and average value. For large sample sizes, the variance for the noise electrons is:

$$\sigma_N = \sqrt{\sigma^2_{shot} + \sigma^2_{dark} + \sigma^2_{quantization} + \sigma^2_{readout} + \sigma^2_{KTC}} \tag{4.9}$$

Each noise source can be attenuated using different techniques. For example, dark current will decrease if the detector is cooled by an active or passive device; quantization noise will be reduced if a higher number of bits are chosen for the ADC converter.

4.3 Sensor Characterization

The parameters of different sensors are highly interconnected but since information in the data sheets was not uniform between manufacturers, it was difficult to determine some of the constraint fulfillment due to incomplete data. They either were undefined or could not be provided by the manufacturer.

In the end a test campaign based on identical tests conditions was the best way to compare and select a suitable sensor. Due to cost and time restrictions, a simple selection process had to be implemented. The objective of this test is to measure the performance of different image sensors and select the best available. They will be measured according to the same methods for better comparison of the sensor that meets the Facet Nano requirements.

Demo kits of the sensors, provided by the manufacturers, were tested as it would consume too much time in developing individual development boards for each sensor.

The main disadvantages of the demo kits are:

- The kits had pre-set parameter value ranges that could not be modified
- Some sensors came with pre-built micro-lenses. These effects would be removed in the results post-analysis
- Data extrapolation had to be done to determine the best performance for Facet Nano application
- Limitations on the software provided by the manufacturer (e.g., maximum and minimum integration times)

4.3.1 Dark Noise

The aim of these tests is to measure the sensor read-out in absence of any external light source. The output is pure noise, whose main source was dark current. Measuring this noise for different temperatures was necessary for computing the SNR values for each sensor.

Figure 4.1 shows a comparison among the magnitude of the different noise sources that are present on the two CMOS sensors under study. It is observed that the main contributor is the dark current, by a factor of 20 greater than the second noise contributor. Hence this project will focus on the dark current when characterizing the CMOS noise of these two imagers.

The sensors were tested in a dark thermal chamber in order to characterize the thermal behaviour of the sensor and the dark current as the major noise source from the sensor. The first sensor was the 0.5MPix CMOS sensor, which did not allow removing FPN noise (a 0ms integration-time is needed) nor was binning capability available.

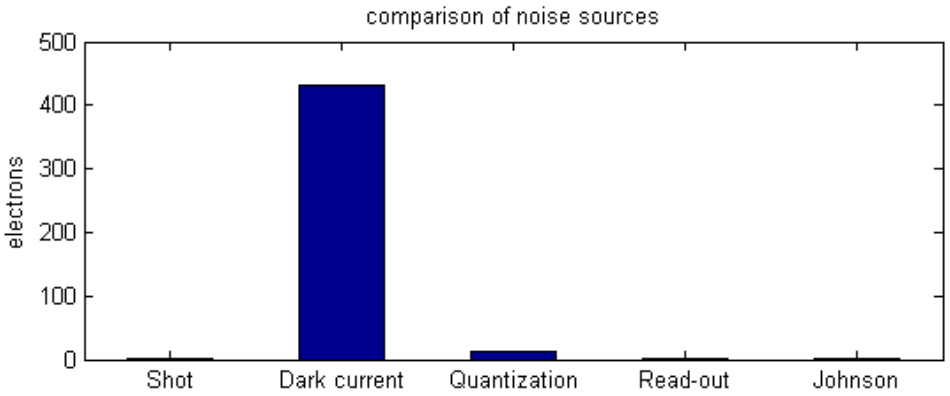

Figure 4.1: Comparison of the order of magnitude of different noise sources

Figure 4.2: Thermal chamber with the window covered at ISIS facilities

The second sensor was a 1.3MPix CMOS sensor, which allowed 0ms integration-time images in order to remove FPN noise a posteriori. Moreover 2x2 pixel binning was available. This feature increases the sensitivity and reduces the temporal noise.

4.3.1.1 Test Set up

The complete setup is placed within the ISIS thermal chamber.

Some considerations on the test procedure:

- A temperature sensor is attached to the image sensor using kapton tape.
- A measurement cycle comprises of the following set of measurements:
  - Time
  - Chamber temperature
  - APS CMOS temperature sensor
  - Acquisition of two images
- All these measurements shall be documented in a spreadsheet.
- Set the thermal chamber to +60 degrees C and let the temperature settle (15 mins), do a full measurement cycle.
- Set the thermal chamber to 0 degrees C and let the temperature settle (15mins), do a full measurement cycle.
- Step the temperature up to 40 degrees C in 10 degrees increment, at every increment let the temperature settle (15mins) and do a full measurement cycle.
- After temperature stabilization, two pictures are taken for different integration times (0ms = FPN image, 50ms, 100ms and 150ms) at every temperature for every sensor.

4.3.1.2 Results

Results for the noise measurements are shown in Table 4.3.

Table 4.3: Dark noise results from the thermal measurements, T = 25 deg

Sensor	**3σ [e-/s]**	**Mean[e-/s]**
0.5 MPix	1917	7167
1.3 MPix	1916	4516
1.3 MPix binning	636	4003
1.3 MPix FPN removed	958	342
1.3 MPix FPN removed, binning	615	308

It can be inferred that pixel binning reduces the noise dispersion to a third of its value. FPN removal could have been applied to a 0.5Mpix breadboard and the same trend could have been expected. After removing the FPN noise the average background ground is reduced by a factor of 13.2. Hence, FPN removal and primarily the pixel binning placed the 1.3MPix sensor ahead of the 0.5MPix in terms of noise performance.

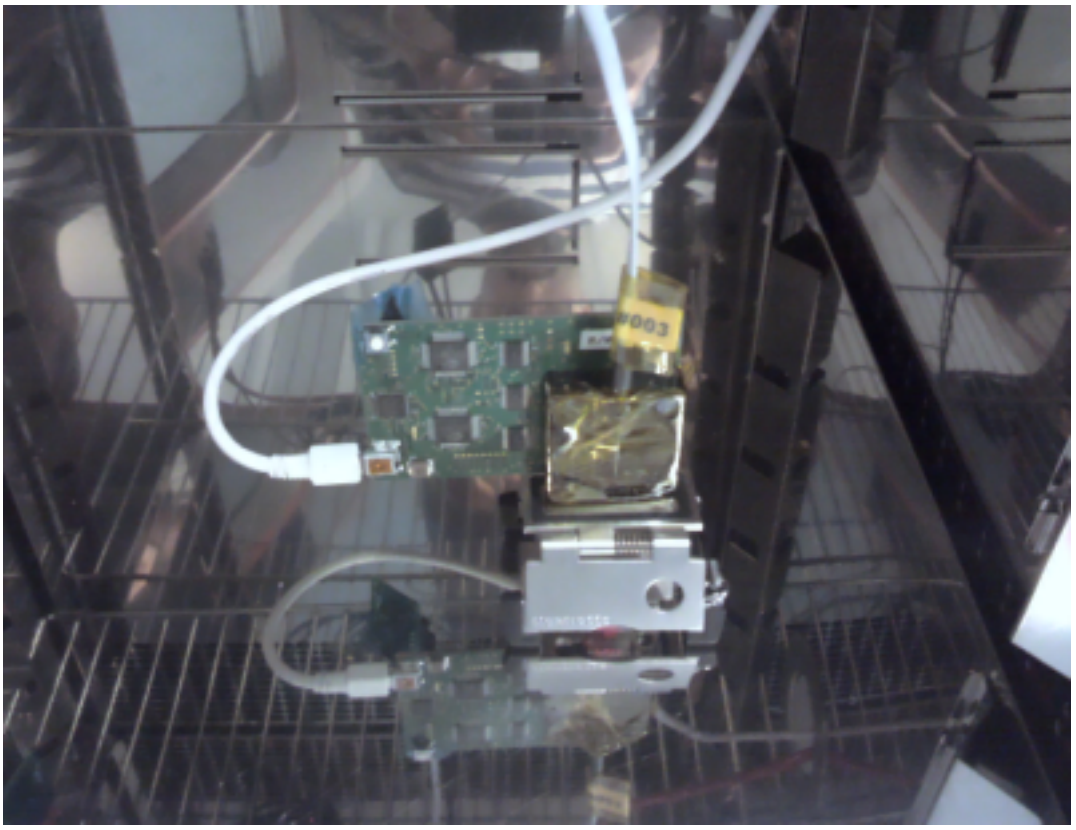

Figure 4.3: Test set up in the thermal chamber with the sensor covered to block all light

4.3.2 Signal-to-Noise Ratio

The first aim of the electro-optical tests was to calculate the signal to noise ratio of each sensor. This was done by acquiring images of a homogeneously illuminated surface. By measuring the output signal and computing the average signal value μ_{sig} to the standard deviation of the background σ_{sig}, SNR could be computed:

$$SNR = \frac{\mu_{sig}}{\sigma_{sig}} \tag{4.10}$$

The second aim was to estimate the minimum detectable intensity of each sensor. This was a key factor for selecting a final CMOS sensor as it determined the number of stars that Facet Nano would see, thus affecting the accuracy and the rate of success directly.

4.3.2.1 Test Set up

It was necessary to know the exact amount of light intensity that was coming out of the pinhole in order to estimate several sensor parameters such as the product QE*FF.

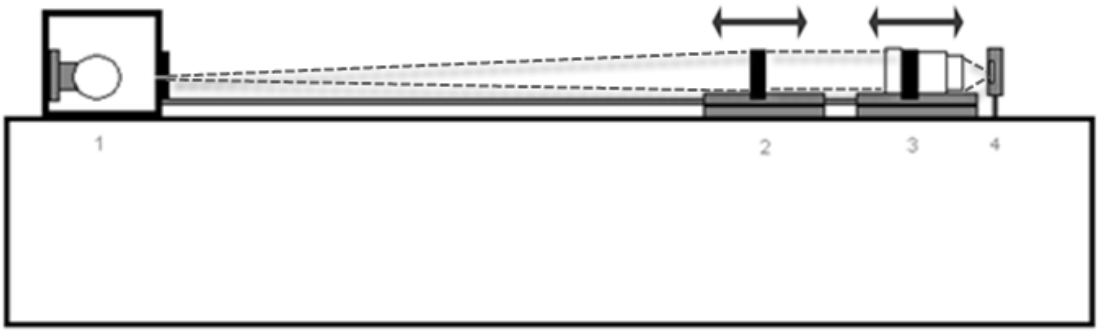

Figure 4.4: Schematic of the optical test set up
1) Light source 2) Collimator on translation stage 3) Optical triplet on translation stage
4) CMOS Active Pixel Sensor.

In this experiment, a 10W bulb was used as the light source for the experiment. The assumption of a black body radiation allowed a simple model of the light source. The intensity falling within the spectrum (200-1100nm) was estimated as one fifth of the total energy from the light source. However, this was only a theoretical approach and therefore had to be verified in the lab.

The 10W light bulb was enclosed in a metal cube. One surface of the cube had a pinhole, through which light was emitted. A Thorlabs PDA100A Si photo-detector was used to measure light intensity. It was situated at two positions:

- Immediately after the pinhole. Here the intensity coming out of the pinhole, in the spectral range of the detector, was measured.
- At the CMOS sensor position. Here the light intensity hitting the sensor was measured.

Light intensity was modified by making use of different ND filters. This allowed simulation of different star magnitudes. For very low intensities, the PDA100A photo-detector provided a gain factor from 0 to 70dB, in 10dB steps.

Measurements of the background light were carried out using different gain factors in the photo-detector. This will allow estimation of the offset produced by dark noise and background light.

The output of the photo-detector was displayed on an oscilloscope using a BNC cable to connect them. This output was given as an analog voltage. The light to voltage conversion could be estimated by factoring the wavelength-dependent responsivity of the silicon detector with the transimpedance gain as shown below:

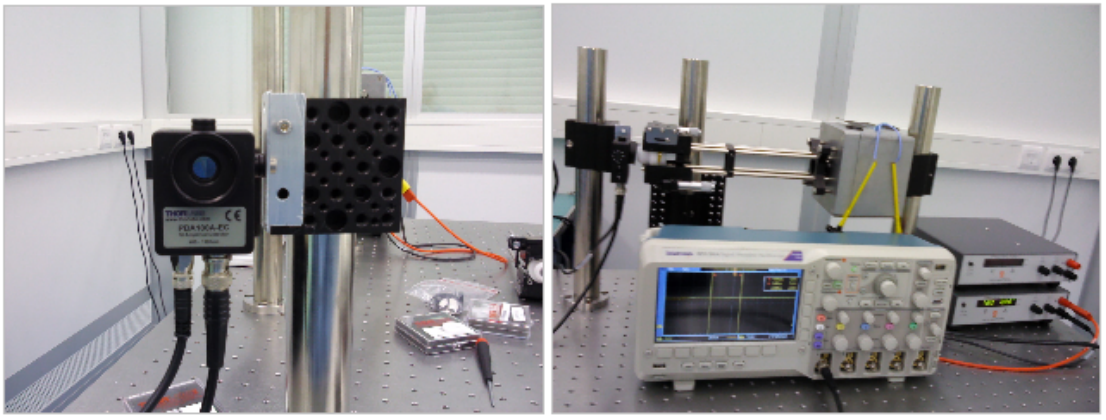

Figure 4.5: Thorlabs PDA-100A photo detector connected to the oscilloscope

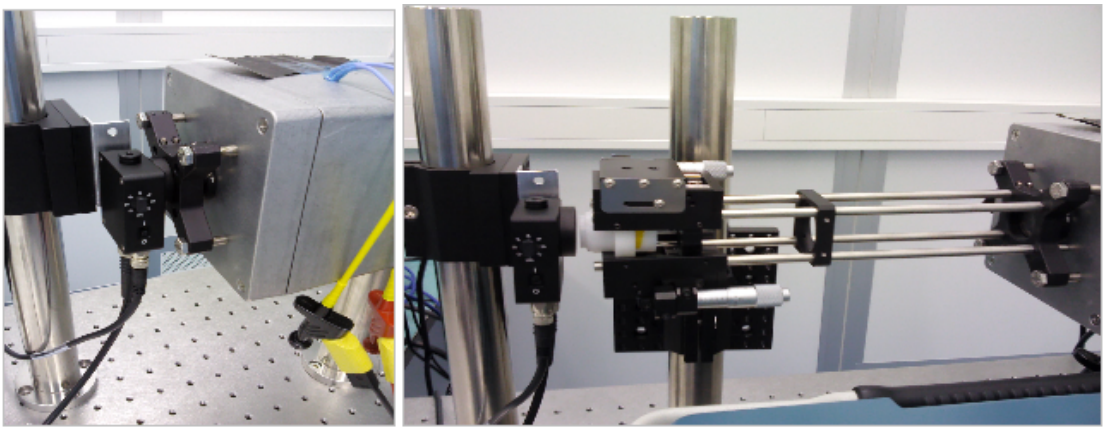

Figure 4.6: Left: photo detector immediately after the pinhole. Right: photo detector at the sensor position.

$$Output[V/W] = transimpedance\,gain[V/A] \cdot R(\lambda)[A/W] \tag{4.11}$$

Light characterization In these tests, the same light source setup as above was implemented. Firstly, a best form lens located at one focal length from the pinhole produced a collimated beam. The collimated beam was a parallel beam that simulated light coming from a star, an object located at virtually infinite distance. In order to simulate the intensity of a star, it was necessary to use ND filters to decrease the emitted light irradiance.

Secondly, the collimated light transmitted through a triplet lens system formed the star image in the focal plane. This is where the CMOS image sensor was placed. Therefore the spot size of the image would remain the same for every sensor, reproducing the same conditions in each test and making a fair comparison among sensors. Figure 4.4 describes the optical test set up.

Figure 4.5 shows the photo detector used to measure light intensity. Figure 4.6 includes a collection of pictures taken at the clean room showing the test set up for the light-source characterization.

Minimum detectable intensity Sensors were illuminated with the intensities listed in Table 6. All the sensors were capable of measuring up to 8.61E-07 W/m^2 (with a ND40 filter). Only the e2v EV76C454 sensor was capable of detecting lower intensities, specifically 8.61E-10 W/m (with a ND70 filter).

This elevated e2v sensor as the best candidate before assessing the signal-to-noise performance. Nonetheless this sensor was tested with micro-lenses, which improved the fill factor of the sensor by 70

Signal-to-noise ratio The correct method to estimate the SNR was to find the Standard Deviation (STD) around the maximum level, the "noise" around what is defined to be the "signal". Ideally, one would introduce in the sample a large homogeneous high intensity plateau, where (because of the present noise) some standard deviation would be measured. This maximum signal was represented by the light coming from the star simulated in the optical tests.

Unfortunately this is experimentally difficult, and it is also difficult to create a large volume with high constant intensities. The maximum signal in a real sample is usually very much localized (in some features of the objects) and intensity varies quickly around it, therefore it cannot be assumed that its STD is due to noise only. Fortunately, this was solved by computing the standard deviation of the dark images from the thermal tests.

"Noise" is defined here as three times the standard deviation of the sensor dark current. It should not be confused with other spurious signals, like the background noise contribution (mean of the dark current). To calculate the SNR it is not required to divide "signal" by "background", that would be something else (this can be called signal-to-background ratio SBR). The SNR measures the roughness or granularity of the image, and this is independent of the relation between signal and background.

4.3.2.2 Results

Thermal tests

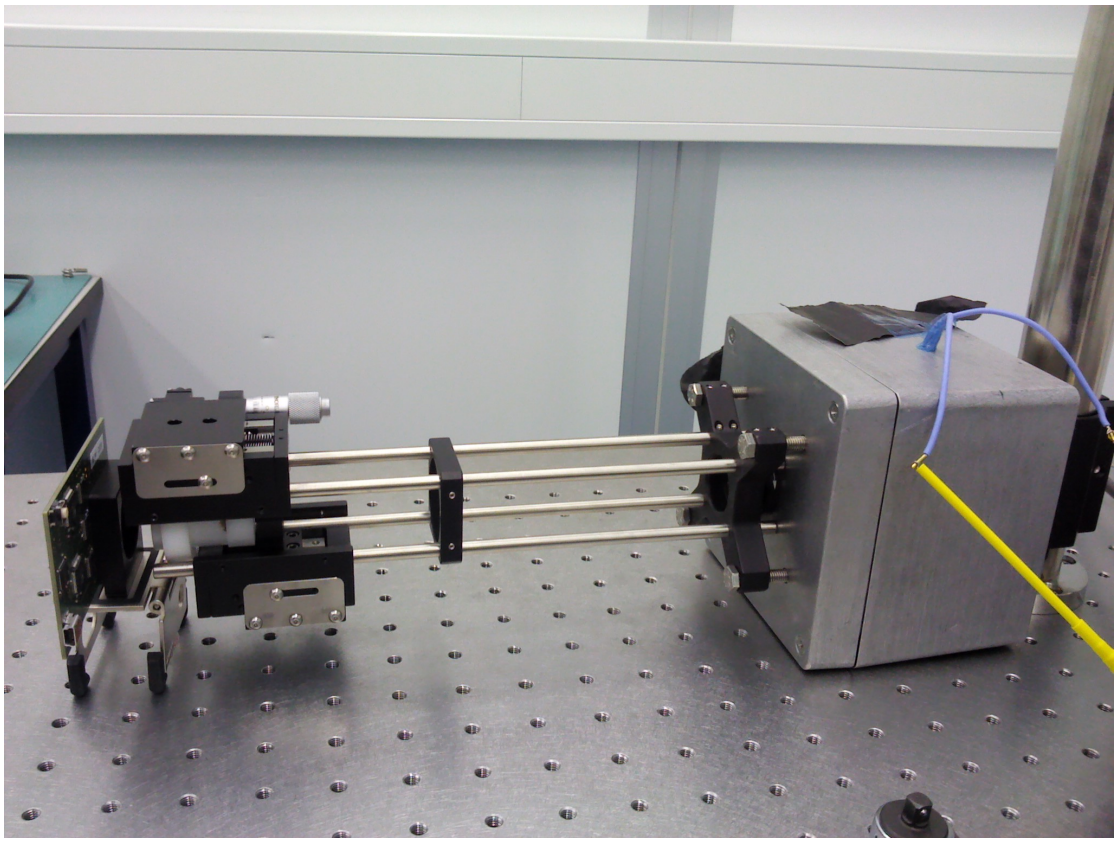

Figure 4.7: Optical test setup in the clean room for measuring SNR of CMOS sensors

Light source Table 4.4 shows the measured intensity values used to illuminate the sensors:

Table 4.4: Light characterization results, T = 23 deg

Photo-detector position	**Intensity [W/m^2]**
In front of the pinhole	2,13E-01
Sensor position	8,61E-03
Sensor position and ND40*	8,61E-07
Sensor position and ND70*	8,61E-10

**Note: ND40 (ND70) means that neutral density filters of 40 (70) dB were used.*

Signal-to-noise ratio Table 4.5 shows the signal-to-noise ratio values computed from the measurements taken during the optical tests in the ISIS clean room.

**Note: ND80 filter is equivalent to 80dB intensity attenuation. Thus the power received by the sensor equals a magnitude +5 star. Likewise, ND76 equals a M+4 star, ND70 equals M+2.5 and ND40 equals M-5.*

Table 4.5: SNR results from the optical measurements, T = 23 deg

SNR Table	**1.3 MPix**		**0.5 MPix**
ND Filter*	Binning ON	Binning OFF	Binning OFF
ND80	13.42	7.31	N/A
ND76	16.07	9.32	N/A
ND70	28.16	16.04	8.2
ND40	N/A	N/A	52

Table 4.5 shows that the minimum detectable intensity for the 1.3MPix sensor is the amount of light passing through the ND80 filter: 3.26E-10 W/m2. Likewise, this magnitude is 3.26E-09 W/m2 for the ND70 filter.

4.4 Trade-off Matrix

4.4.1 Trade-off Criteria

The selection criteria to assess the performance of the image sensors are:

1. Minimum detectable light intensity: the lower the intensity the better. It determines the number of stars the sensor is able to detect. The number of stars in the field of view is a critical parameter for centroiding accuracy and attitude determination and overall success rate.

2. Quantum Efficiency: the larger the better. It represents the number of electrons produced when a photon hits a sensor pixel.

3. Spectral response: It is linked to the Quantum Efficiency. It should have a high efficiency over the whole visible spectrum.

4. Full Well Capacity: the larger the better. It limits the sensor maximum response when it reaches the saturation level. A larger full well capacity allows detecting brighter stars without reaching pixel saturation.

5. Noise level: the lower the better. It affects directly the signal-to-noise ratio, which is proportional to the centroiding accuracy. It limits the minimum detectable intensity.

6. Bit depth: the larger the better. The number of bits used by the analog-to-digital converter determines the quantization noise and the minimum solvable value.

7. Operating Temperature Range: the larger the better. During launch and in space the sensor shall be exposed to extreme temperatures. This increases its reliability and durability.

8. Power consumption: the lower the better. This reduces demand on external sources.

9. Sensor availability: the higher the better. If a sensor is not available by the manufacturer, then it cannot be readily used in the Facet Nano star tracker.

10. Cost: the lower the better.

The killing criteria are: minimum detectable intensity, noise level and sensor availability. Sensor availability is a parameter dependent only upon the manufacturer. Noise level and minimum detectable intensity were discussed in the previous sections.

Trade-off table (1)						Criteria and relative weight factors									
					1 if potential killer criterion:	1								1	
					Relative criteria weight factors:	100,	40,	20,	40,	80,	10,	10,	10,	100,	10,
Options	final normalized scores	final scores	=0 if killed by killer reqt else = 1	normalized scores all options	scores all options	(killer) MinLight	QE	Spectral Response	Qsat	Noise	Bit depth	Temperature range	Power	(killer) Availability	Cost
						Relative compliance factors. Fill-out per-column ! Only the relative ratios in one and the same column are relevant.									
0.5 Mpix	**28**	118,73	1	28	118,73	1,	1,	1,	1,	1,	0,8	1,	1,	1,	1,
1,3 Mpix w/o binning	**33**	137,30	1	33	137,30	3,	1,2	1,2	1,2	0,5	1,	1,	0,5	1,	0,8
1, Mpix w/ binning	**39**	163,97	1	39	163,97	3,	1,2	1,2	1,2	1,5	1,	1,	0,5	1,	0,8

Figure 4.8: Trade-off criteria and scores

Trade-off table (2) filled-out automatically						Criteria and normalized weight factors									
					1 if potential killer criterion:	1								1	
					Normalized criteria weight factors in %:	23,8	9,5	4,8	9,5	19,0	2,4	2,4	2,4	23,8	2,4
Options	final normalized scores	final scores	=0 if killed by killer reqt else = 1	normalized scores all options	scores all options	(killer) MinLight	QE	Spectral Response	Qsat	Noise	Bit depth	Temperature range	Power	(killer) Availability	Cost
						Normalised weighted compliance factors in %									
0.5 Mpix	**28**	118,73	1	28	118,73	3,4	2,8	1,4	2,8	6,3	0,7	0,8	1,2	7,9	0,9
1,3 Mpix w/o binning	**33**	137,30	1	33	137,30	10,2	3,4	1,7	3,4	3,2	0,9	0,8	0,6	7,9	0,7
1, Mpix w/ binning	**39**	163,97	1	39	163,97	10,2	3,4	1,7	3,4	9,5	0,9	0,8	0,6	7,9	0,7

Figure 4.9: Trade-off Result

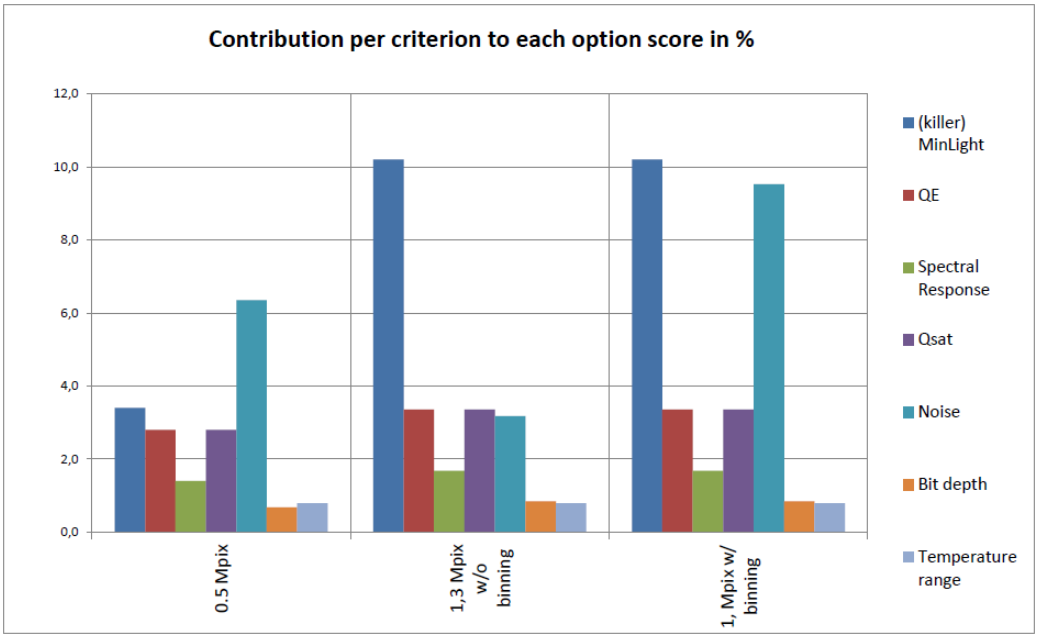

Figure 4.10: Contribution of each criteria to the trade-off

Figure 4.10 shows the contribution of each of the criteria to the options in the trade-off. It is observed that the minimum light intensity criteria removes the 0.5MPix sensor from the trade-off. As for the noise level, it swings the balance towards the 1MPix sensor with binning capability enabled.

The binning capability allows the sensor to reduce the noise, increasing the signal-to-noise level. Furthermore, it enlarges the dynamic range, making the sensor able to detect fainter stars. In combination, it provides a device capable of acquiring an attitude faster and more accurate.

4.5 Summary

The starting point for this chapter was a pre-selection of three CMOS image sensors. CMOS technology consumes less power than CCD and is more robust against radiation. All three sensors used global shutter technology, as rolling shutter was not as adequate for moving imaging instruments. It would possibly require more effort of additional software and hardware work to create a proper functioning instrument based on the rolling shutter design. This cannot fit into the remaining time for the project.

The results presented in this document demonstrate that the e2v EV76C454 has shown the best performance both under low and high light conditions. It is superior to the old Cypress LUPA300 in terms of noise (5.7 times lower) and signal-to-noise ratio (3.71

times better). It was also the only sensor of the 3 able to detect light intensities near the magnitude +5 range.

According to [32], the e2v sensor has survived gamma radiation exposure up to 120krad, making it suitable for space applications. It is also robust to proton radiation. The tests showed the dark noise performance of the sensor between exposure ranges of 5 to 20krad. As a reminder, these were just experimental tests highlighting the potential for space applications and not proper space qualification methods.

To account for the micro-lenses effect, simulation data provided by e2v engineers was used. Despite this, the simulation values agreed with other industry values [20] and with information provided by post-doc researchers.

It is recommended that a sensor without micro-lenses be ordered for further work in order to validate this simulation. After this validation, the next step consists of ordering a sample of e2v sensors without micro-lenses in order to test them under irradiation and compare the results with the experiments in [32].

A backthining process has been proposed by e2v engineers in order to avoid micro-lenses blackening due to space radiation. Backthinning of CMOS sensors provide certain advantages. These can include improved light sensitivity as a result of improved effective fill factor and sensitivity to UV light or low energy electrons [33].

Backthinning of image sensors is a very well established process for achieving high QE for high-specification space and science applications [33]. A side effect of the backthinning would allow application of an anti-reflective coating to improve QE [34].

Chapter 5

Facet Nano Software

5.1 Facet Nano Software Development

In this chapter we will review the software developed for the Facet Nano project. As we may recall, in the systems engineering chapter, we presented the Functional Block Diagram 3.1 and the N2 Chart 3.2. Both references identify explicitly the algorithms that will be executed within the Facet Nano system. These algorithms are:

1. **Image Processing**: this is the processing step where the raw image acquired by the sensor is read-out and filtered in order to remove noise, remove hot or dead pixels. This allows minimizing the number of false positives detected in the next step. Input is a raw image and the output is the Level-0 (L0) image.

2. **Star Detection**: this is the step where the L0 output is processed to detect the stars located in the FoV of the optics. It is based on a correlation technique against an artificial PSF. Afterward, the algorithm outputs the location of the star in the focal plane with sub-pixel accuracy, which is called the Level-1 (L1) output.

3. **Star Identification and Attitude Determination**: Based on the L1 output, this algorithm is able to match the detected stars against a star catalog stored in the system memory. The result is sent to the Attitude Determination algorithm, which, based on the reference system of the FN system, can estimate the attitude quaternion of the system (L2 output).

Figure 5.1 shows a high-level flow-diagram of the Facet Nano software, including a brief description of each module.

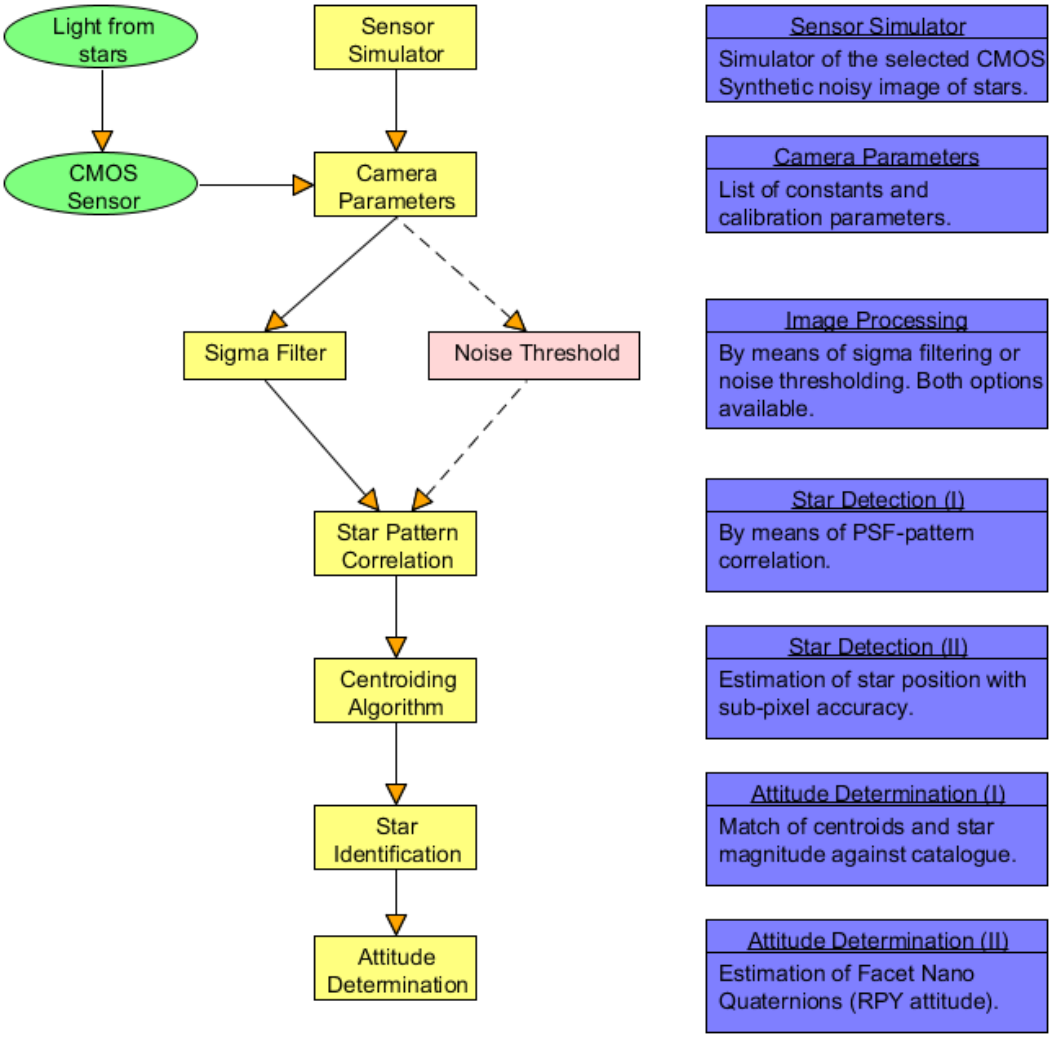

Figure 5.1: Facet Nano Software Diagram

Apart from this algorithms, a sensor simulator was built in MATLAB in order to debug, test and validate the software developed for this project. A more detailed explanation of this simulator can be found in section 5.5. and results of using the simulator to evaluate star tracker performance in section 7.5.

5.2 Image Processing

The image processing algorithm consists of reducing the noise produced sensor and in the readout electronics and maintaining the. Thereby, the effective SNR of the image can be improved and the centroiding algorithm will be more accurate. The algorithm receives the raw data as input. The output is an image with a lower amount of noise.

This module may be implemented using two different approaches: by means of filtering or by applying a noise threshold.

5.2.1 Noise Thresholding

This method implies defining a threshold. Any digital number of the raw image below this threshold is considered noise and is, therefore, set to zero. This is the simplest and quickest method of decreasing random noise in the image. Nonetheless it presents to important drawbacks:

- Noise threshold affects indistinctly both signal and noise data. As the star PSF is spread all over a wide area of the sensor, so does the intensity of the signal decreases with the distance to the center of the PSF. Therefore, the low-intensity part of the PSF signal will be inevitably removed by the noise threshold. Therefore, the accuracy of the centroid algorithm decreases due to a loss of information.
- Secondly, the dimmest stars may be removed by the algorithm if the threshold is set too high.

5.2.2 Filtering

Second method is filtering. It aims to enhance the image from the sensor before applying the centroiding algorithm. One implementation uses a Gaussian filter in order to remove noise. Is is applied in both directions succesively in order to have a circularly symmetric filter. Another implementation uses the sigma filter developed by Jong-Sen Lee et al [35]. It consists of a speckle suppression filter, designed for synthetic aperture radar (SAR) images, where granular noise is produced by interference and scatter. Figure 5.2 shows the result of this filter. We can see that the 2D PSF of the star is not circular due to optical aberrations. Before filtering, we can appreciate random noise all over the CCD array. Nonetheless, after applying the sigma filter, noise is removed and the PSF is not affected at all.

Comparing the performances of both methods (sigma filtering and noise thresholding), the filtering option if preferred over the noise thresholding as we are aiming to improve the star tracker accuracy as far as the low cost constraints allow it. Noise thresholding option would decrease the accuracy: it removes part of the star signal and does pass over the dimmest stars.

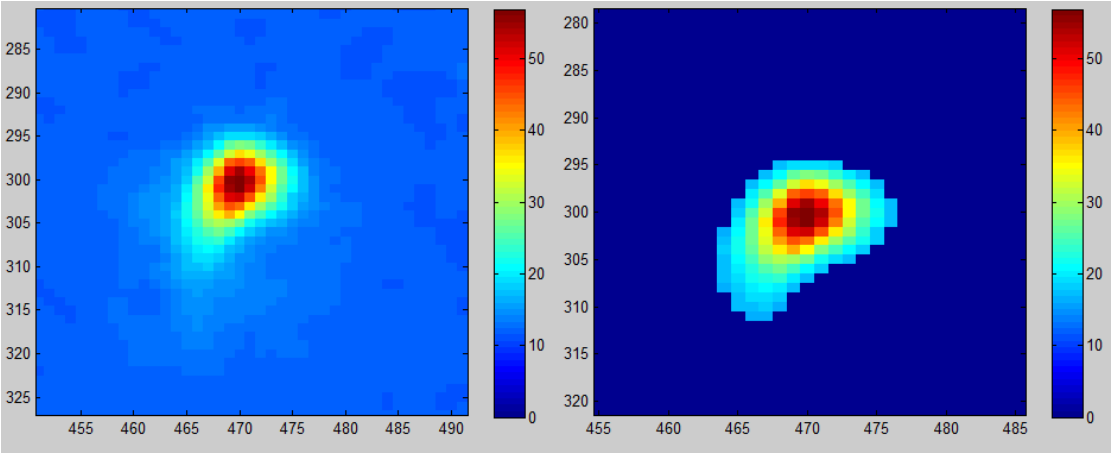

Figure 5.2: Left: Raw star before filtering; Right: Raw Star after applying sigma filter

Nevertheless, we will see in 5.3 *Star Detection* section that the image processing step, as conceived here, is skipped. Noise removal is no longer needed as long as the correlation step is implemented.

5.3 Star Detection

The star detection algorithm consists on two consecutive steps: the correlation step and the centroiding step.

5.3.1 Correlation Technique

This module represents the star pattern correlation box in the Facet Nano Software Diagram shown in figure 5.1. It aims at detecting stars in the digital matrix captured by the CMOS image sensor. After the image is filtered out by the filtering algorithm, the image can then be processed in order to find at least two stars. Three is the minimum number known variables needed to resolve the attitude problem. Two are the positions of the detected stars on the focal plane, and the remaining is the difference in magnitude between the two stars.

The detection could consider all the pixels with a digital number above zero. This method would be rely on the filtering technique to remoe all the noise. However, as the filter is not perfect, it cannot remove absolutely all the noise in the image. Therefore, selecting any pixel with a DN greater than zero would imply many false positives as star candidates.

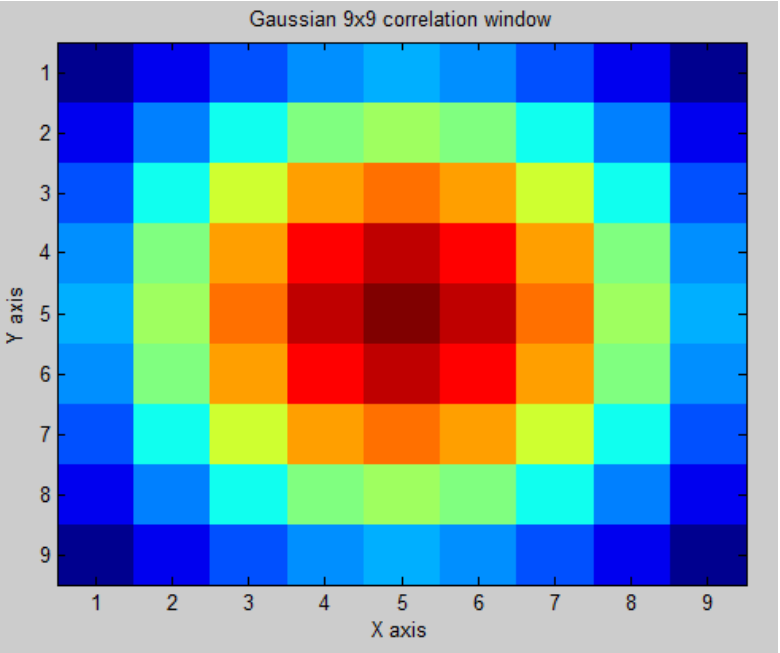

Figure 5.3: 9x9 Gaussian correlation window

It is therefore that we need to implement a method to detect stars that is robust enough against false positives. At the same time, the detection algorithm shall not discard true stars, i.e., it shall detect all the stars contained in the image. The designed module for this task is the detecStars.m file; its source code can be found in the annex of this document.

At this stage of the project, a two-dimensional Gaussian window is assumed for the PSF of a projected star onto the CMOS sensor. A finer tuning can be made to this Gaussian shape by including the effect of optical aberrations such as barrel and coma distortions. For example, in barrel distortion image magnification decreases with distance from the optical axis. Therefore, if a star is not located in the intersection between the sagittal and meridional optical axis, it will be affected by barrel distortion. The further the star is from the optical axis, the more distorted the PSF will be. Also, an example of simulated coma distortion can be found in figure 5.2. Barrel distortion is shown in figure 5.5.

5.3.2 Centroiding Algorithm

The centroiding algorithm is responsible for detecting the stars and computing the position of the star according to the image generated by the image sensor. It requires as input:

- The digital image captured by the CMOS sensor and processed by the image processing modules.
- The rough estimation of the star positions computed by the correlation module. This position will be used as a starting point by the centroiding module.
- The size of the centroiding window. This parameter shall be configured as a trade-off between processing speed (greater window size) and centroiding accuracy (smaller window size).

As a result, the output of the algorithm are:

- Star positions in the focal-plane of the star tracker-reference frame. These positions will serve as input for the attitude determination algorithm. Centroiding algorithm provides sub-pixel accuracy.
- Estimated star magnitude. The accuracy of the estimated star magnitude is highly dependent on the absolute radiometric calibration of the optical system. A detailed description of the absolute calibration can be found in section 6.3.

5.4 Star Identification and Attitude Determination

The star identification and attitude determination modules were developed by a third party. Is uses as input the centroid positions on the focal plane of the sensor and the estimated magnitude of the detected stars, computed by the star detection software. It requires at least two stars and their relative magnitude difference in order to identify the stars. The centroid positions and magnitudes are checked against a star catalog: a reduced version of the Hipparcos ESA's catalog. Hipparcos (High Precision Parallax Collecting Satellite) is an ESA mission launched in August 1989. It successfully observed the celestial sphere for 3.5 years before operations ceased in March 1993. Calculations from observations by the main instrument generated the Hipparcos Catalog of 118 218 stars charted with the highest precision. An auxiliary star mapper pinpointed many more stars with lesser but still unprecedented accuracy, in the Tycho Catalog of 1 058 332 stars. The Tycho 2 Catalog, completed in 2000, brings the total to 2 539 913 stars, and includes 99% of all stars down to magnitude 11, almost 100 000 times fainter than the brightest star, Sirius [36].

Hipparcos star catalog is used as the reference to match the detected stars. Nonetheless, we shall consider the catalog contains errors, as it is the result of several measurement taken years ago. Therefore, star positions in the catalog present certain error due to

Figure 5.4: ESA Hipparcos spacecraft [37]

the measurement uncertainty and to the movement of the stars produced since the position measurement. In spite of this error, for the project scope we shall consider the error contained in the star catalog is negligible compared to other errors such as the centroiding algorithm accuracy or the misalignment between reference frames.

Note that when the FN star tracker is integrated within a nanosatellite, the body reference frame of the spacecraft might be different from the star tracker reference frame. Therefore, a reference frame transformation shall be implemented in order to compute the correct attitude of the satellite. In fact, the main contributor to the attitude determination error budget is the misalignment between the star-tracker reference frame and the spacecraft body frame. The on-board computer receives attitude coordinates refereed to the star tracker reference frame. Then, OBC transforms these coordinates into the S/C body frame in order to estimate S/C attitude. At that time, body-frame attitude shall feed the attitude control subsystem of the spacecraft to point the satellite according to the satellite operations concept: solar panels perpendicular to sun vector, optical payload pointing a predefined target, communications antenna pointing towards a ground station, etc.

5.5 Electro-Optics Simulator

This module includes a complete model of the image sensor and optics. Using star positions and magnitudes, it generates the image outputted by one image sensor. This simulator is primarily concerned with a single aperture star tracker in order to simplify the development and scalability of the simulations.

It also takes into account optical aberrations (such as barrel distortion, shown in figure 5.5), sensor temperature and the configurable integration time in order to generate the final image. A complete noise model is implemented as well including, inter alia, dark current, quantization, thermal and shot noise. The noise model implemented in the simulator is described in detail in section 4.2.

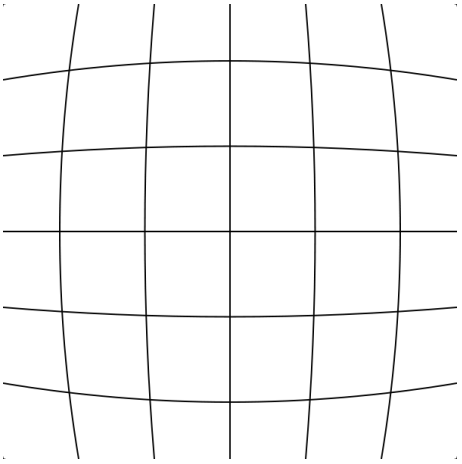

Figure 5.5: In barrel distortion, straight lines bulge outwards at the center, as in a barrel

Optics design of the star tracker is sketched in figure 1.1. In the simulator, the optical behaviour of 3 passive lenses made out of Fused Silica is modelled for the visible and near-infrared ranges of the electromagnetic spectrum. This two ranges are chosen because they match the spectral response of the image sensor given by the manufacturer. In other words, photons with a wavelength out of this range that reach the sensor are not detected and, therefore, they are not able to produce electrons.

The visible and NIR wavelengths go through the fused silica and hit the sensor not as point sources, but as spread areas due to diffraction. The amount of light that reaches the sensor is proportional to the star magnitude configured in the initial parameters. The detector then is able to produce an output current that is digitized. At this point is where all the noise adds to the signal. The amount of signal and noise that is produced depends on physical constants and the parameters that model the sensor. These parameters are:

- CMOS pixel fullWell capacity: accounts for the amount of signal that a pixel can detect before saturation during the exposure time.
- Quantum efficiency: accounts for the number of electrons produced for each photon reaching the sensor. It is usually measured alongside the fill factor.
- Fill Factor: this factor accounts for the amount of light that hits the sensor but reaches electronic circuit instead of the sensitive part of the sensor. Fill factor can be increased by the used of micro-lenses that focus the light in the sensitive areas of the focal plane. Nonetheless, micro-lenses are discouraged in space environment for optical applications due to their blackening when exposed to radiation during long periods of time.
- Noise parameters: mostly dark current, as shown in 4.1.
- Sensor temperature: it highly influences the amount of noise in the system. It is very related to dark signal.
- Analog Gain: sensor output is multiplied by a factor called analog gain.
- Pixel size: the higher it is, the better will SNR be. However, bigger pixels will make spatial resolution decrease, pushing centroiding accuracy down.
- Bit depth: it affects quantization noise and radiometric resolution. Higher bit depth leads to better star-magnitude estimation.
- Dynamic Range: represents the energy-gap between the dimmest and the brightest detectable stars. Sensors with a wide dynamic range are able to detect a bright and a dim star at the same time in the same image.
- Responsivity: it measures the inputoutput gain of the detector. Specifically, responsivity measures the electrical output per optical input. We are modeling responsivity as a linear function, except for the area below sensitivity point and the area above the saturation point. Figure 5.6 shows the three areas of the typical response curve of a photo-detector.

Once the signal is digitized, all the readouts from the different sensor pixels can be collected to compound the raw image. This raw image is sent then to the Camera Parameters module to start the software cycle designed for Facet Nano and shown in figure 5.1. The simulator was validated by comparing its output against star images produced in the laboratory. The comparison was done in terms of digital output, star shape and image noise.

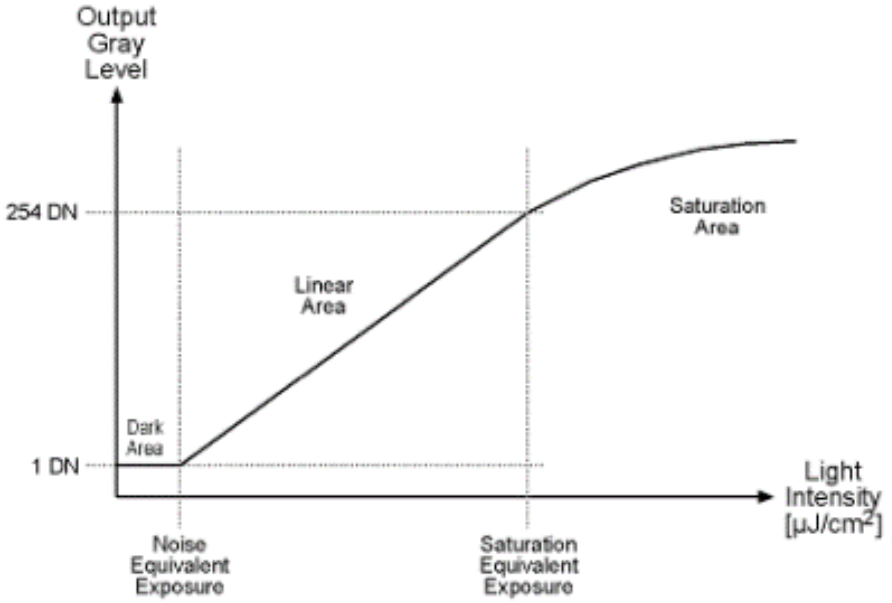

Figure 5.6: The response curve for a light sensitive sensor can be divided into three parts: the dark area, the linear area and the saturation area

5.6 Summary

In this chapter, the design and development of the Facet Nano software is presented: from the algorithms designed to be executed by the star tracker electronics, to the sensor simulator coded in MATLAB to be executed on ground in order to estimate the performance of the system.

The design of the software started in the diagram block shown in figure 5.1, where all the functionalities were identified from functional analysis described in section 3.3. The interfaces and the data exchange between functions were extracted from the N2 Chart shown in section 3.4.1.

In the following chapters, the Facet Nano software will be subjected to unit testing in order to assess the functional performance of the code (section 7.4). After that, it will be integrated to check the communication between modules and the right internal and external interfaces (section 7.5). Once the code is properly working, it will be ready to contribute to the Facet Nano end-to-end test (section 7.6).

Chapter 6

Calibration and Validation

Calibration and Validation (often referred as Cal/Val) corresponds to the process of updating and validating on-board and on-ground configuration parameters and algorithms to ensure that the product data quality requirements are met. To meet the baseline product quality requirements, a well-defined Calibration and Validation (Cal/Val) plan will be systematically applied.

6.1 Definitions

6.1.1 Calibration

Calibration is the process of quantitatively defining the system response to known controlled signal inputs. Hence the calibration process aims at determination of the sensor model parameters precisely.

6.1.2 Validation

Validation is the process of assessing, by independent means, the quality of the data products derived from the system outputs. The validation process aims to check the quality of the data. According to the accuracy results obtained in the validation process, the calibration procedure might be repeated. On the other hand, the validation process can be applied for the methods as well.

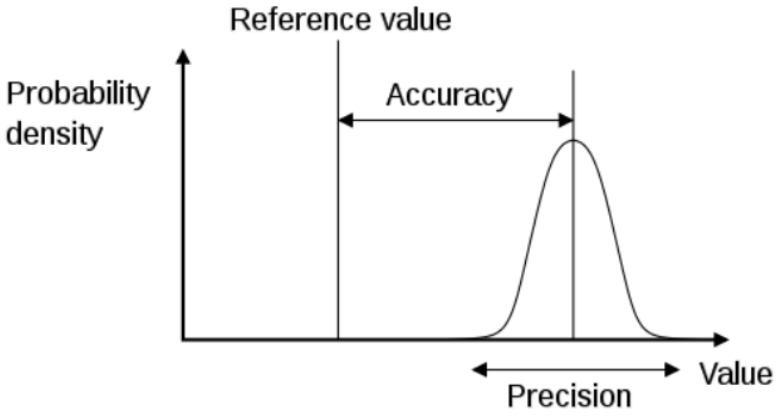

Figure 6.1: Accuracy and precision

6.1.3 Accuracy

Accuracy is defined as: "Closeness of agreement between a quantity value obtained by measurement and the true value of the measurand".

As indicated in figure above, accuracy indicates proximity of measurement results to the true value, precision to the repeatability or reproducibility of the measurement.

6.1.4 Precision

Precision is defined as: "closeness of agreement between quantity values obtained by replicate measurements of a quantity, under specified conditions".

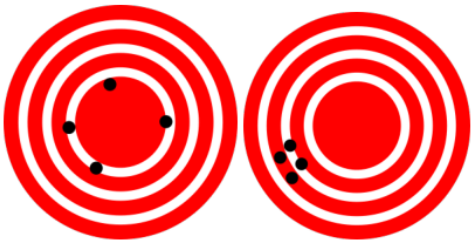

Figure 6.2: High accuracy, but low precision (left); high precision, but low accuracy (right)

6.1.5 Uncertainty

Uncertainty is a non-negative parameter characterizing the dispersion of the quantity values that are being attributed to a measurand (quantity), based on the information used. Where possible this should be derived from an experimental evaluation but can also be an estimate based on other information, e.g. experience.

6.2 Pre-Launch Calibration

6.2.1 Radiometric Calibration

Radiometric calibration aims at determining the parameters of the radiometric model for each pixel and each spectral channel of the instrument. This translates in reconstructing the curve of the correspondence between the digital count values of the instrument and the physical radiance measured at the sensor.

For this purpose it is necessary to know accurately the radiometric model of the Facet Nano instrument. Once known the radiometric model of the sensed image, it is necessary to define the methods for the estimation of the proper coefficients in order to retrieve the measured physical quantities.

The Facet Nano radiometric model describes the link between the raw digital count at the output of instrument and the equivalent radiance (R_{eq}) at the input of the instrument. The signal measured at the output of the detection chain can be modeled by:

$$V(p,d,R(\lambda)) = \Omega(b,p) \cdot \int_{\lambda} R(\lambda) \cdot T_{tel}(\lambda) \cdot T_{filter}(p,\lambda) \cdot S_{det}(p,\lambda,\Omega(p,d) \cdot R(\lambda)) \cdot \mathrm{d}\lambda \tag{6.1}$$

where:

- p is the considered pixel
- d is the considered detector
- R(λ) is the input spectral radiance (expressed in $W \cdot m^{-2} \cdot sr^{-1} \cdot \mu m^{-1}$
- T_{tel} is the spectral transmission of the telescope
- T_{filter} is the spectral transmission of the filter located in front of the pixel p
- Ω is the slid angle nder which the pixel p sees the output pupil
- S_{det} is the spectral sensitivity of the detector, for the current input radiance

In addition, we need to consider as inputs to the radiometric model the measured signal when the input radiance equals zero (dark current) and the possible alterations of the current instrument (e.g. instrument noise, stray-light).

The electronic gain and the quantization lead then to the following model of the digital counts coming out of the Facet Nano instrument:

$$X(p, d, R(\lambda)) = trunc[G(p, d, R(\lambda)) \cdot V(p, d, R(\lambda))] \tag{6.2}$$

where:

- trunc(x) is the truncation function, that produces the integer between 0 and 2^{10-1} the closest to x
- $G(p, d, R(\lambda)$ is the video chain gain, which might depend on input signal

The raw count result $Xp, d, R(\lambda))$ is the a 10 bit coded integer.

This radiometric model describes the link between the digital counts received on the on-board computer and the estimated radiances at the entrance of the Facet Nano.

The processing of the radiometric model aims to correct the measurement from:

- the natural offset (or dark signal)
- the pixel relative gains non-uniformity (or PRNU)

6.2.1.1 PRNU measurements

Photo Response Non-Uniformity (PRNU) accounts for the different response-level of each pixel to a uniform light input. Each pixel has a different sensitivity and, therefore, converts energy into electric current by a different factor than its neighbors. This effect produces noise on the raw image, affecting the centroiding algorithm accuracy and, possibly, increasing the number of false positive stars detected by the detection algorithm.

Radiometric tests were conducted to characterize the PRNU for the Facet Nano EOS. A cropped sample of the PRNU is shown in figure 6.4.

PRNU is usually measured using an integrating sphere: a light source that ensures uniform light distribution spread over its aperture. A diagram of a integrating sphere is shown in figure 6.3.

6.2.1.2 Dark current measurements

As we saw in Figure 4.1, dark signal is the major component of the noise in the system. Thus, measuring dark signal we will be able to characterize the system noise accurately enough.

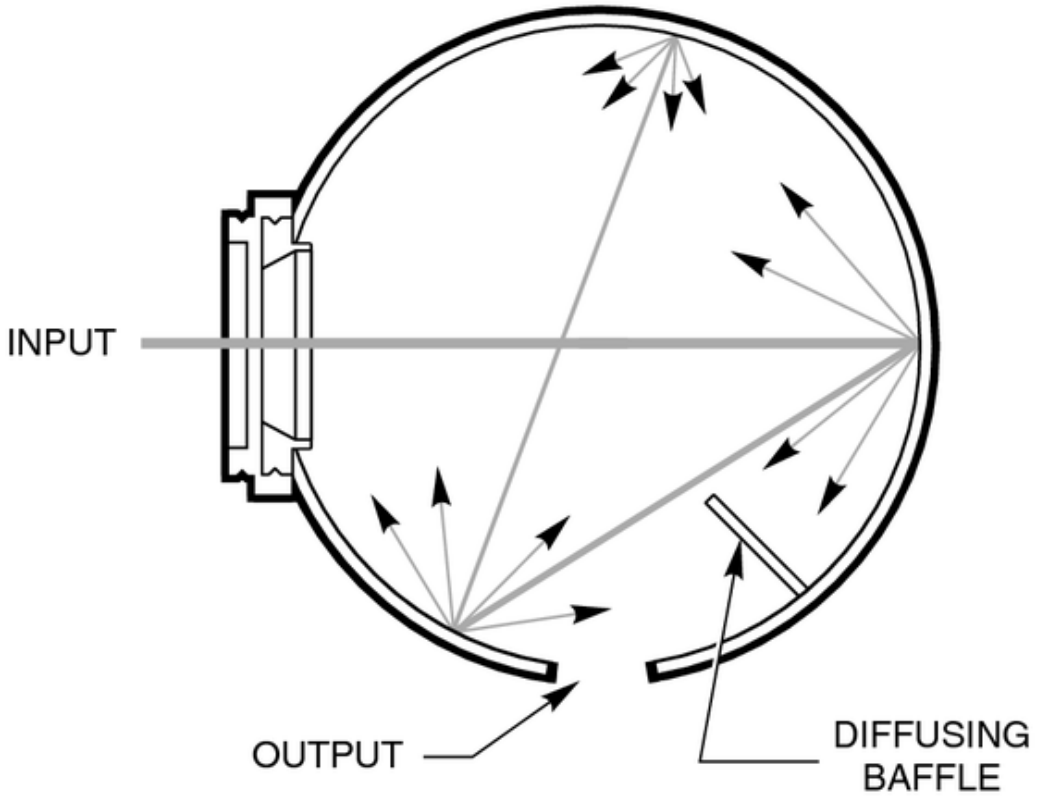

Figure 6.3: Integrating Sphere

Dark current is a result of thermally generated electrons, which are present even in the absence of light. The majority of these electrons are created in the forbidden energy gap between the valence and conduction band, where photons excite electrons between interface energy levels. The number of dark current electrons at a given temperature is provided by the manufacturer in the data sheet. In space, dark current noise can be further reduced by directly connecting the imager to a passive radiator or active cooling device.

Dark current is temperature-dependent. Therefore a thermal chamber is needed to reproduce the temperature environmental conditions in order to take measurements for different integration times and different temperatures. The set-up of the test to characterize the dark signal noise is shown n figure 6.5.

In this test, the sensor will be placed inside a thermal chamber. The read-out electronics are required, as we will acquire images with no incoming light. The result is the output current in absence of light, which is known as the dark current. There is also readout

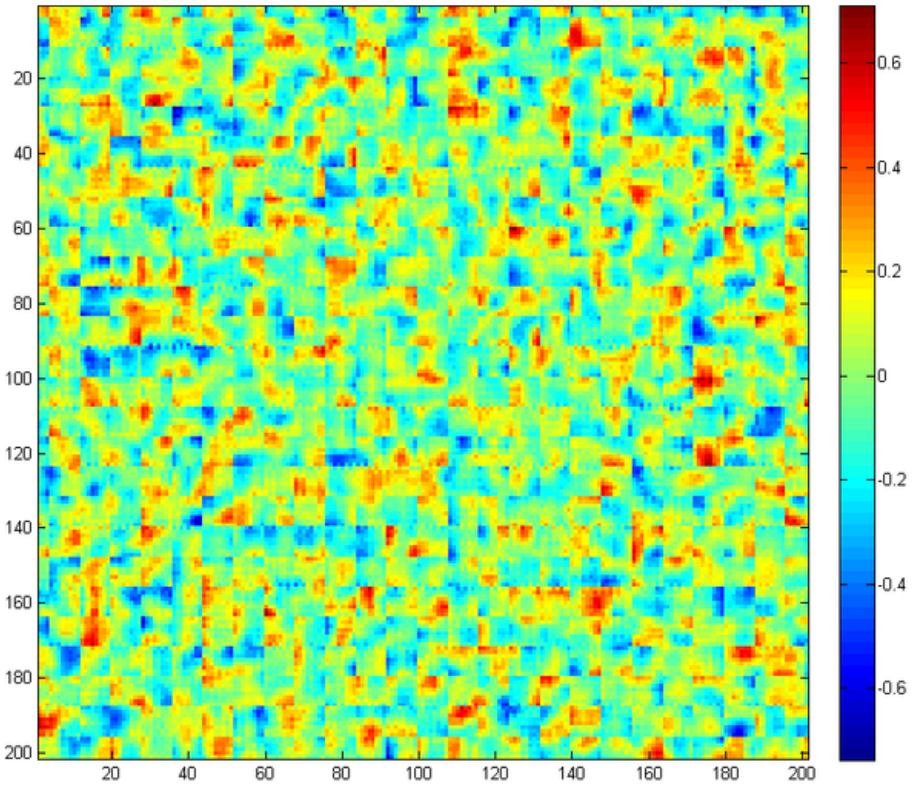

Figure 6.4: Sample case of Photo Response Non-Uniformity

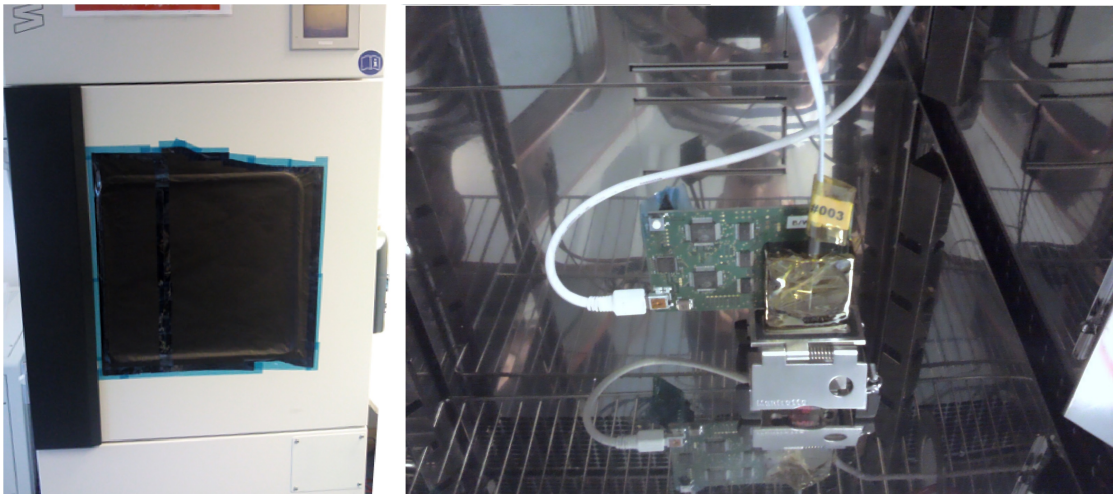

Figure 6.5: Left - Thermal chamber with window covered to avoid any light reaches the inside sensor; Right - Imaging module inside the thermal chamber, with sensor covered

noise in this signal, but as it is several orders of magnitude lesser than dark current (see figure 4.1), we can neglect it.

Thermal chamber will be covered so no light can reach the sensor Temperature is controlled with thermal chamber Integration time is controlled with acquisition software Output level is recorded for different combinations of temperature and exposure time After read-out, the data can be processed and dark signal data can be used for image post-processing on-board the star tracker

6.2.2 Modulation Transfer Function

Modulation Transfer Function (MTF) is an important figure of merit in focal plane array sensors, especially for accurate target positions such as star trackers. MTF is the magnitude component of the Optical Transfer Function (OTF). The OTF is the Fourier transform of the Point Spread Function (PSF).

The PSF is the resulting output of a focused light source. Stars are at such a great distance that for all practical purposes they can be considered points. When a star is observed with a telescope, the image will not look like the point source. There are two main reasons: First, the image will spread over a finite area due to aberrations in the optical system. Second, even in an aberration free system the image will spread over a finite area due to the diffraction limit of the system.

The diffraction pattern resulting from an illuminated circular aperture looks like a bright spot surrounded by a number of faint rings, as shown in figure 6.6. This pattern is known as the Airy pattern, and the central bright spot is called the Airy disk.

By integrating the irradiance over the central region of the pattern, most of the light energy is contained within the Airy disk. The remainder is distributed in the concentric rings, but depending on the intensity threshold for detection, the fainter rings may not be apparent. The size of the Airy disk depends on the wavelength of the light and the size of the aperture. (It also depends on the distance from the aperture, but in this context it is measured at the focal plane of a lens.)

Since diffraction is also wavelength dependent, each wavelength contributes a different diffraction pattern. This can affect the PSF output at different radial lengths from the optical axis. For a star tracker, a trade-off is required upon how much the energy of a point is 'concentrated' on the focal plane. The more concentrated, the fainter the stars that the star tracker will be able to detect. On the other hand, the more the energy is spread, the higher the accuracy of the centroiding algorithm will be, enabling even sub-pixel accuracy.

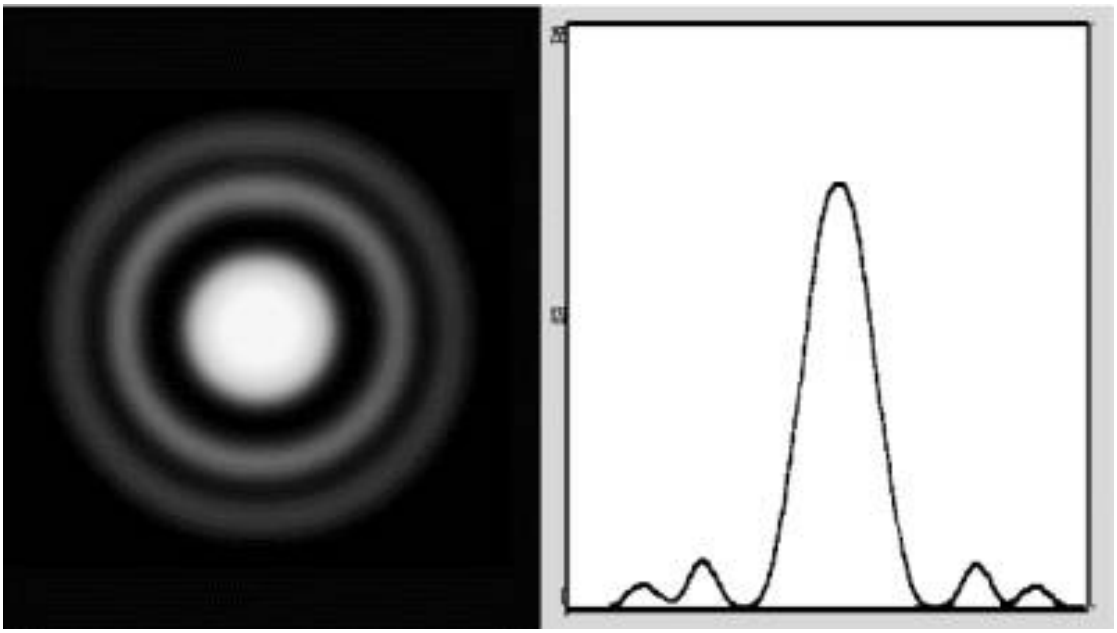

Figure 6.6: Left: Airy pattern, Right: Intensity distribution

6.3 Commissioning Calibration and Validation

Nominal radiometric calibration activities encompass:

- Dark Signal CAL/VAL
- Relative Gains CAL/VAL
- Absolute Radiometric CAL/VAL

Dark Signal CAL/VAL will be performed through the processing of images with the lowest possible incoming radiance. This is achieved by acquiring images during the eclipsed part of the orbit of ocean targets at night. Acquisitions are as well optimized in order to cover areas without lucent plankton (e.g. South Pacific CEOS test site) and to avoid full moon conditions. The output will be a DSNU matrix for each one of the integration times defined in the operations document, and an average value of the dark current. DSNU will be subtracted on-board to all images acquired by the star tracker. The average value of the dark current can be used to establish a threshold value and remove false-positive stars.

Relative gains CAL/VAL activities will be based on images of a homogeneously-illuminated target. This calibration is performed over the North Pole typically once per month, the instrument remaining nadir pointing. The output will be a set of NUC tables, one per integration time defined in the operations document. During the early phase of the

mission, the CAL/VAL team may consider that pre-launch radiometric characterization is good enough and may skip the commissioning radiometric CAL/VAL campaign.

Absolute radiometric CAL/VAL consists in determining the gain in order to convert the input signal into equivalent radiance at the entrance of the star tracker. The determination of the absolute calibration coefficients is coupled with the determination of the relative gains functions and it is performed on the same calibration measurement data-set. The output will be the absolute calibration coefficients, that will be used to estimate the magnitude of the stars in the FOV of the sensor.

6.4 Summary

In this chapter, the calibration activities of the Facet Nano project were discussed. Firstly, we focused on the distinction between calibration and validation, as well as the differences among accuracy, precision and uncertainty. Then, two different phases of calibration were presented:

- Pre-launch calibration: this is the calibration to be undertaken during the on-ground development and verification of the instrument. The radiometric model of the sensor was presented and the PRNU, DSNU and MTF measurement methodologies were described.
- Commissioning CAL/VAL: in this section, the CAL/VAL activities to be performed in-orbit are presented. It comprises radiometric characterization, as well as validation of the instrument by means of measuring different electro-optical variables: SNR, MTF, saturation levels, maximum and minimum detectable star magnitudes, etc.

Once the Facet Nano instrument is calibrated and validated in-orbit, it could start providing spacecraft attitude measurements, enabling operation modes were the precision of the attitude knowledge is key to achieve the required performance, such as imaging operation modes, or download operations with high-gain antennas of narrow beam-widths.

Chapter 7

System Verification

7.1 Verification Plan

The verification plan is a specification for the verification effort. It is used to define what is first-time success, how a design is verified, and which testbenches are written [10]. This chapter addresses the description of a verification plan for the Facet Nano star tracker system.

7.1.1 Why a verification plan?

A verification plan provides a strawman document that can be used by the unit under-test (UUT) design community to identify, early in the project, how the design will be tested. Early mistakes in the verification approach can be identified and corrected. A byproduct of the verification plan exercise is the revisit on the validity and definition of the requirements. This enforces the process of verifying those requirements, thus helping in the identity of poorly specified or ambiguous requirements.

7.1.2 Verification Processes

There are four fundamental methods for verifying a requirement [10]:

1. Inspection
2. Analysis
3. Demonstration (simulation)
4. Test

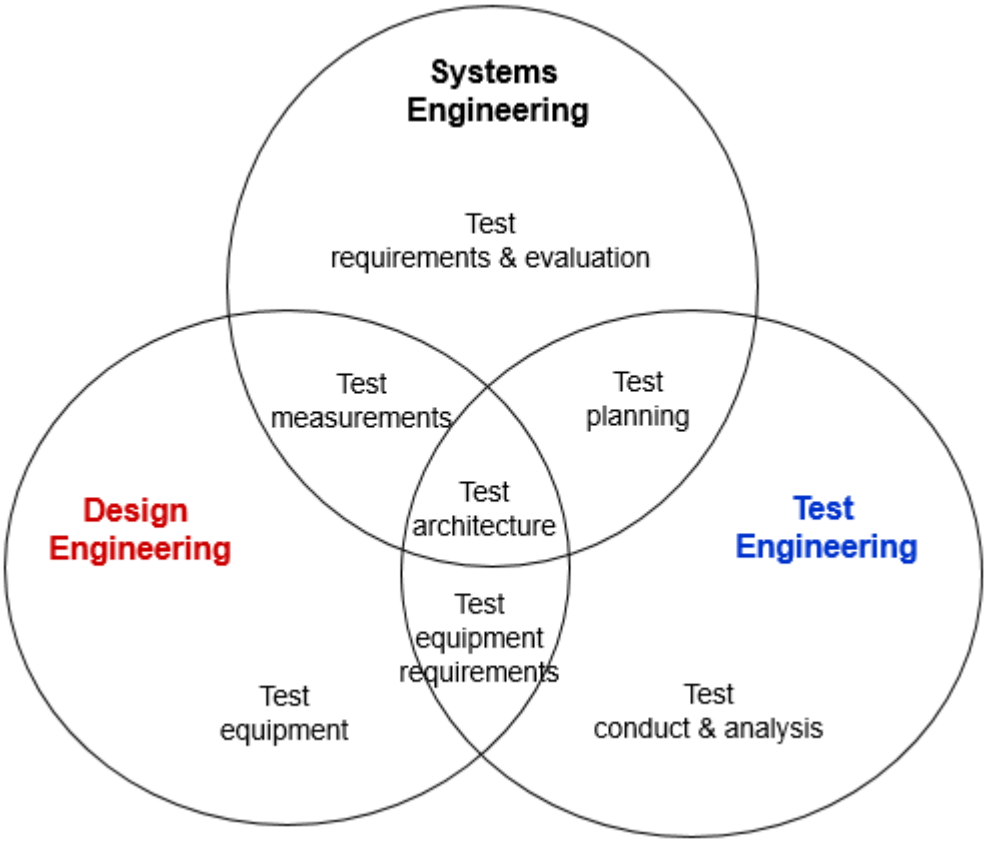

Figure 7.1: System test and evaluation team [10]

7.1.2.1 Analysis

Analysis is the evaluation of data by generally accepted analytical techniques to determine that the item will meet specified requirements.

Analysis techniques: systems engineering analysis, statistics, and qualitative analysis, analog modeling, similarity, and computer and hardware simulation.

Analysis is selected as the verification activity when test or demonstration techniques cannot adequately or cost-effectively address all the conditions under which the system must perform or the system cannot be shown to meet the requirement without analysis.

7.1.2.2 Inspection

Inspections determine conformance to requirements by the visual examination of drawings, data, or the item itself using standard quality control methods, without the use of special laboratory procedures or equipment.

Inspections include a visual check or review of project documentation such as, drawings, vendor specifications, software version descriptions, computer program code, etc.

Inspection includes examining a direct physical attribute such as dimensions, weight, physical characteristics, color or markings, etc.

7.1.2.3 Demonstration (simulation)

Demonstration determines conformance to system/item requirements through the operation, adjustment, or reconfiguration of a test article. Demonstration generally verifies system characteristics such as human engineering features, services, access features, and transportability.

Demonstration relies on observing and recording functional operation not requiring the use of elaborate instrumentation, special test equipment, or quantitative evaluation of data. For this project, we will rely on simulation results as a demonstration method to verify a requirement or specification.

7.1.2.4 Test

Test is a verification method in which technical means, such as the use of special equipment, instrumentation, simulation techniques, or the application of established principles and procedures, are used for the evaluation of the system or system components to determine compliance with requirements.

Test consists of operation of all or part of the system under a limited set of controlled conditions to determine that quantitative design or performance requirements have been met.

Tests may rely on the use of elaborate instrumentation and special test equipment to measure the parameter(s) that characterize the requirement. These tests can be performed at any level of assembly within the system assembly hierarchy.

The analysis of data derived from tests is an integral part of the test program and should not be confused with analysis as defined earlier. Testing is the preferred method of requirement verification and used when:

a. Analytical techniques do not produce adequate results,

b. Failure modes exist which could compromise personnel safety, adversely affect flight systems or payload operation, or result in a loss of mission objectives, or

Figure 7.2: Facet Nano development model, without baffle

c. For any components directly associated with critical system interfaces.

7.2 Baffle Performance: Star Tracker Availability

Availability is the fraction of all possible pointing directions for which the star tracker can provide 3-axis attitude information, when taking into account blinding and stray light from the earth, moon and sun.

The light baffle must suppress light from the moon at angles down to 40 degrees from boresight for the one orbit example. Over the lunar month it is unavoidable that the moon will be within the star tracker field of view for some short periods, however analysis of this has not been done as it will be very specific to the mission and is likely avoidable with intelligent pointing algorithms. The brightness of the moon varies with the phase, reaching a maximum of visual magnitude -12.6.

The light baffle must also supress from the limb of the Earth at angles down to 44 degrees from bore sight. The Earth is a very large light source, 132 degrees in angular diameter as seen from LEO altitude, with light conditions changing between eclipse and direct specular reflection of the sun. To make the problem tractable and use an approximate worst case, the reflection was modelled as uniform as seen from the satellites, with the whole earth disk having a brightness of 30% of the sun (the highest albedo levels experienced). By integrating over the portion of the earths disk that is within 90 degrees of the star camera board sight, and assuming that incoming light is reduced in intensity

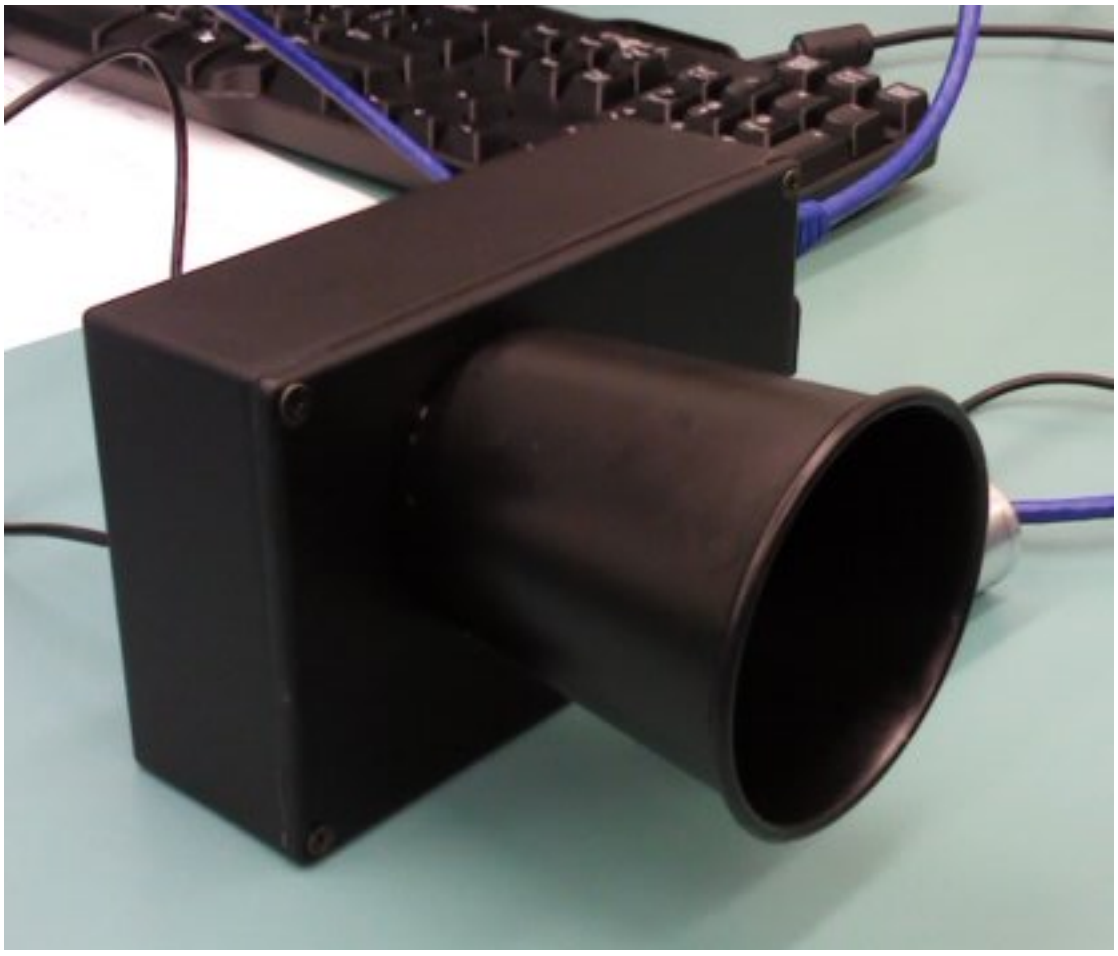

Figure 7.3: Facet Nano baffle integrated into the FN development model

by the square of the cosine for the angle away from bore sight, it is found that about 3% of the light coming from the earth will enter the aperture, making it equivalent in brightness to 1% of the sun, or an equivalent visual magnitude of -21.6.

7.2.1 Baffle Simulation

Using a simple model of a black, cylindrical optics housing, shown in figure 7.4, and stray light originating from 45 degrees off bore sight, it was found that the stray light is supressed by a factor of about 100,000 and the distribution of stray light is nearly homogenous across the detector. This will have a significant effect on star detection for stray-light sources 20 visual magnitudes brighter than the stars to be detected, so for +5 magnitude stars the suppression is sufficient for -15 magnitude sources.

Firstly, the optics usually has internal ridges, as shown in Figure 7.5, to increase the suppression. It is estimated to add 3 to 5 visual magnitudes suppression but will vary depending on the design.

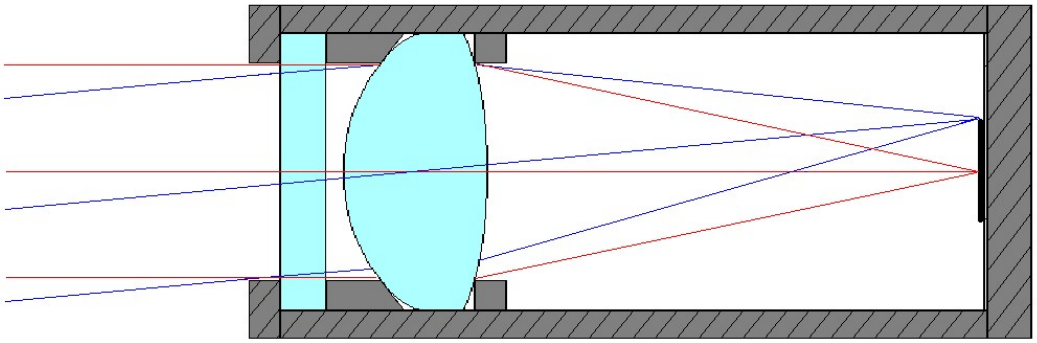

Figure 7.4: Cylindrical optics housing

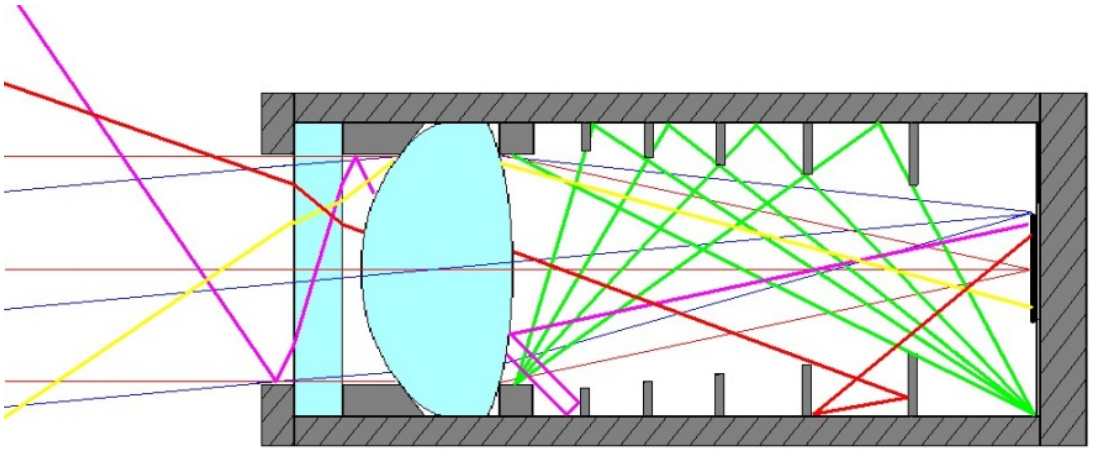

Figure 7.5: Internal Vanes

Secondly, an external baffle could be added as shown in Figure 7.6. Based on size of the optics mounting for Facet Nano, a 38mm long baffle can be accomodated within the maximum volume envelope. A simple, black, cylindrical baffle was modelled, indicating an additional 2.5 visual magnitudes stray light supression. The addition of vanes to this baffle will greatly increase the suppression, by an estimated 5 visual magnitudes.

By these means it is possible to avoid blinding from any sources for this orbit. The availability is thus equal to the sky coverage.

7.2.2 Sun Exclusion Angle Test

Baffles are the largest mechanical component of a star tracker. As technology is making star sensors smaller, baffer are still big. For this phase of the Facet-Nano project, a simple baffle design is chosen in this test. This decision is based on the multi-aperture

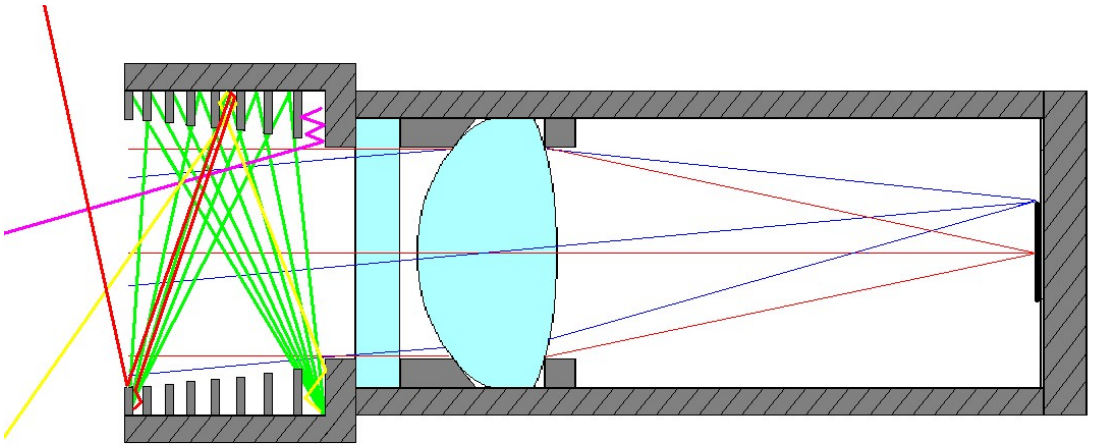

Figure 7.6: External baffle, with vanes

feature of the FN star tracker. With multiple apertures, the sun-blinding of one of the apertures is acceptable (100% availability). The baffle aspect is shown in figure 7.3.

The sun-exclusion angle test is performed in order to measure the impact of the of the background noise detected by the image sensor on the star tracker availability. The test consists of measurements of the background noise detected on the sensor for different incidence angles of the artificial sunlight. Sunlight is simulated under laboratory conditions. The test takes place in a cleanroom ISO7 environment due to the handling of optical elements. All the required equipment and instruments are located on an optical table isolated from ground vibrations, providing stability for all the measurements. In order to enhance the background noise measurement, all the sources of light must be removed and the test subject is surrounded by black aluminum to absorb as much reflected radiation as possible. Pictures of the FN development model are shown in figures 7.2 without baffle and 7.7 in the test setup configuration with the baffle attached.

As we can see in figure 7.7, light coming from the simulated Sun hits the star tracker. Some of the light is blocked by the baffle. The amount of blocked light depends on the incident angle of the sun-vector with respect to the optical axis of the system. In picture 7.2, sunlight is coming from a 0 deg angle, therefore, all the light shall be blocked by the baffle and background noise shall be near to zero. Figure 7.7 shows the average measured signal in the image sensor versus the incident angle on the sunlight. As expected, maximumstray-light is detected when the incident angle is approaching 90 deg. Minimum values are not zero due to dark current, which is the output of the detector when the input light is zero. From these results, we can extract that background noise remains below the intensity of the dimmest detectable star when the Sun angle is lesser than 30 deg.

Figure 7.7: Facet Nano baffle and development mode, surrounded by black aluminum foil to absorb reflections

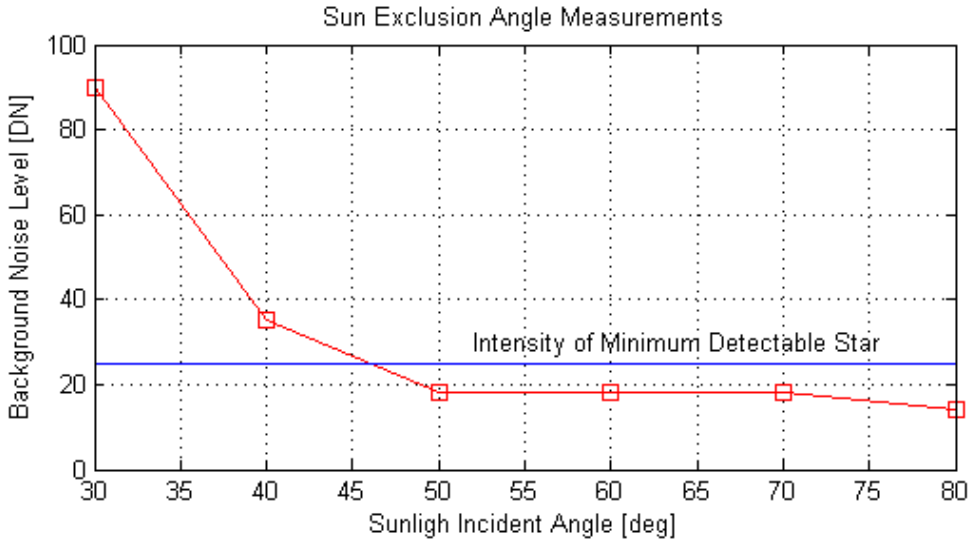

Figure 7.8: Sun-exclusion angle, test results

7.3 Radiation Test

Radiation test will measure the ionizing radiation effects on the e2v image sensor and selected optical materials from Co-60 gamma irradiation. It is necessary to verify thex performance of the sensor as it has not been rated for space applications. Furthermore this specific sensor is designed with microlenses made of an organic material. The information on the organic material could not be disclosed by the supplier but some polymers are known to have been used. Commercial optics will also be tested to determine their viability.

The objective of the radiation test is to determine if these critical components can survive the operational lifetime requirement. This is based on the 3 year mission life. Radiation levels are estimated based on this period. The 0.8mm minimum aluminium shielding that is provided by the CubeSat side panels amounts to 73 krad(Si) exposure on a Silicon device. The design should have as much as shielding as possible. After considering the likely minimum material mass of the FN structure, 2.5mm of aluminium shielding is a minimum constraint. This is equivalent to 5 krad(Si) TID. Since there is an inverse exponential relation between shielding thickness and TID, the 2.5mm thickness was considered a good all round choice for a wide variety of uses cases in LEO orbit.

Therefore the criteria that should be fulfilled by this test are:

1. The sensor must be able to survive the radiation levels for a 3 year mission.
2. The noise performance must meet the criteria to allow sufficiently high signal threshold at EOL.
3. The degradation of the microlenses must allow the SNR at EOL to meet the performance requirements.
4. The degradation of the optical elements must allow the SNR at EOL to meet the performance requirements.

Table 7.1 shows the possible parameters that should be tracked before, during and after testing.

The objects under test will be:

1. PMMA
2. Polycarbonate
3. Zeonex

Table 7.1: Parameters to be monitored before and after the test

Optical Materials	**Image Sensor**
Optical Transmission	Readout noise
Visual darkening	Dark current noise
	SNR
	Photo response non-uniformity
	DC voltage level output
	Power consumption
	Number of hot pixels and dead pixels

4. Fused silica (Suprasil from Heraeus)

5. e2v 1 Megapixel Monochrome sensor (EV76C661)

PMMA and polycarbonate represent the most common organic material used for microlenses. As mentioned earlier there is no available data on the microlens material of the e2v sensor aside from its organic origin, so the results from the materials is an attempt to give a baseline for separating the performance of the microlens from the sensor electronics.

Fused silica lenses have been proven to have good radiation resistance properties and are commercially available. N-BK7 glass is a common commercial glass used in a lot of high end applications. It can be compared against known results though more data is required for the expected TID levels.

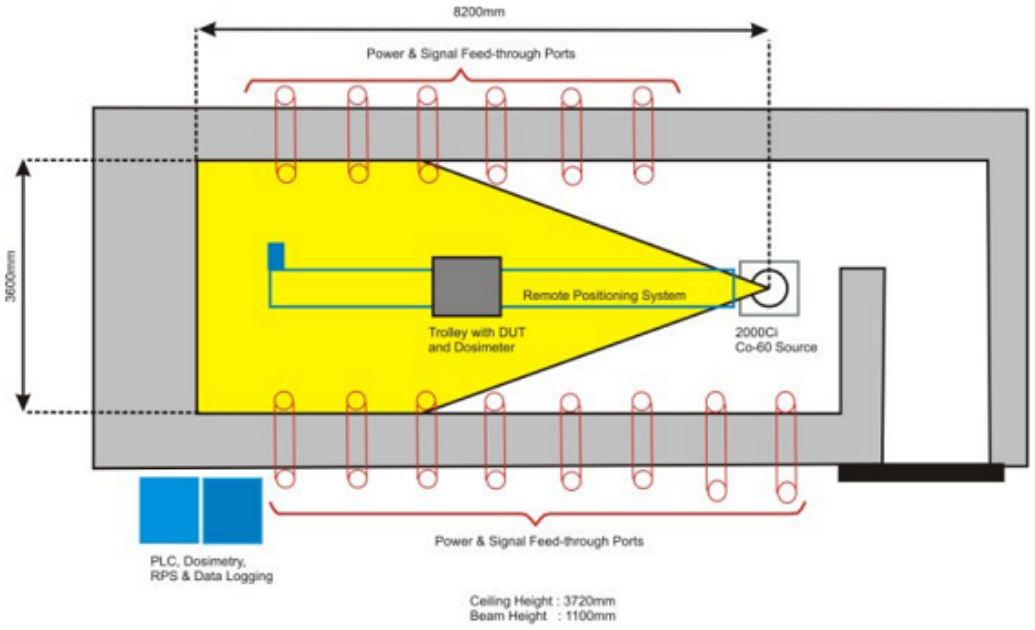

Figure 7.9: TID Facility Schematic

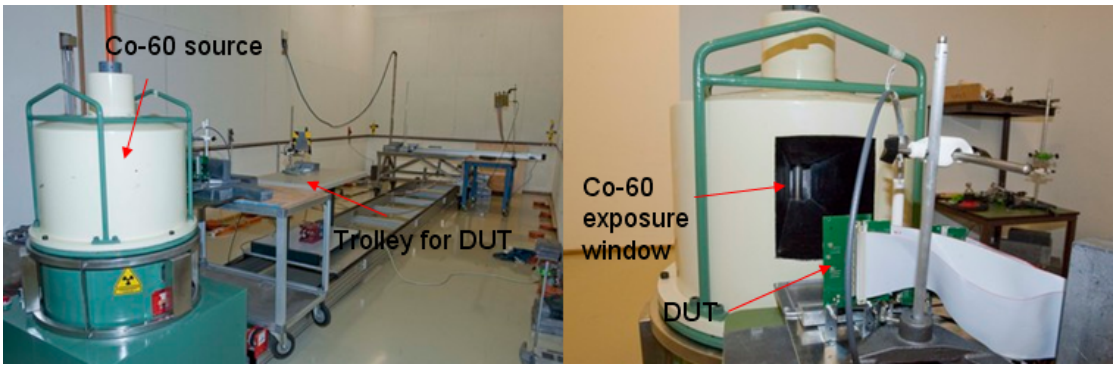

Figure 7.10: ESTEC TID Facility [38]

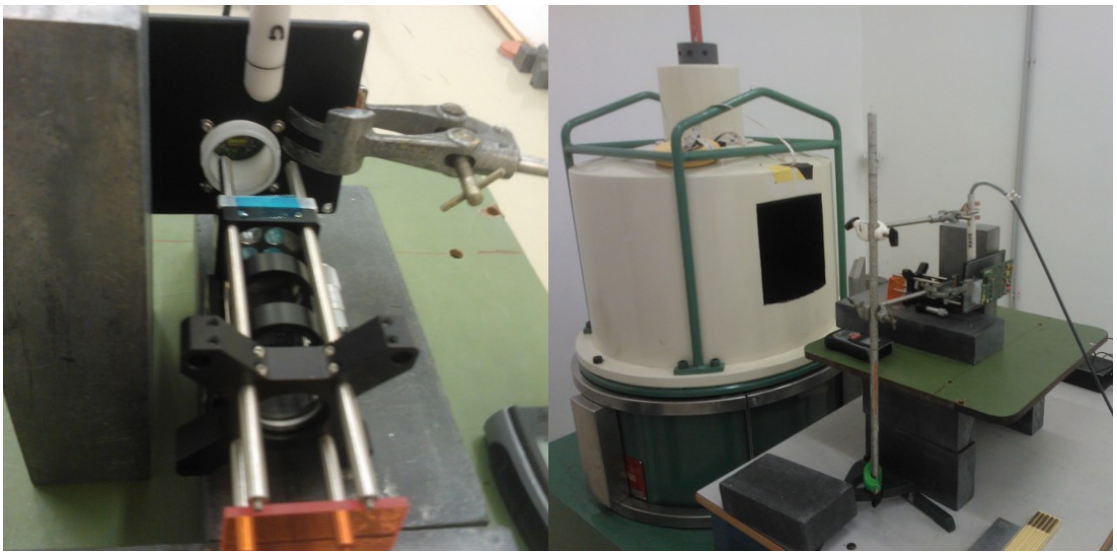

Figure 7.11: Radiation TID Test Set-up at ESTEC Facility

Figures 7.9 and 7.10 show the Co-60 irradiation facility at ESTEC [38]. It allows variation of the dose rate from a collimated source. The samples to be irradiated shall be placed with their front surface perpendicular to the irradiation direction.

Figure 7.11 shows the set-up of the radiation TID test set-up at the Co-60 facility at ESTEC. The sensor is located in front of the Co-60 source. Between the sensor and the source, a series of optical materials are located to study their transmisivity degradation after a TID exposure of several krads. The readout electronics of the sensor is protected by a 3mm thick aluminum layer, which is representative of the shielding that will provide the spacecraft platform that hosts the instrument.

Table 7.2: Minimum and maximum dose rates

	Min dose rate	**Max dose rate**
Dose rate (rads/min) (H_2O)	0.6	239.375
Corresponding radiated area (cm^2)	360x360	14x14

E2V demo kit is approximately 12 x 7 cm. The ESTEC Co-60 facility has a calculator [39] to help adjust the dose rate and irradiated area. To accommodate the 3 sensors and optical material for the 4 krad(Si) TID, 7.13 shows the initial conditions for the test setup. Firstly a conversion is required between the Silicon and Water based dose rate. For a silicon device:

$$4krad(Si) = 4 * 1.11423 = 4.45692 \approx 4.5krad(H_2O) \tag{7.1}$$

The 75 rads per minute dose rate (4.5 krad H_2O over 1 hour), results in a 30x30 cm^2 irradiated area.

The TID steps were obtained from SPENVIS calculations. An assumption to note about SPENVIS is that the default AP-8 and AE-8 models were not used for the trapped particle calculations. These models are based on data that was collected from the 60s and 70s. It was decided to use the models based on the CRRES satellite that was launched in the 90s specifically for gathering data on energetic particles in the Earth magnetosphere. The data should be more up to date and accurate. As only LEO orbit operation is considered, only certain radiation sources in SPENVIS were calculated. In this case only the basic Trapped proton and electron fluxes were calculated. Solar particle fluences and Galactic cosmic rays were not included, assuming a stable magnetic field.

7.3.0.1 Radiation Test Results

The radiation levels are expressed in total dose, and are provided below. The values are a result of:

1. Simulations in SPENVIS, at an altitude of 800km, with an inclination of 90 degrees
2. TID test at ESTEC Co60 facility

In the SPENVIS simulation, the sensor is assumed to be sensor fully covered by an aluminum sphere of 2.5mm thickness. There is no consideration yet for the radiation going through the optics.

For a lifetime of 3 years, the system level values, for different thickness of shielding are expected to be:

1. 0.1mm: 157 krad
2. 1.0 mm: 37 krad

Dose Rate to Distance & Area Calculator

Input required date & dose rate. If no date is input then the current date is used

Input Day	19	
Input Month	2	
Input Year	2012	
Required Dose Rate	75	Rads/min (water)
	185	Days Since Reload (18/08/2011)
Distance	70.93	cms (Practical min 40 cm Practical max 815 cm)
Rate at 1m	37.73	Rads/min (water)
Activity	79.72	TBq
Uniform Irradiation Area	30.04 X 30.04	cms X cms
	Calculate	

Figure 7.12: Test setup initial conditions, obtained from [39]

Material	^{60}Co Dose Rad[Mat]/(γ/cm^2)	Conversion for Rad[H_2O]	Conversion for Rad[TLD]	Conversion for Rad[Si]
Water	5.93930E-10	1.00000	0.87776	0.89748
CaF_2:Mn	5.21326E-10	1.13927	1.00000	1.02247
Silicon	5.33039E-10	1.11423	0.97803	1.00000

How to use this table:
1. Determine the units of the dosimetry. (Ex. – Alanine dosimetry results in Rad[H_2O].)
2. Find that entry in this table. (Ex. – Row 1 give the dose in Rad[H_2O].)
3. Find the appropriate conversion factor in table. (Ex. – I want to convert to Rad[TLD], so the conversion factor is 0.87776.)
4. Multiply to dosimetry results by the conversion factor.

CAUTION: These conversion factors are only valid in a ^{60}Co γ-field.

Figure 7.13: Conversion between Si & H2O radiation dose; taken from [40]

3. 2.0 mm: 13 krad

4. 2.5 mm: 10 krad

Technical information provided by Schott, a UK company that produces glass, describes an experiment done with BK7 glass. It was irradiated with 10krads of Co60. As a result

the average transmittance dropped to 73.6% on average, for the visible spectrum (from 350 to 900 nm). Roughly, that means a drop of one fourth in the incoming signal. The result is shown in Figure 7.14.

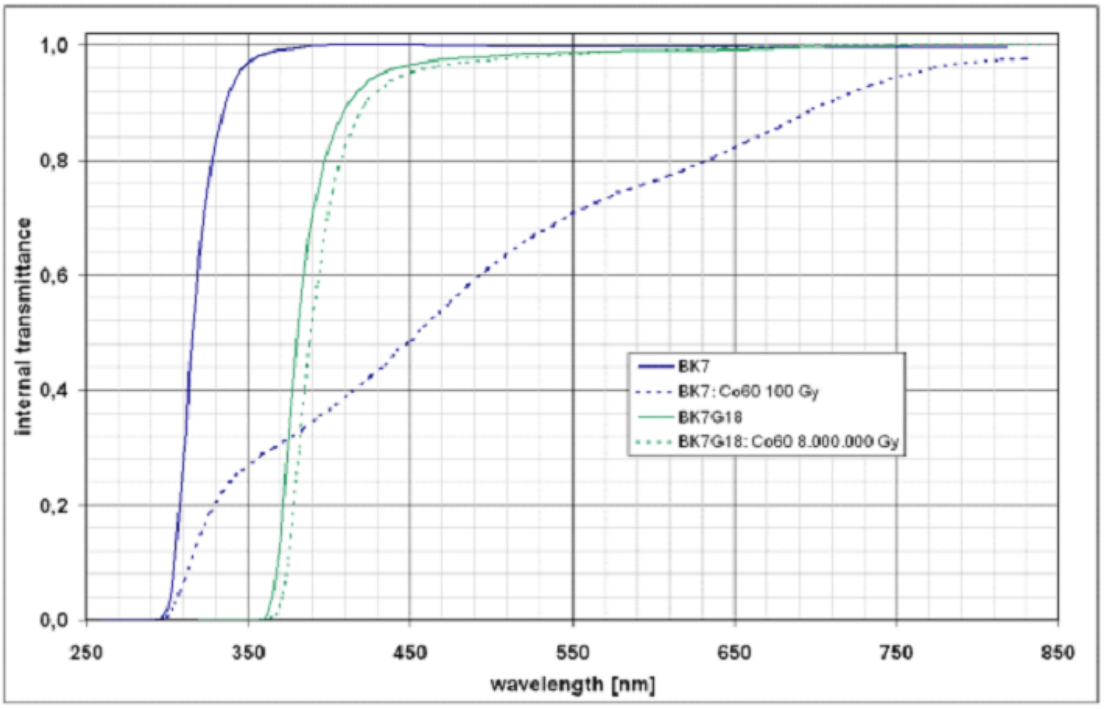

Figure 7.14: Transmittance of BK7 before and after irradiation

As the degradation is wavelength-dependent, a simulation was carried out to evaluate the effect on a possible microlens array made out of BK7. The input light is the black-body spectrum coming from a magnitude 5 star at 5000K. Result of the simulation is shown in Figure 7.15.

Total intensity hitting the sensor after irradiation, in the 350-900 nm band (visible), is 24.75% lower (BK7 glass). Losses are caused by transmittance degradation.

Regarding the TID test, there is a responsivity decrease after the irradiation. It is mainly limited to short wavelengths and responsivity global shape is still conserved. Figure 7.16 shows the results.

Regarding dark current, new measurements were taken. An increase in dark current after 10krad exposure is 2%. This effect is consistent with the results obtained at Open University when radiating other E2V image sensor with a Co60 source [41].

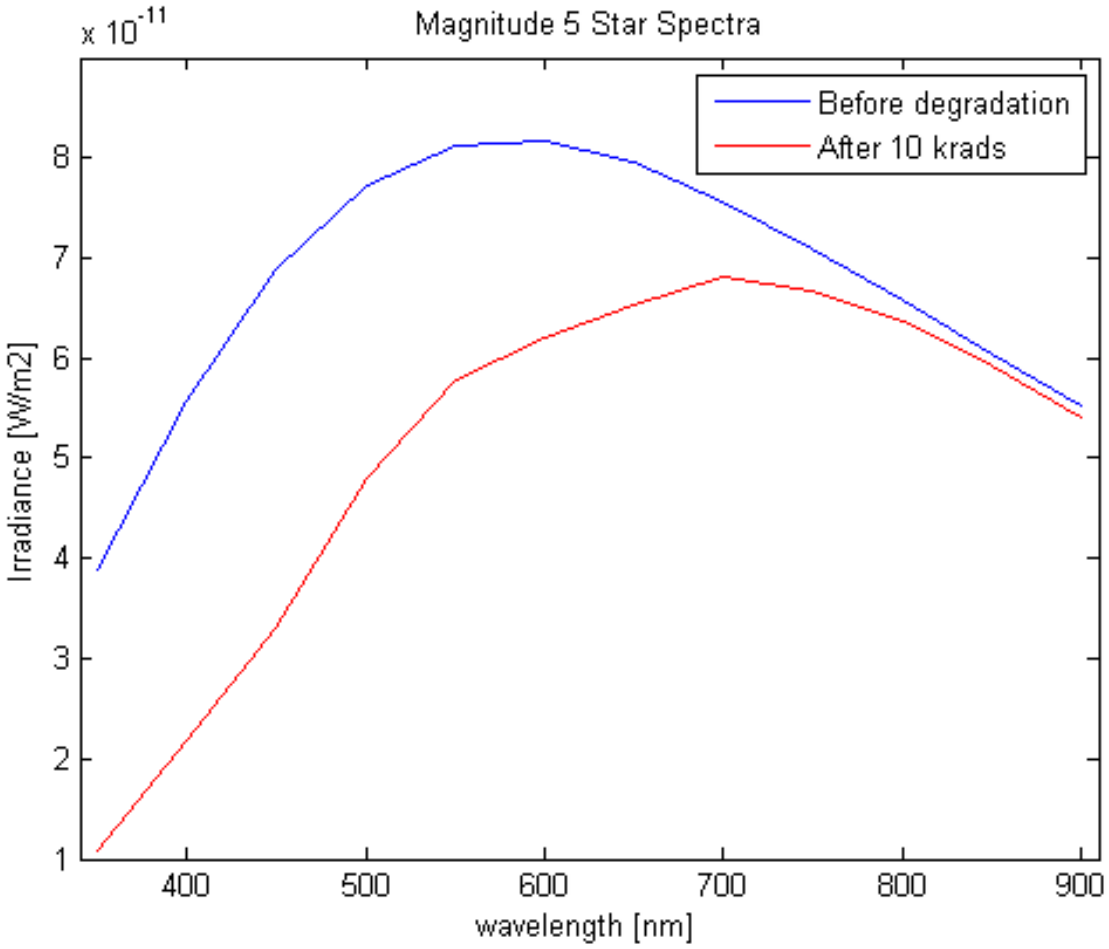

Figure 7.15: Degradation of transmittance after 10krad exposure

7.4 Software Verification: Unit Testing

Unit (module or component) level testing focuses on the early examination of individual functionality and ensures that functionalities not visible at the system level are examined by testing. Unit testing ensures that quality system units are furnished for integration into the finished product.

This document describes the unit tests defined to assess the Facet Nano software functionality and compatibility among modules.

7.4.1 Nomenclature

A name is assigned to each test in the following way:

Test_XXXMM_module.m

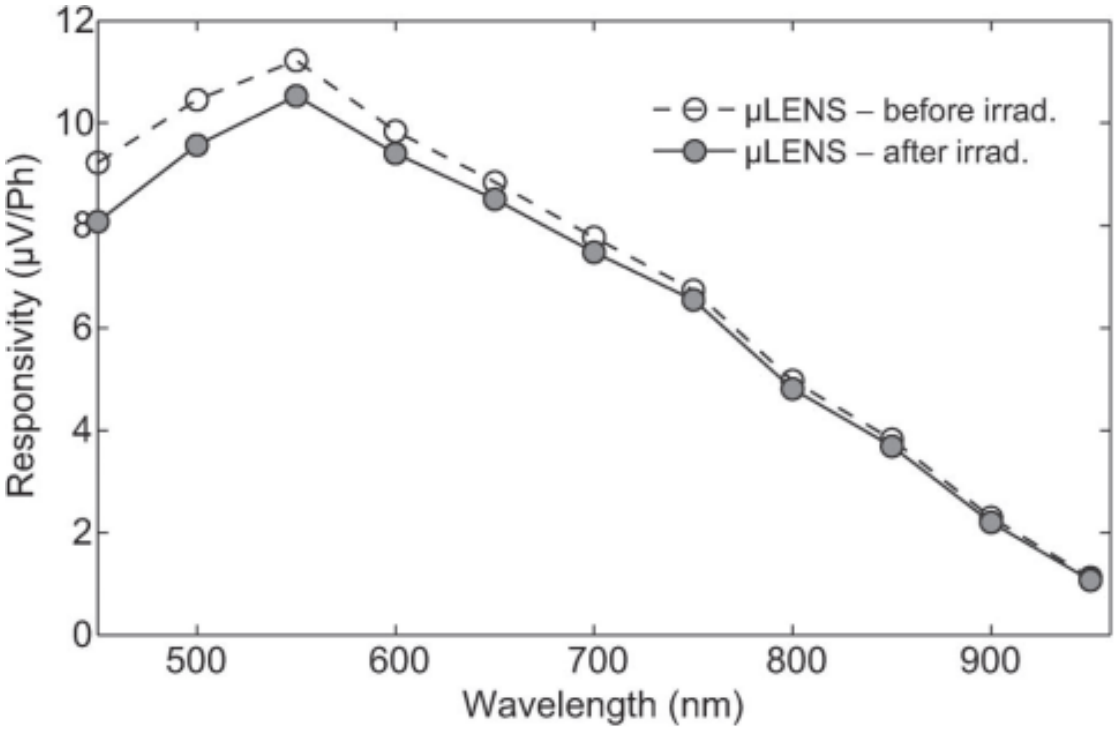

Figure 7.16: Responsivity values before and after TID tests

Where XXX is a number to sort the execution of the tests, MM is the subcomponent where the module belongs:

- IP: Image Processing
- SD: Star Detection
- SC: Star Catalogue
- AD: Attitude Determination

XXX is the number of the test within module MM, and module shall be substituted by the name of the module under test. As an example: *Test_001SD_centroiding.m* represents the first test (001) of the Star Detection subcomponent (SD). The module under test is *centroiding.m*.

7.4.1.1 Run the tests

The tests are located in the folder Tests under the FacetNano project on a subversion repository. In order to run the tests, execute the runtests script from the MATLAB command line. The script is also located under the Tests folder.

7.4.2 Test Description and Results

Table 7.3: Unit-Test Result of SigmaFilterMatlab Module

IP001			**SigmaFilterMatlab.m**
Test Designer:	EF	**Date**:	17/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	N/A	**Performance Parameters**:	SNR
Comments:	Filtered SNR is 1.28 times larger. Input parameters are not checked.		
Requirements:	2.5 2.7 3.7 12.21 13.23 13.24 13.25 13.26		
Status:			**PASS**

Table 7.4: Unit-Test Result of enhanceImageSensorOutput Module

IP002			**enhanceImageSensorOutput.m**
Test Designer:	EF	**Date**:	17/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	N/A	**Performance Parameters**:	
Requirements:	2.5 2.7 3.7 12.21 13.23 13.24 13.25 13.26		
Comments:	Filtered and thresholded SNR is 6.08 times larger. Input parameters are not checked. Threshold should depend on integration time. It is recommended to add integration time as another input parameter.		
Status:			**FAIL**

Table 7.5: Unit-Test Result of centroiding Module

SD001			**centroiding.m**
Test Designer:	EF	**Date**:	
Inputs:	Not checked	**Functionality**:	Fail
Outputs:	Fail	**Performance Parameters**:	Centroiding accuracy
Requirements:	3.12 12.21 13.23 13.24 13.25 13.26		
Comments:	Inputs are not checked. The SigmaFilter is is not able to eliminate all the noise in the image; centroiding algorithm assigns false positives to this noise. It is recommended to add another step between the sigma-filtering/thresholding and the centroiding algorithm, in order to remove noisy-isolated pixels.		
Status:			**FAIL**

Table 7.6: Unit-Test Result of StarID Module

SC100			**StarID.m**
Test Designer:	EF	**Date**:	23/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	Pass	**Performance Parameters**:	# stars in a frame, Attitude accuracy, Performance Star ID
Requirements:	1.3 3.8 6.1 6.12 12.21 13.23 13.24 13.25 13.26		
Comments:	Developed by Delta UTEC. Inputs parameters are not checked. Required inputs do not match the output from the centroiding.m module. It is recommended to rewrite StarID.m as a function instead of a script.		
Status:			**FAIL**

Table 7.7: Unit-Test Result of StarIDConstants Module

SC002			**StarIDConstants.m**
Test Designer:	EF	**Date**:	21/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	N/A	**Performance Parameters**:	-
Requirements:	12.21		
Comments:	This module loads the value of some constants. Developed by Delta UTEC.		
Status:			**PASS**

Table 7.8: Unit-Test Result of FindBrightest Module

SC003			**FindBrightest.m**
Test Designer:	EF	**Date**:	23/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	Pass	**Performance Parameters**:	Performance Star ID
Requirements:	12.21		
Comments:	This module finds the brightest star amongst those detected by the sensor.		
Status:			**PASS**

Table 7.9: Unit-Test Result of FillFOVGrid Module

SC004			**FillFOVGrid.m**
Test Designer:	EF	**Date**:	27/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	COTS	**Performance Parameters**:	# stars in frame
Requirements:	12.21 13.1		
Comments:			
Status:			**PASS**

Table 7.10: Unit-Test Result of FindCandidates Module

SC005			**FindCandidates.m**
Test Designer:	EF	**Date**:	28/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	COTS	**Performance Parameters**:	# stars in frame
Requirements:	12.21		
Comments:			
Status:			**PASS**

Table 7.11: Unit-Test Result of ValidateStar Module

SC006			**ValidateStar.m**
Test Designer:	EF	**Date**:	28/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	COTS	**Performance Parameters**:	# stars in frame
Requirements:	1.3 12.21		
Comments:			
Status:			**PASS**

Table 7.12: Unit-Test Result of FillQuestInputArrays Module

SC007			**FillQuestInputArrays.m**
Test Designer:	EF	**Date**:	28/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	COTS	**Performance Parameters**:	-
Requirements:	1.3 12.21		
Comments:			
Status:			**PASS**

Table 7.13: Unit-Test Result of DetermineAttitude Module

AD001			**DetermineAttitude.m**
Test Designer:	EF	**Date**:	29/02/2012
Inputs:	Not checked	**Functionality**:	Pass
Outputs:	COTS	**Performance Parameters**:	Accuracy around 3 axis Reliability of the Accuracy
Requirements:	1.1 2.12 12.21 13.5		
Comments:	Recommended to set attitude values as output parameters, instead of global variables.		
Status:			**PASS**

Conclusions of the unit tests will be discussed in the next section.

7.4.3 Software Test Conclusions

Unit tests were developed for all the software modules comprising the Facet Nano algorithms that will be run by the Facet Nano camera-prototype. The whole code was analyzed and divided in four sub-components:

- Image processing: including the sigma filtering and the noise thresholding.
- Star detection: implemented using the centroiding method.
- Star catalog matching: responsible for assigning a star vector to each star detected by the centroiding algorithm.
- Attitude Determination: computes the spacecraft attitude using the star vectors.

The last two sub-components were developed by a third-party company (Delta UTEC), as well as the star catalog generation software (not included in the embedded software).

During the tests, two major faults were discovered:

- The interface between the image-processing and the star-detection sub-components are missing. The output from the centroiding algorithm needs to be transformed in order to match the required input for the StarID module. As a consequence, the software cannot provide the attitude data from a sky image autonomously.
- The centroiding algorithm detects many false positives which misleads the star identification algorithm. This can be solved by changing the noise threshold dynamically based on the integration time.

In addition, the following points are proposed to increase the code quality:

- Change scripts to functions. Many algorithms are coded as scripts, hindering the traceability of input/output parameters and therefore, obstructing the verification of software requirements 13.23 and 13.24.
- Remove all global variables. Code is generally clearer and easier to maintain when it does not use globals. Besides it avoids non-locality problems: global variabless can potentially be modified from anywhere. A global variable therefore has an unlimited potential for creating mutual dependencies, and adding mutual dependencies increases complexity. Global variables also make it difficult to integrate and to test modules because one cannot readily set up a 'clean' environment between runs.
- Check input parameters: in critical software, such as space software, it is crucial to install a procedure to detect malfunctioning. For this reason, checking and validating input parameters should be introduced in all functions.
- Compute code coverage and other software metrics in order to remove dead code and meet the ESA software quality standards for small projects.
- Addition of more error checks and debugging instructions: error codes to know where the software crashed and why when an incidence is produced in the embedded software.

7.5 Integration Testing

Integration tests are designed to verify system functionality and to check the performance using FN software and sensor simulator described in section 5.5. They include a series of simulations in order to assess the star tracker performance and availability. They are also used to compare single- and multiple-aperture star trackers.

Integration tests make use of all the software modules developed for the Facet Nano star tracker, including the sensor simulator. A flowchart of these modules can be found in figure 5.1.

7.5.1 Assumptions

The mission is Earth Observation (EO), for example an optical payload. It is assumed that the pointing requirement for this mission is to maintain the instrument pointing

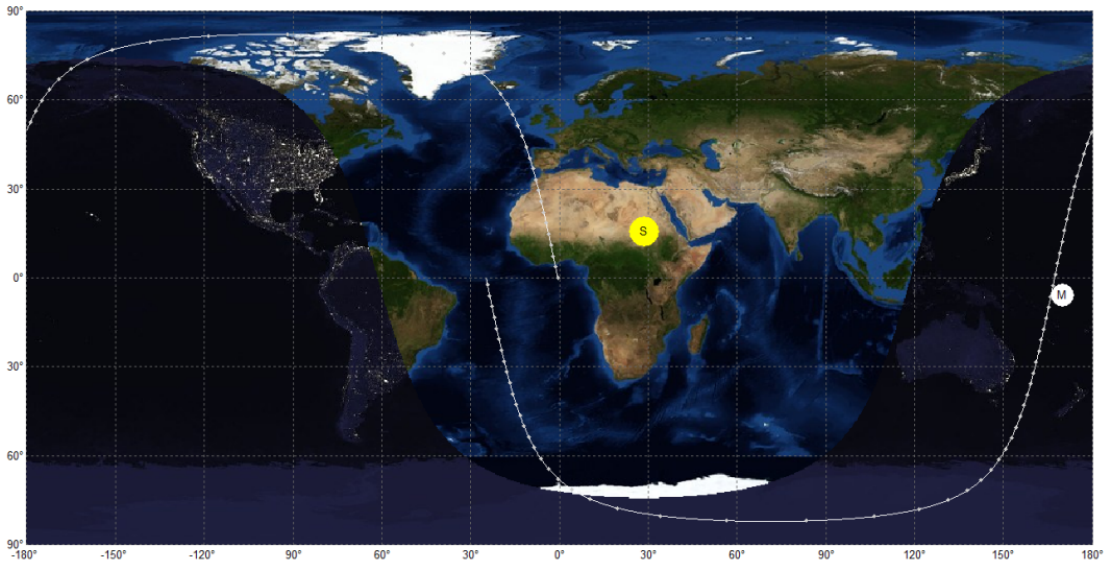

Figure 7.17: Orbit path of the satellite

exactly in the nadir direction. One orbit has been simulated to show the representative performance.

- The Facet Nano star tracker is used in single-aperture configuration, with a field of view (FOV) of 14 degrees and limiting magnitude of 5.0.
- The instrument is a 1 mega-pixel imager with 10m ground resolution, for a FOV of around 1 degree and pixel angle of 3.6 arcseconds. This drives a pointing requirement of 180 arc-seconds (5% of the FOV for tightly over-lapping images), and a stability of 1 pixel per maximum exposure or 3.6 arc-seconds in, say, 10 ms.
- There are no pointing restrictions in roll around the Nadir-pointing axis, meaning that this axis is used to point the star tracker away from the sun. The instrument is aligned with the Nadir pointing axis.
- Orbit is Sun-synchronous, 10am ascending node, 600km altitude.
- The moon is modeled in the correct orbit, and assumed to be close to opposition to the sun but not in eclipse. This is intended to be representative of a real situation, but no attempt was made to determine the worst case conditions for the moon.

The orbit path of the satellite is shown by the gray line in Figure 7.17, indicating the sub-satellite point on the surface of the earth. The dots indicate the position every minute. The sun position is indicated by the yellow circle, and the moon by the white circle, which do not change position appreciably during the satellite orbit.

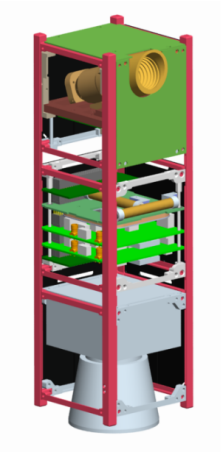

Figure 7.18: Satellite configuration

The satellite is assumed to be a three-unit CubeSat, with the long axis pointed directly to nadir. The design includes a single star tracker tilted at 20 degrees away from the instrument axis (ie. away from nadir). The tilt angle was chosen to provide a reasonably compact design with an increased angle between boresight and the earths limb. Mounted at this angle, the star tracker takes up an effective volume of approximately 100 x 69 x 50 mm3, or approximately one quarter of a CubeSat Unit as shown in Figure 3. For a 3U CubeSat this leaves plenty of space for bus systems and payload. The mass is estimated to be 92 grams, with an average power consumption of ¡ 0.15 W.

The star tracker field of view (FOV) is circular as this is the projection of most star tracker optics onto the detector. The path of the FOV across the sky is shown in Figure 7.19. The Milky Way is visible as the region with the higher density of bright stars that cuts across the sky. It can be seen that the FOV traverses areas of both high and low star density.

7.5.2 Sky Coverage

Sky coverage is the fraction of all possible pointing directions for which the star tracker can provide 3-axis attitude information, ignoring blinding from the Earth, Moon and Sun.

Using a limiting magnitude of 5.0 and a circular FOV of 14 degrees diameter, the numbers of detected stars during the orbit are shown in Figure 7.20 for images taken once per second during the entire orbit.

The visual magnitude of the stars was used to generate these figures; however the detector in the star tracker has a different response spectrum, requiring conversion to

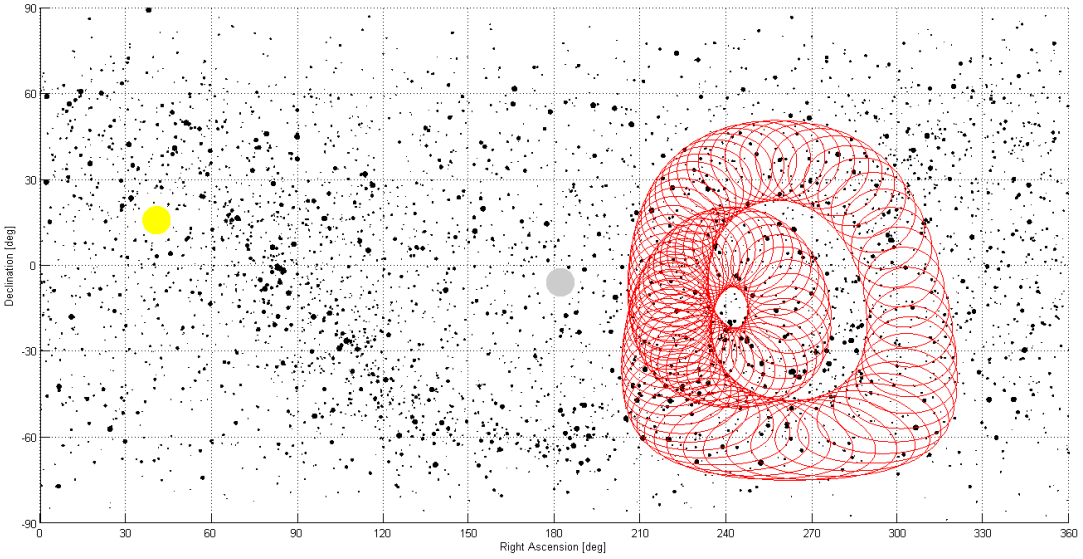

Figure 7.19: Path of the FOV (red circles) across the star field. The sun is shown as a yellow circle and the moon as a gray circle

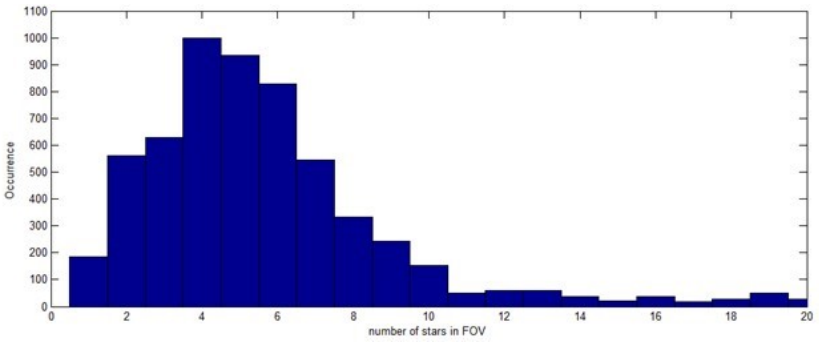

Figure 7.20: Histogram of the Number of stars visible in the FOV over one orbit

instrumental magnitude. Most detectors have a stronger response in the red region than the V-band filter, and luckily most stars are also redder than the reference zero B-V index. When converting to instrumental magnitude for the Facet Nano detector, as shown in Figure 7.21, the stars shift towards a lower magnitude (note that the original catalog contained only stars brighter than magnitude 6). Therefore they will appear brighter, meaning we have under-estimated the number of stars by about a factor of 2.

Attitude determination takes place in three modes:

- In 87.1% of images there are at least three visible stars, allowing a unique fit to the star catalog

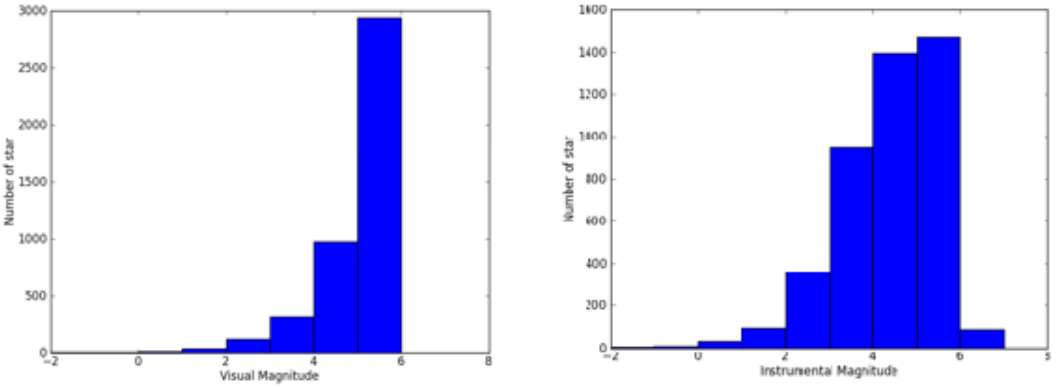

Figure 7.21: Histograms of Visual Magnitude and Instrumental Magnitude for all stars in the catalogue

- In 6.9% of images there are two visible stars of sufficiently different magnitude, also allowing a unique fit to the star catalog. The required magnitude difference is assumed to be 0.25 magnitudes.
- In 2.7% of images there are two visible stars of similar magnitudes, allowing a unique fit only if the angular separation is unique in the catalog (within measurement errors), and there is prior attitude knowledge that can resolve the unknown symmetry.
- In 3.2% of images there is 1 star visible and the single star tracker cannot provide 3-axis attitude information

7.5.3 Performance

7.5.3.1 Performance Parameters

There is usually some confusion about how to correctly specify attitude sensor performance. The definitions in Table 7.14 may be helpful. The same parameters can be stated in different units and different distributions or confidence levels, so it is often quite difficult to compare the performance of two different star trackers.

7.5.3.2 Accuracy

Star centroiding accuracy, and subsequent attitude determination accuracy can be affected by the following things:

Table 7.14: Star tracker performance parameters

Parameter	Description	Units	Distribution
Accuracy	for a large number of different attitudes, this is the mean of the absolute value of the difference between the mean attitude estimate and the actual attitude (the center of the black circle in Figure 16, where it is 0.5 both horizontal and vertical)	degrees, arcminutes, arcseconds, miliradians	Typically quoted at the 1-sigma or 3-sigma level
Sky Coverage	Ignoring blinding from the Earth, Sun and Moon (ESM), the star tracker can return an attitude result (i.e. lower than a threshold error limit) for some % of all pointing angles. Can be type 1, 2 or 3 tracking, as described in the discussion of Sky Coverage.	%	Confidence interval
Availability	Taking into account blinding from the ESM, the star tracker can return an attitude result (i.e. lower than a threshold error limit) for some % of expected pointing angles and orbit positions.	%	Confidence interval

- Star catalog errors
- Alignment of the optics during assembly
- Alignment of the star tracker relative to the instrument
- Thermal expansion
- Stray light causing a constant offset type error
- Software behaviour

It is beyond the scope of this project to discuss catalog errors, assembly issues, alignment, thermal design or unexpected algorithmic artifacts. It is also very difficult to model accurately, and is usually best calibrated on orbit.

7.5.4 Results

The single-aperture Facet Nano star tracker was designed to have 36 arcsecond accuracy (1-sigma) in horizontal and vertical directions, the theoretical results of which are shown above. A full system Monte Carlo simulation of the example orbit using first generation star tracker algorithms, demonstrates a centroiding accuracy of 0.41 pixels, as shown in Figure 7.22. This results in a cross-boresight accuracy of 17.5 arcseconds and roll

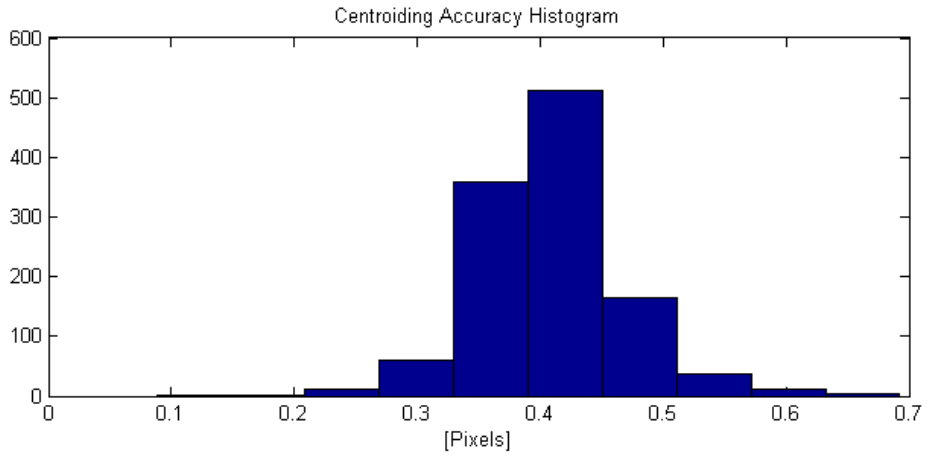

Figure 7.22: Histogram of calculated star centroid accuracy

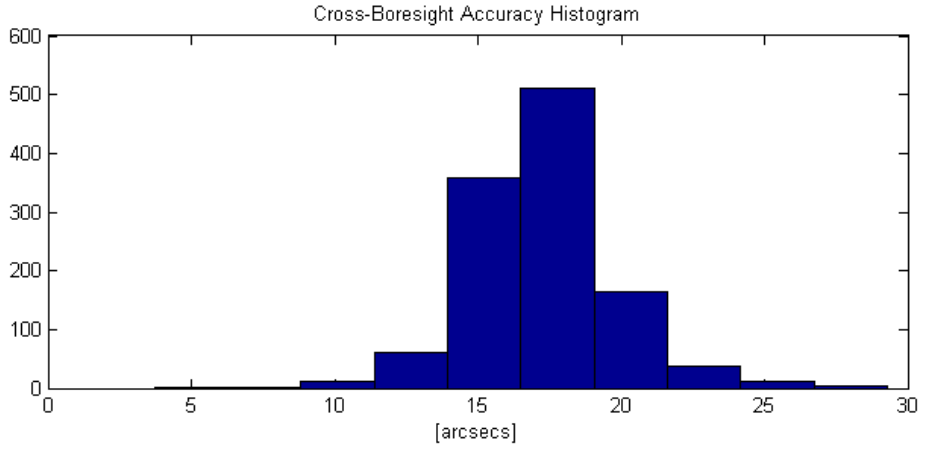

Figure 7.23: Histogram of Facet Nano star tracker cross-boresight accuracy

accuracy of 180 arcseconds, as show in Figure 7.23 and Figure 7.24, exceeding the design goal by a considerable margin.

The Facet Nano star identification algorithm detected 99.5% of stars down to Vmag=6.0, with 6% false positives. Availability of attitude determination in mode 1, as discussed previously, was calculated to be 86.7%, compared to the theoretical maximum equal to the sky coverage of 87.1%.

- Sky coverage: 86.7%, confidence interval 98.5%
- Availability: 86.7%, confidence interval 98.5%
- Accuracy: 17.5 arcseconds, 1-sigma (using first generation algorithms)

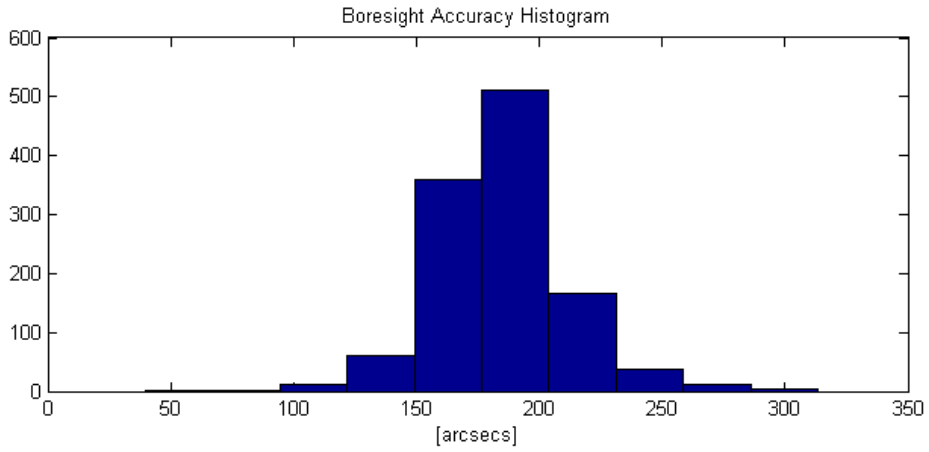

Figure 7.24: Histogram of Facet Nano star tracker roll accuracy

7.5.5 Multi-Aperture Comparison

It has been shown that a single aperture star tracker is suitable for:

- Nadir pointing missions that are free to rotate around the earth vector, with accuracy requirements similar to those presented in this chapter and orbit configurations that avoid blinding
- Sun-pointing missions that are free to rotate around the sun vector, with accuracy requirements similar to those calculated in this chapter and orbit configurations that avoid blinding

However, multi-aperture star tracker systems are preferred for:

- Earth or sun pointing mission with large slant angles
- Earth or sun pointing mission with more onerous accuracy requirements
- Dynamic pointing missions (e.g. dynamic pointing, scanning and tracking, inertial pointing, astronomy)

Roll angle accuracy is increased dramatically with multi-aperture systems. With two apertures, the first aperture has high precision in X and Z axes, the second aperture at 90 degrees to the first will have high precision in Y and Z axes, resulting in high precision in all (X, Y and Z) axes.

7.6 End-to-End Test

The main purpose of this activity is to test all the modules, components and interfaces of the current Facet Nano hardware and software versions. The objective is to validate the system functionality by obtaining attitude measurements capturing real night-sky images.

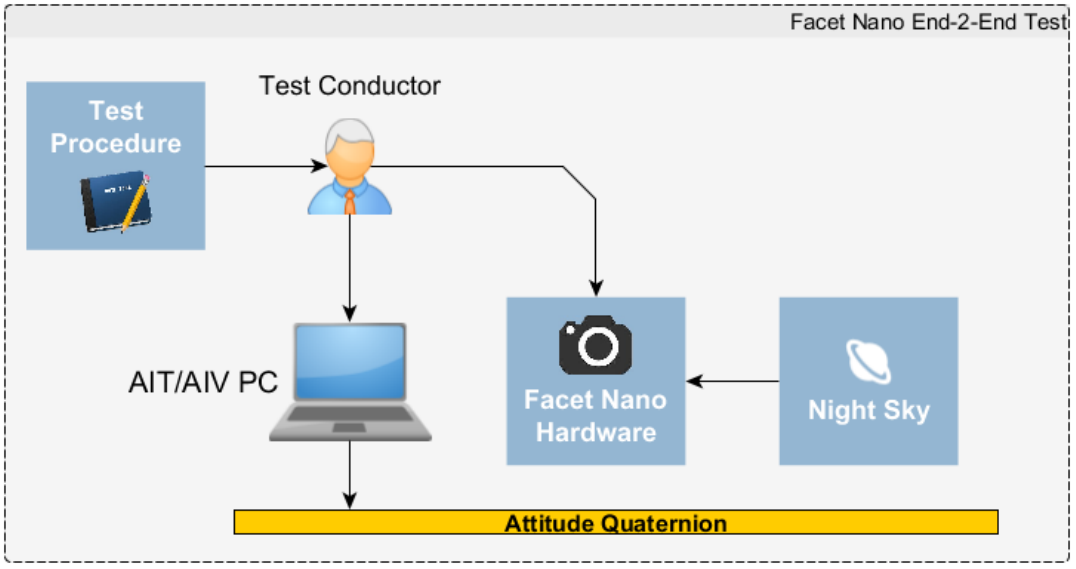

Figure 7.25: Facet Nano End-to-End Test Set-up

Correctness and accuracy of the attitude measurement is not assessed in this test for complexity reasons. Precision of the attitude determination measurement is evaluated using the electro-optical simulator in the integration tests, section 7.5.

7.6.1 E2E Test Set-up

The setup of this test is shown in figure 7.25. A test specification is written, including the procedure to be followed by the test conductor. In a nutshell, it includes:

- Selection of the place where the test will take place: the objective is to acquire as many star images as possible. Therefore a place with low light pollution is required. We can see in figure 7.26 that the Netherlands is not the best place for this kind of night tests. However, a place in The Hague near the coast is selected due to budget constrains. It shall be noted that the effect of the atmosphere reduces the amount of starlight that reaches the earth surface. Therefore, the minimum star magnitude that is detectable by Facet Nano is greater in space than on the ground.

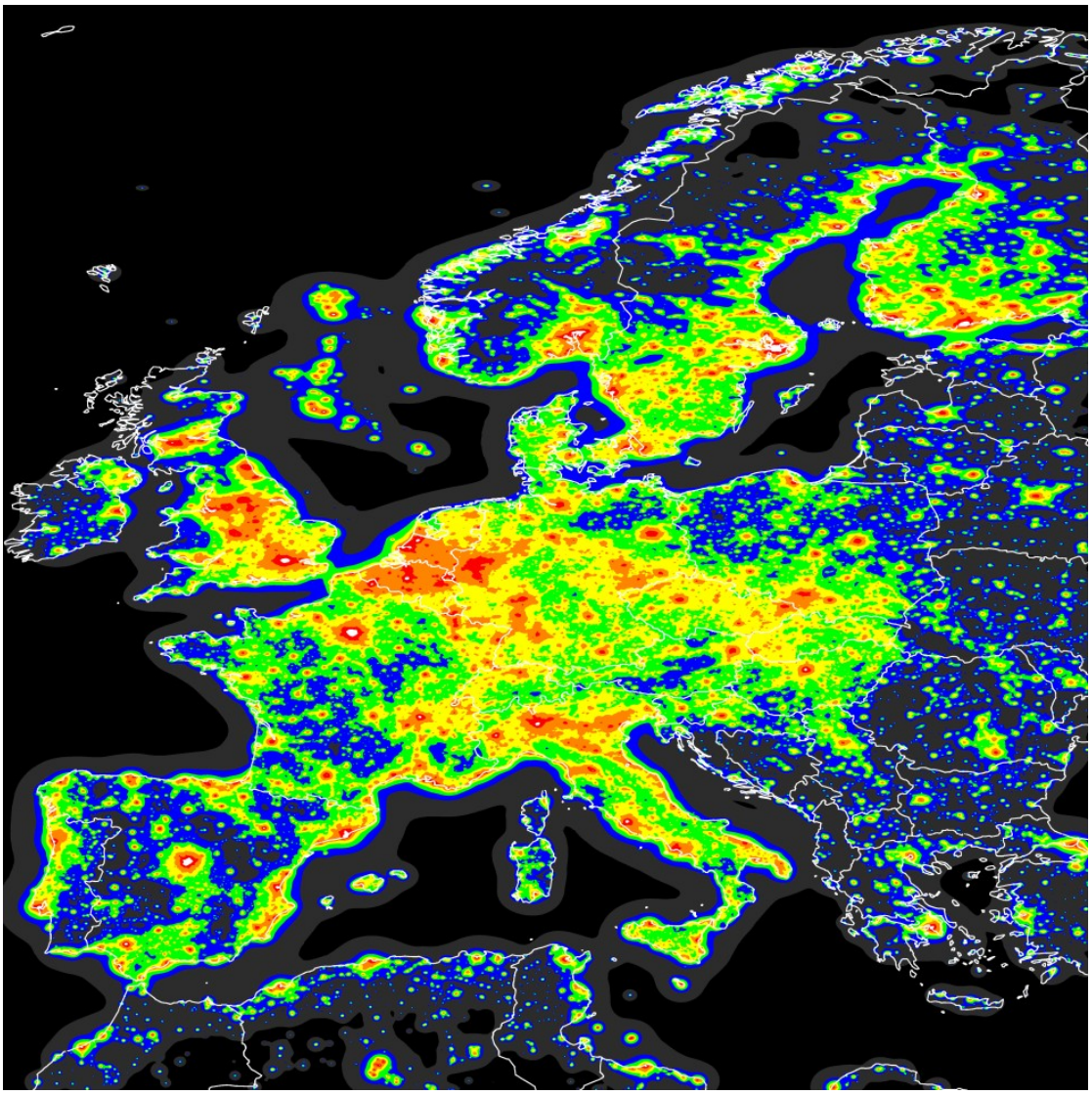

Figure 7.26: Light Pollution in Europe

- Acquisition of the images: Facet Nano Camera shall be pointing towards the sky, preferably towards the brightest visible group of stars. This will increase the likelihood to detect at least two or three stars, the minimum amount of centroid needed by the attitude determination algorithm. The camera shall remain steady on the ground to avoid capturing blurry images.

- Processing of the images: a test computer is carried to the designed place. Facet Nano is connected to the computer via USB connector, enabling imaging transmission to the PC hard drive. Once the image is on the PC, the data is ingested in the processing chain (see figure 5.1). The result of the processing will show an attitude quaternion as long as there is a match between the detected stars on the field of view of the star tracker and the star catalog.

Figure 7.27: E2E Field Test Sky Image

7.6.2 E2E Test Results

A sample image captured during the end-to-end field test is shown in figure 7.27. With a naked eye, we could dare to say that there is one single star detected on the field of view.

Next step is to process the image and extract the centroid position of the star:

Table 7.15: Star centroids detected in E2E field test sample image

#	Centroid X [pixels]	Centroid Y [pixels]
1	-423	187
2	-469	156
3	-22	47

As we can see in table 7.15, our first visual check was wrong and there are actually three stars in the field of view detected by the Facet Nano software. Let's do some tricky offline processing on the image capture to see whether we can visually see the stars that the Facet Nano software has detected. Figure 7.28 shows the colors inverted, so bright spots will appear in black.

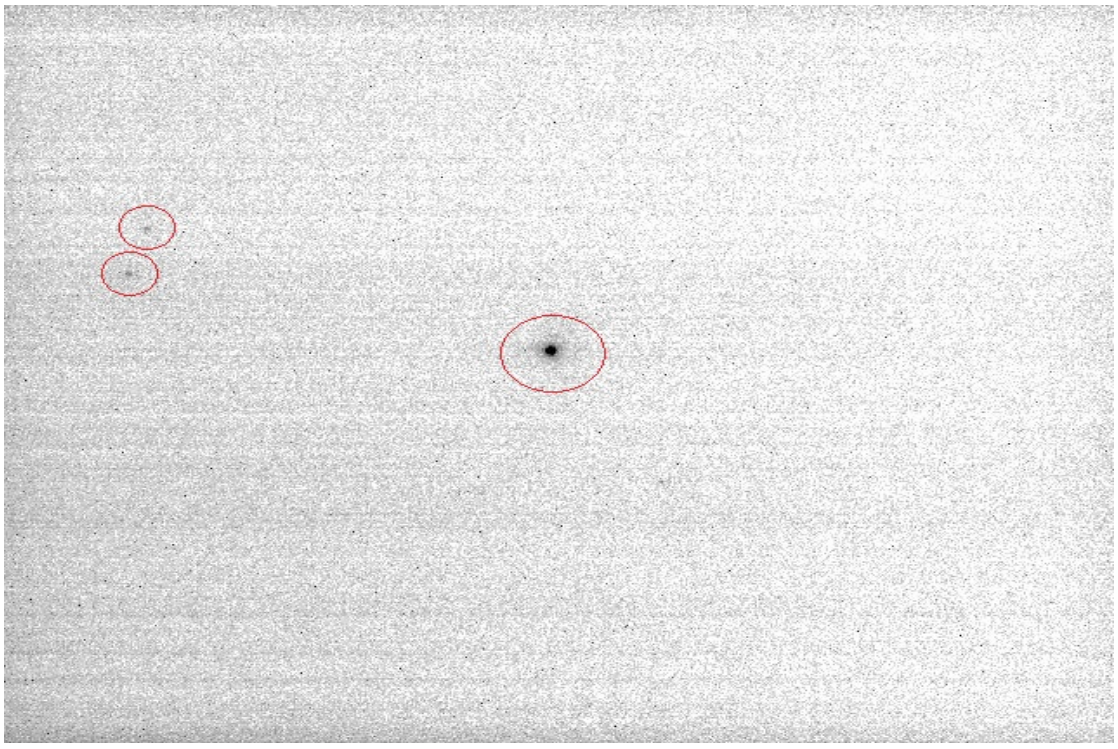

Figure 7.28: E2E Field Test Sky Image - Inverted Colors

Indeed we can see three black spots, that is, three stars in the field of view: the brightest one in the middle of the image, and two very close in the top-left quadrant of the image.

Lastly, the centroid positions are ingested into the attitude determination algorithm. The software is able to output a quaternion estimation, meaning there is match between the centroids and the stars in the catalog. As referred before, the accuracy of the estimated attitude will not be addressed here, as it will imply a more precise and complex set-up of the test, which is out of the scope of this phase of the project.

7.7 Summary

In this chapter the verification of the Facet Nano system is approached. Firstly, the verification plan and the verification methods are presented: inspection, analysis demonstration and test. Then, the different tests undertaken in this project are described.

Starting with the baffle performance test, the sun exclusion angle was measured. Conclusion is that an improved baffle is required in order to block sunlight coming from angles closer to the optical axis. This will improve the minimum star magnitude detectable by the system, therefore, increasing the number of detectable stars and enhancing the star tracker availability and accuracy.

Next we presented the radiation test performed at ESTEC Co-60 facility. Analysis of the radiation dose that the system would undergo in orbit were computed using SPENVIS software. Results of the analysis were used as an input to the radiation test specification to define the maximum TID dose and the aluminum thickness required by the instrument to survive.

In addition, software verification was undertaken from a bottom-up approach. First each of the software modules was verified by the unit testing method. Once the functionality of each module was verified, they were integrated and tested against the electro-optical simulator developed for this purpose. In the integration tests, we run a Monte-Carlo simulation and we extracted some figures for the system performance, concluding that system accuracy is exceeding the design goal.

In the end, an E2E field test was undertaken, facing Facet Nano hardware towards the Dutch night sky. The system was able to detect several stars and produce an attitude estimation in spite of the clouds and light pollution.

Chapter 8

Conclusions

One of the most limiting factors of applications for nanosatellites is their relatively poor attitude determination and control capability: poor both in terms of accuracy and rate of success. Within the Facet Nano project, we aimed at developing a star tracker for small satellites capable of overcoming such limitations, from a systematic perspective.

The starting point of this project was a series of system requirements. The requirements lead to a functional analysis of the instrument. From the functional diagram, we could clearly see the interdependence of the different blocks and they were directly captured into a N2 chart, including functional interfaces as well.

An initial analysis was perform up to a performance budget, in order to assess the feasibility of the project and to estimated the required performance for the image sensor. Besides, the verification process was present from the very beginning, including a verification plan and other activities described in chapter 7.

From the estimated performance of the sensor, we could pre-select several image sensors based on the data-sheet values claimed by manufacturers. A bunch of CMOS and CCD sensors were evaluated. A trade-off was performed in order to select a 1Mpix CMOS sensor as the imaging detector for the star tracker. Trade-off criteria included spectral and radiometric performance, operating temperature range and power consumption.

With the optics and the sensor already selected for the instrument, a first iteration of the baffle design was undertaken. The design turned to be simple but effective against low exclusion angles. It might be used for the multi-aperture configuration of the star tracker, as the requirement for sun-exlusion angle are more laxative due to the increased availability of the multi-aperture star-tracker.

A radiation test was designed for the selected image sensor. It was carried out at ESTEC ESA facilities and it demonstrated not only that the COTS sensor would survive a LEO

mission of 3 years, but it would also maintain its radiometric characteristics within the system requirements.

Furthermore, a detailed description of the calibration activities performed on ground was included in chapter 6. In addition, a calibration and validation plan was established for the early operations and commissioning phase of the mission.

Figure 8.1: Facet Nano development model, without baffle

With respect to software, several algorithm were developed and tested: from raw data processing, star detection, centroiding estimation and electro-optics simulator. The simulator was validated by comparing its output against star images produced in the laboratory and was used to test and debug the Facet Nano functionality. A Monte-Carlo simulation was implemented to estimate some system parameter figures, such as such as accuracy, sky coverage and instrument availability. We could conclude that a star tracker with multi-aperture would enable better attitude accuracy and higher availability.

Finally, an end-to-end activity was undertaken in order to put all the pieces of the system together and test it out of the laboratory: engineer, optics, electronics and software in a night-sky field test. The result was promising as the Facet Nano development model was able to detect and identify real stars, even under adverse unfavorable conditions: after receiving 5krad TID dose, a light-polluted environment and the attenuation of the atmosphere.

8.1 Future Work

Facet Nano star tracker has demonstrated its feasibility and is close to achieve a full engineering model. Following there is a series of recommended lines of research to continue the Facet Nano development:

- Introduce a geometric model to correct distortions and enhance centroiding accuracy
- Fine-tune the Gaussian shape used in the correlation technique to make it more similar to a star PSF. This way, false positive likelihood would decrease.
- Work on the selection of COTS electronics for the read-out board of the sensor.
- Once read-out electronic is chosen, Facet Nano software can be ported to a language that runs on that board.
- Create a new tracking algorithm that works after the Lost-In-Space operation mode finishes successfully. This would reduce processing time and therefore, power consumption.
- Make a more detailed analysis of radiation on the SAA zone. SEU events are statistically more intense in this zone and could affect the ACS performance. Tracking algorithm would mitigate radiation effects in this area.
- Consider the alignment of the star tracker with respect to the platform reference frame. Relative measurements of both reference frame shall be taken in order to transform attitude quaternions from the star tracker reference to the spacecraft main reference frame.
- Further testing of the system under launch and space environment, including vibration tests and thermal-vacuum cycling.

Appendix A

Gantt Chart

A Gantt chart is a graphical representation of the duration of tasks against the progression of time. It is a useful tool for planning and scheduling projects. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project.

During this project, the Gantt chart was utilized in order to:

- Plan and schedule: to plan how long the Facet Nano project should take and the order in which the tasks need to be carried out.
- Monitor the project: A Gantt chart lets you see immediately what should have been achieved at any point in time. It is very useful for representing deadlines and other significant events known as milestones.

A sample of the Gantt chart used during the Facet Nano development is shown in Figure A.1.

Facet Nano
ISIS

Today's Date: 24/02/2009 Tuesday
(vertical red line)

Thesis Lead: E. Fraile
Start Date: 18/11/2011 Friday

First Day of Week (Mon=2): 2

WBS	Tasks	Task Lead	Start	End	Duration (Days)	% Complete	Working Days	Days Complete	Days Remaining
1	**Simulation I**	Congying	18-09-11	16-12-11	90	100%	65	90	0
1.1	Sensor model update I	EF	18-09-11	16-12-11	90	100%	65	90	0
2	**Requirements**	EF	09-12-11	26-01-12	49	100%	35	49	0
2.1	Electrical reqs.	EF	09-12-11	22-12-11	14	100%	10	14	0
2.2	Software reqs.	EF	06-01-12	19-01-12	14	100%	10	14	0
2.3	Verification reqs.	EF	13-01-12	26-01-12	14	100%	10	14	0
3	**Electrical**	EF	20-01-12	13-03-12	54	100%	38	54	0
3.1	Sensor Selection	EF	20-01-12	18-02-12	30	100%	21	30	0
3.2	Electro-optical Characterization	EF	13-02-12	13-03-12	30	100%	22	30	0
4	**Software**	EF	01-03-12	04-05-12	65	93%	47	60	5
4.1	Design	EF	01-03-12	10-03-12	10	100%	7	10	0
4.2	Image Processing Algorithm	EF	11-03-12	30-03-12	20	100%	15	20	0
4.3	Star Detection Algorithm	EF	01-04-12	20-04-12	20	100%	15	20	0
4.4	Optimization	EF	15-04-12	04-05-12	20	80%	15	16	4
4.5	Unit Testing	EF	01-05-12	30-05-12	30	50%	22	15	15
5	**Environmental testing**	AH EF	01-04-12	03-06-12	64	59%	45	37	27
5.1	Test design	AH EF	01-04-12	10-04-12	10	95%	7	9	1
5.2	Thermal Test	AH EF	11-04-12	24-04-12	14	80%	10	11	3
5.3	Radiation Test	AH EF	14-05-12	27-05-12	14	60%	10	8	6
5.4	Single Event Effects	AH EF	21-05-12	03-06-12	14	10%	10	1	13
6	**Simulation II**	Congying	23-04-12	30-06-12	69	62%	50	42	27
6.1	Sensor model update II	EF	23-04-12	06-05-12	14	100%	10	14	0
6.2	Monte-Carlo Simulation	CH EF	01-06-12	30-06-12	30	10%	21	3	27
7	**Opto-Mechanical**	AH	03-05-12	05-06-12	34	100%	24	34	0
7.1	Phase-A Mechanical Design	RS EF	03-05-12	09-05-12	7	100%	5	7	0
7.2	Production	RS	10-05-12	16-05-12	7	100%	5	7	0
7.3	Integration	RS EF	17-05-12	23-05-12	7	100%	5	7	0
7.4	Optical Characterization	EF	23-05-12	05-06-12	14	100%	10	14	0
8	**Verification & Validation**	EF AH	22-05-12	16-06-12	26	50%	19	13	13
8.1	Software	EF	22-05-12	30-06-12	40	50%	29	20	20
8.2	Sensor Performance	EF	22-05-12	10-06-12	20	50%	14	10	10
8.3	End-to-end Test	AH EF	25-05-12	16-06-12	23	50%	16	11	12

Figure A.1: Gantt Chart

Appendix B

MATLAB Code

The present appendix contains the source code of the most relevant software routines wirtten in MATLAB.

CameraParameters.m

```
%*************************************************************************
% This script contains all the different parameters of the Image sensor.

%*************************************************************************
%  Physic constants
%*************************************************************************

KB          = 1.38066e-23; % [J/K] Blotzmann constant
q_electron  = 1.60218e-19; % [C] elementary charge of the electron
LigthSpeed  = 2.99792e8;   % [m/s] ligth speed in vacuum
h           = 6.62607e-34; % [J s] Planck constant

%*************************************************************************
%  Image sensor Parameters
%*************************************************************************
CMOSsizeX = 844;        % [pixel] image sensor dimension
CMOSsizeY = 640;        % [pixel]

Pixel_size  = 5.8e-6;   % [m] image sensor pixel size
sensorArea = Pixel_size^2; %[m2]

%% %%%%%%% This value needs to be adapted!!! Study more in depth
Sensitivity = 68e6;       %[LSB*m2/J] a linear response is supposed
%%
```

```
ADC_nb_bit   = 8;       % Number of bit of the image sensor output (ADC)
Vref_ADC     = 1;        % [V] reference Voltage for the image sensro ADC

% Noise specific parameters
% G           = 34e-6;    % [V/e-] Conversion gain
% Capacitance = q_electron/G;   % [F] pixel capacitance
% avgQEFF     = 25;%38;        % [%] Average Quantum efficiency * fill factor
                          % max 45% for LUPA300
%avgLambda   = 550e-9;    % [m] Average wavelength of the light

%FPN         = 2.5;       % [% RMS] Fixed pattern noise: NOT CONSISDERED!!!

PRNU_RMS     = 3.1;%2.5;       % [% RMS] Photo response non uniformity / 1 sigma va
% V_dark_rate_avg = 0.3;       % [V/s] Dark current voltage rate @Tref_idark
% DSNU_max     = (0.40-V_dark_rate_avg)/V_dark_rate_avg*100; % [%] maximum dark cur
               %0.35
% Tref_idark  = 273.15+21; % [K] reference temperature for dark current
% Ea_avg      = 0.80;%0.75;      % [eV] Activation Eneergy for the dark current
%                          % temperature dependance.
%                          % 0.7eV corresponds to a doubling of idark every
%                          % 8 C , 0.75eV every 7.5 C.
%                          % (approx valid only around -40 C  to +60 C)
%                          % Ea is usually comprised between 0.5Egap and 1Egap
%                          % Egap = 1.16eV for Silicon


%************************************************************************
%  Optics Parameters
%************************************************************************
PSF_pitch = 1/80;%1/40;       % [pixel] Pitch used for computing the point spread
                         % function

% [W] total power coming from the star received on the image sensor,
% smagn = apparent star magnitude.
% TotalStarPowerFunction = '2.4579e-12*exp(-0.92069*smagn)';
TotalStarPowerFunction = '6.823e-13*exp(-0.82057*smagn)';%'2.354015E-12*exp(-0.916

% Amount of energy received in a square; usefull since the exact PSF is not
% really known and has an "infinite" length.
%EnergyRatioInSquare = 0.62;
%EnergySquareSize = 5;    % [pixel] size of the side of the square where the
                         % 'EnergyRatioInSquare' is considered.

%Background_intensity = 0;% 9.92e-13; % [W/m2] space backgound light intensity at
```

```
%Background_NU = 10; % [%] background ligth non uniformity
```

centroiding.m

```
function [star_xy_irf,est_magn, nb_star] = centroiding(img,window_size,numiterCent
%*************************************************************************
%
% Function description:
% Centroiding algorithm. Detect stars and Extract the position of the star
% according to the image generated by the image sensor.
%
% Inputs:
%  - img            : image from image sensor in image reference frame:
%                     (the center of the 1st pixel is at (1,1))
%                     (1,1)-----> x,(1<=x<=camSize(1))
%                        |
%                        |
%                        V  y,     (1<=y<=camSize(2))
%  - window_size    : Size of the window used for centroiding [pixel]
%  - numiterCentroiding: Total number of centroiding iterations when a star
%                     is detected in order to increase precision
%  - makeplot
%
% Outputs:
%  - star_xy_irf    : Star position in IRF, nb_star x 2 matrix
%  - est_magn       : Estimated star magnitude. This value still needs
%                     scaling + offset to match with the star magnitude.
%                     nb_star x 1 matrix
%  - nb_star:       : Number of star detected
%
%*************************************************************************


%*************************************************************************
% Parameters
%*************************************************************************
CameraParameters;

%*************************************************************************
% Process image to detect stars
%*************************************************************************
[sy sx] = size(img);     % image size

nb_star = 0; %Initialize number of star detected
star_xy_irf = []; % No star
```

```
est_magn = [];

% Move the window on the complete image
for x = window_size/2+1:window_size:sx+window_size/2
    for y = window_size/2+1:window_size:sy+window_size/2
        [xy, est_magnitude] = moment_1st_order(img,[x y],window_size);

        if ~isempty(xy)   % if some pixels are different than 0 in the window
            nb_star = nb_star +1;    % star detected
            for i = 1:numiterCentroiding-1
                [xy,est_magnitude] = moment_1st_order(img,xy,window_size); % do re
            end

            star_xy_irf(nb_star,:) = xy;    % write/store star position
            est_magn(nb_star,1) = est_magnitude;

            % Erase star in the image in order not to detect it a second
            % time: the star is supposed not to be bigger than window_size,
            % the erased part is rectangular.
            x_index = round(xy(1)-window_size/2):1:round(xy(1)+window_size/2);
            y_index = round(xy(2)-window_size/2):1:round(xy(2)+window_size/2);
            x_index = x_index(x_index > 0 & x_index <= sx);   % limit out of image
            y_index = y_index(y_index > 0 & y_index <= sy);

            img(y_index,x_index) = 0; % Erase

            if makeplot
                figure
                [x_,y_] = meshgrid(1:sx,1:sy); % create grid for interpolating poi
                surf(y_,x_,img);   % Invert x y axes for imaging IRF conventions
                colorbar; axis square;
                xlabel('y [pixel]'); ylabel('x [pixel]');
                s = ['Star ',num2str(nb_star),' removed'];
                title(s); set(gcf,'Name',s);
            end
        end
    end
end

%*************************************************************************
% Internal Functions
%*************************************************************************

function [xy,est_magn] = moment_1st_order(img,pos_xy,window_size)
```

```
% Compute first moment = center of gravity
%
% Inputs:
% - img         : image
% - pos_xy      : position of the window [pixel] (pixel value = 0 for out of
%                 image position)
% - window_size: size of the squared window: size x size [pixel]
%
[sy sx] = size(img);

s1 = 0;
s2 = 0;
s3 = 0;
for x = round(pos_xy(1)-window_size/2):1:round(pos_xy(1)+window_size/2)
    for y = round(pos_xy(2)-window_size/2):1:round(pos_xy(2)+window_size/2)

        % Handles out of image values by replacing them with 0
        if x<1 || y<1 || x > sx || y > sy
            pixel_value = 0;
        else
            pixel_value = img(y,x);

        end

        if pixel_value  % for improving speed
            s1 = s1 + x*pixel_value;
            s2 = s2 + y*pixel_value;
            s3 = s3 + pixel_value;
        end
    end
end

if s3  % there was a star in the window
    xm = s1/s3;
    ym = s2/s3;
    xy = [xm ym];
    est_magn = -log(s3);
else
    xy = [];
    est_magn = [];
end
```

enhanceImageSensorOutput.m

```
%function imgOut = enhanceImageSensorOutput(imgIn, gaussianWindowSize, noiseThresh
function imgOut = enhanceImageSensorOutput(imgIn, noiseThreshold, sigmaWindowSize,
%**************************************************************************
%
% Function description:
% Enhance the image from image sensor in order to apply centroiding.
%
% Inputs:
%   - imgIn         : image from image sensor
%
% Outputs:
%   - imgOut        : filtered / enhanced image
%**************************************************************************

%**************************************************************************
% Bad pixel detection and correction
%**************************************************************************
% The Sigma filter acts as a selective(exclusive towards noise) averaging
% filter. Therefore, it also acts as a kind of low pass filter.
figure; imagesc(imgIn);
img_filtered = SigmaFilter(imgIn, sigmaWindowSize, sigmaThreshold, minPixFraction)
figure; imagesc(img_filtered);
%**************************************************************************
% Low pass Filter
%**************************************************************************
%img_filtered = filterImage(img_filtered, gaussianWindowSize, makeplot);

%**************************************************************************
% Threshold
%**************************************************************************
%imgOut              = img_filtered-noiseThreshold;   % substract threshold
%imgOut(imgOut<=0)   = 0;
img_filtered(img_filtered<=noiseThreshold) = 0; % clip below threshold
imgOut = img_filtered;
figure; imagesc(imgOut);
%**************************************************************************
% Plot
%**************************************************************************

if makeplot
```

```
    [sy sx] = size(imgOut);
    figure
    [x,y] = meshgrid(1:sx,1:sy); % create grid
    surf(y,x,imgOut,'EdgeColor','None')   % Invert x y axes for imaging IRF conver
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]');zlabel('[LSB]');
    s = 'Thresholded image (3D)';
    title(s); set(gcf,'Name',s);

    figure;
    imagesc(imgOut, [min(imgOut(:)) max(imgOut(:))]); % values are scaled for disp
    colormap(gray(256));
    colorbar; axis image;
    xlabel('x [pixel]'); ylabel('y [pixel]');zlabel('[LSB]');
    s = 'Thresholded image';
    title(s); set(gcf,'Name',s);

end
```

filterImage.m

```
function imgOut = filterImage(imgIn, filterWindowSize, makeplot)
%*************************************************************************
%
% Function description:
% Enhance the image from image sensor in order to apply centroiding.
%
% Inputs:
%   - imgIn         : image from image sensor
%
% Outputs:
%   - imgOut        : filtered / enhanced image
%*************************************************************************

%*************************************************************************
% Filter
%*************************************************************************
% use gaussian filter applied to both direction succesively in order to
% have a circularly symmetric filter

sigma = filterWindowSize/3 /2; % +-3*sigma interval = 99.7% energy
half_interval = filterWindowSize/2;
x = linspace(-half_interval, half_interval,filterWindowSize);
g = 1/(sqrt(2*pi)*sigma) * exp(-1/2*x.^2/(sigma)^2);
g = g / sum(g); % integral of the gaussian is 1

% Filter x and y direction
img_filtered = filter2(g,imgIn,'same');
img_filtered = filter2(g',img_filtered,'same');
[sy sx] = size(img_filtered);    % image size

imgOut = img_filtered;


%*************************************************************************
% Plot
%*************************************************************************

if makeplot

%     figure
%     stem(x,g)
```

```
%     xlabel('[pixel]');
%     s = 'Image Filter taps';
%     title(s); set(gcf,'Name',s);

    figure
    [x,y] = meshgrid(1:sx,1:sy); % create grid
    surf(y,x,img_filtered,'EdgeColor','None')   % Invert x y axes for imaging IRF
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]');zlabel('[LSB]');
    s = 'Filtered image (3D)';
    title(s); set(gcf,'Name',s);

    figure;
    imagesc(img_filtered, [min(img_filtered(:)) max(img_filtered(:))]); % values a
    colormap(gray(256));
    colorbar; axis image;
    xlabel('x [pixel]'); ylabel('y [pixel]');zlabel('[LSB]');
    s = 'Filtered image';
    title(s); set(gcf,'Name',s);

end
```

generateImageSensorNoise.m

```
function noise = generateImageSensorNoise(intensity, integration_time, Temperature
%**************************************************************************
%
% Function description:
% Generate the noise of the image sensor according to datasheet parameters.
%
% Inputs:
%
%   - camSize_xy        : camera or image size in pixels.
%   - intensity         : [W/m^2] light intensity value from optic on image sensor
%                         pixels, camSize_y x camSize_x matrix
%
%   - integration_time  : [s] camera integration time
%   - Temperature       : [K] Sensor temperature
%   - makeplot
%
% Outputs:
%   - noise          : noise in the image sensor [V]
%
% Note : - RMS_value^2 = mean^2 + std^2
%        - The mean of all noise is zero (since the mean is the signal
%          itself and the noise the deviation).
%          -> noise_RMS = noise_std


%**************************************************************************
% load parameters
CameraParameters;

[m n] = size(intensity);
camSize_yx = [m n];

%**************************************************************************
% 1) Shot Noise:
%**************************************************************************
% - follow a poisson distribution on the number of generated electrons
% - the RMS value is different for each pixel depending in the light
%   intensity (number of generater electrons)
Nb_electron_avg = round(intensity*integration_time*Pixel_size^2*avgLambda*avgQEFF/
    /(h*LigthSpeed)); %[e-] average generated electrons per pixel
```

```
% Compute poisson distribution in [e-]
% Nb_electron = zeros(camSize_yx);
% for x = 1:camSize_yx(2)
%     for y = 1:camSize_yx(1)
%         Nb_electron(y,x) = randPoisson(Nb_electron_avg(y,x)); %[e-] generated el
%     end
% end
Nb_electron = randPoissonImage(Nb_electron_avg);


% Compute shot noise by removing average
n_shot_electron = (Nb_electron - Nb_electron_avg); %[e-]
n_shot = n_shot_electron * q_electron/Capacitance; %[V]

%************************************************************************
% 2) Reset / thermal /KTC noise
%************************************************************************
n_KTC_RMS = sqrt(KB*Temperature/Capacitance);   %[V]
n_KTC = n_KTC_RMS * randn(camSize_yx);          %[V]

%************************************************************************
% 3) Quantization noise from ADC
%************************************************************************
ADC_resolution = Vref_ADC/2^ADC_nb_bit; %[V/LSB]
n_ADC_RMS = ADC_resolution/sqrt(12);    %[V]
n_ADC = n_ADC_RMS * randn(camSize_yx);  %[V]

%************************************************************************
% 4) Flicker noise from internal amplification
%************************************************************************
% - neglected for now  !!

%************************************************************************
% 5) Dark current noise (not a real noise)
%************************************************************************
% - composed of the dark current + dark current non uniformity (DSNU=noise)
% - the noise can be considered as a poisson distribution / shot noise, but
%   since not all pixels have the same impurity concentration, ... this value
%   would be lower than the actual variation. The sensor datasheet can
%   provide information here.
% - temperature dependant, according to an Arhenius law (first approx):
%   Vrate = A*exp(-Ea/KB/T)
% - we consider here that the DSNU is applied on the activation Energy of
%   the Arhenius law with a gaussian distribution and a 3 sigma value
```

```
%   mentionned as the max value in the sensor datasheet

% Compute Arhenius parameters:
A = V_dark_rate_avg*exp(Ea_avg*q_electron/KB/Tref_idark); %[V/s]
% Ea distribution:
Delta_Ea_max = abs(KB/q_electron*Tref_idark*log(1/(DSNU_max/100+1))); % [eV] 3 sig
Ea = Delta_Ea_max/3 *randn(camSize_yx) + Ea_avg;   % [eV], add 3 sigma variation

% Compute dark current
V_dark_rate = A * exp(-Ea*q_electron/KB/Temperature); %[V/s] rate for each pixel
V_dark = V_dark_rate*integration_time;  % [V] dark voltage on pixels
n_dark = V_dark - mean(V_dark(:));      % [V] dark current noise, substract the me



%************************************************************************
% 6) Fixed Patter Noise (FPN)
%************************************************************************
% - Supposed to be eliminated by correlated double sampling (CDS)

%************************************************************************
% 7)Total noise :
%************************************************************************
% - The standard deviation or RMS value is summed in a root
%   mean square sum

% noise = n_shot  + n_KTC + n_ADC + n_dark; % [V]Use noise only, average is 0.
 noise = n_shot  + n_KTC + n_ADC + V_dark;   % [V] Use total value of dark voltage


%************************************************************************
% Plots
%************************************************************************
if makeplot

    % only for debug/comparizon of Ea:
    dT =7.5; %[ C ]
    T = [273-45:1:273+60];
    x = A*exp(-Ea_avg*q_electron/KB./T); %[V/s]
    x0 = x(T == round(Tref_idark));
    y = x0 *2.^((T-Tref_idark)/dT); % simple exponential law for comparison
    figure
    plot(T-273,x,'b',T-273,y,'r')
    legend('Ea','T exp')
```

```
xlabel('T[ C ]'); ylabel('Vdark [V]');
s = 'Dark Voltage for different methods versus temperature';
title(s); set(gcf,'Name',s);


n_shot_RMS_max = sqrt(max(Nb_electron_avg(:))) * q_electron/Capacitance; %[V]
n_dark_RMS = rms(n_dark);
noise_ = n_shot  + n_KTC + n_ADC + n_dark; % Use noise only, average is 0.
noise_RMS = rms(noise_);
figure
y = ones(2,1)*[n_shot_RMS_max, n_KTC_RMS, n_ADC_RMS,n_dark_RMS, noise_RMS];
plot(y);
legend(['shot max';'KTC     ';'ADC     ';'dark    ';'total   ']);
ylabel('[V]');
s = 'Noise RMS values';
title(s); set(gcf,'Name',s);


figure
subplot(2,1,1)
plot([n_dark(:), n_KTC(:), n_ADC(:) , n_shot(:)])
legend(['dark';'KTC ';'ADC ';'shot']);
xlabel('x-y [pixel]'); ylabel('Noise level [V]');
s = 'Noise over all image (1D)'; title(s);
set(gcf,'Name',s);
subplot(2,1,2)
threesigma = 3 * std(noise(:));
hold all
plot(noise(:))
plot([1, length(noise(:))],[threesigma threesigma]);
xlabel('x-y [pixel]'); ylabel('Noise level [V]');
legend('Total noise','3sigma value');
s = 'Total Noise over all image (1D)'; title(s);

figure
subplot(2,1,1)
plot(V_dark_rate(:))
xlabel('x-y [pixel]'); ylabel('V_dark_rate [V/s]');
s = 'Dark current / Voltage over all image (1D)'; title(s);
set(gcf,'Name',s);
subplot(2,1,2)
plot(V_dark(:))
xlabel('x-y [pixel]'); ylabel('V_dark [V]');
```

```
    figure
    [x,y] = meshgrid(1:camSize_yx(2),1:camSize_yx(1)); % create grid
    surf(y,x, noise)   % Invert x y axes for imaging IRF conventions
    colorbar; axis square;
    xlabel('y [pixel]'); ylabel('x [pixel]'); zlabel('[V]');
    s = 'Noise on image (3D)';
    title(s); set(gcf,'Name',s);



end
```

generateImageSensorOutput.m

```
function [img_bit, noise_bit] = generateImageSensorOutput(star_mxy, camSize_xy,PSF
%**************************************************************************
%
% Function description:
% Creates the image generated by the star-tracker image sensor taking into
% account noise, optics, sensors properties,...
%
% Inputs:
%   - star_mxy      : 1 x nb_step cell array containing nb_star x 3 matrix.
%                     Contains magnitude, x and y ideal position for every
%                     star at every time step.
%                     Unit is [pixel (non integer values)]
%                     If a star goes outside of the
%                     image, nb_star just get lower for that time step.
%                     - star magnitude [normalized star magnitude]
%                     - Position x=0 y=0 is the centre of the Field of view
%                       of the sensor
%                         ^
%                         |
%                         |
%                       (0,0)----->
%
%                     - A constant time step is supposed:
%                       time step = total_integration_time / nb_step.
%
%   - camSize_xy    : camera or image size in pixels.
%   - PSF_width     : Full width of the PSF [pixel]
%   - PSF_shape     : shape type / method of the PSF:
%                       - 'SimData' --> Use optical simulation data
%                       (detailed, non symmetrical); rotate PSF if required!
%                       - 'Custom' --> Use old basic data of PSF
%                       (symmetrical)
%                       - 'rectangular' or 'gaussian' basic shapes
%   - total_integration_time : [s]
%   - Temperature   : [K] temperature of the image sensor
%   - makeplot
%
% Outputs:
%   - image         : image object Matrix (camSize_xy) in IRF [pixel,
%                     integer values], image returned by the image sensor,
%                     0 = black
```

```
%*************************************************************************

%*************************************************************************
%  Parameters
%*************************************************************************

CameraParameters;
Img_disp_nb_bit = 8;  % Number of bit for displaying image: Apparently
                      % limited to 8 bit in Matlab...

%*************************************************************************
%  Generate Image
%*************************************************************************
% Initialize image intensity
intensity = zeros(fliplr(camSize_xy));  % [W/m2] average intensity for every pixel

% Get number of step
nb_step = length(star_mxy);
% time_step = total_integration_time/nb_step;


if strcmp(PSF_shape,'SimData') % use method relying on PSF simulations data

    % Note that the loops must be in that order, because a star can go out
    % of the image and disappear
    for step = 1:nb_step   % compute the average intensity for every time step

        star_magn = star_mxy{step}(:,1);    % get star magnitudes
        star_xy_irf = star_mxy{step}(:,2:3); % get star position [pixel]
        nb_star = size(star_xy_irf,1);   % get number of star at this time step

        % Convert coodinates to use same reference frame
        star_xy_irf(:,1) = (star_xy_irf(:,1) - (CMOSsizeX+1)/2)*Pixel_size; % x [m
        star_xy_irf(:,2) = (star_xy_irf(:,2) - (CMOSsizeY+1)/2)*Pixel_size; % y [m

        % Generate image for each star
        for star_index = 1:nb_star

            % Compute star power
            smagn = star_magn(star_index);  %value for eval function
            star_power = eval(TotalStarPowerFunction); % [w] total star power

            % Get rotated psf
            [pixels] = double(PSFpixels(star_xy_irf(star_index,1), star_xy_irf(sta
```

```
            % Scale to right value
            pixels = pixels * star_power / nb_step; % [W/m2] average value for the

            % Add all stars at every time step
            intensity = intensity + pixels; % [W/m2]
        end
    end

else % Use old method relying on PSF estimates

    % Compute 2D PSF normalized shape
    persistent psf_x_2D psf_y_2D psf_value_2D;  % to increase speed in case of mul

    if isempty(psf_x_2D)
        [psf_x_2D, psf_y_2D, psf_value_2D] = PSF_shape_2D(Pixel_size, PSF_pitch, P
    end
    numel_per_pixel = (1/PSF_pitch)^2; % number of element in PSF matrix per pixel

    for step = 1:nb_step   % comput the average intensity for every time step

        star_magn = star_mxy{step}(:,1);    % get star magnitudes
        star_xy_irf = star_mxy{step}(:,2:3); % get star position [pixel]
        [nb_star, n] = size(star_xy_irf);   % get number of star at this time step

        % Generate image for each star
        for star_index = 1:nb_star

            % Compute star power
            smagn = star_magn(star_index);  %value for eval function
            star_power = eval(TotalStarPowerFunction); % [w] total star power

            psf_intensity = psf_value_2D * star_power / nb_step; % [W/m2] average

            % Compute star position / spread
            % Star position [pixel, non integer value], negative value is possible
            % Use only 1 row/columns on the psf position matrix.
            x = star_xy_irf(star_index,1) + psf_x_2D(1,:);  % row vector
            y = star_xy_irf(star_index,2) + psf_y_2D(:,1);  % column vector

            x_min = round(x(1));        % [pixel, integer] round to nearest intege
            x_max = round(x(length(x))); % [pixel, integer] round to nearest integ
            y_min = round(y(1));
            y_max = round(y(length(y)));
```

```
        % handle x y out of range values
        if x_min < 1
            x_min = 1;
        end
        if y_min < 1
            y_min = 1;
        end
        if x_max > camSize_xy(1)
            x_max = camSize_xy(1);
        end
        if y_max > camSize_xy(2)
            y_max = camSize_xy(2);
        end

        % Compute average value for each pixel
        for img_x = x_min:x_max      % [integer pixel values], pixel for which
            for img_y = y_min:y_max

                % min max values for the interval where average is computed (1
                avg_x_min = img_x - 0.5;
                avg_x_max = img_x + 0.5;
                avg_y_min = img_y - 0.5;
                avg_y_max = img_y + 0.5;

                % Find indexes corresponding to this range
                x_indexes = x>= avg_x_min & x<avg_x_max;    % (with a [a,b] ir
                y_indexes = y>= avg_y_min & y<avg_y_max;    % in some cases af

                % Compute average and place it on the image
                intensities = psf_intensity(y_indexes,x_indexes);
                avg_pixel = mean(intensities(:));

                % Correct average in cases part of the values are zeros
                % (outside of the PSF 2D function)
                number_indexes = sum(x_indexes)*sum(y_indexes);
                if number_indexes < numel_per_pixel
                    avg_pixel = avg_pixel * number_indexes/numel_per_pixel;
                end

                intensity(img_y,img_x) = intensity(img_y,img_x) + avg_pixel; %
            end
        end
    end
```

```
    end

end

% For debug:
% s = sum(intensity(:)*Pixel_size^2); % total power

%************************************************************************
%  Generate Noise or use file from measurements
%************************************************************************
ADC_resolution = Vref_ADC/(2^ADC_nb_bit-1); %[V/LSB]

% Add background noise
%intensity = intensity + Background_intensity*(1+Background_NU/100*randn(fliplr(ca
% Add other noise

% Use noise file from measurements
% unit is [DN]
noise_bit = readSensorNoise(total_integration_time, Temperature);

%************************************************************************
% Generate final image
%************************************************************************

% Perform conversions to convert image from [W] to output of image sensor
% [bit]
% Take into account photo response non uniformity (PRNU)
Sensitivity_PRNU = Sensitivity *(1 + PRNU_RMS/100*randn(fliplr(camSize_xy))); %[DN
imgDN = intensity .* Sensitivity_PRNU * total_integration_time; % [DN]

% Take into account ADC quantization and add noise
img_bit = round(imgDN + noise_bit);  % [LSB]
saturation = img_bit > (2^ADC_nb_bit-1);
img_bit(saturation) = 2^ADC_nb_bit-1;

% Saturate image output to maximum ADC value
% saturation_indexes = img_bit > 2^ADC_nb_bit-1;
% img_bit(saturation_indexes) = 2^ADC_nb_bit-1;


%% ************************************************************************
% Generate plots if required
%************************************************************************
if makeplot
```

```
    figure;
    imagesc(intensity, [0 max(intensity(:))]); % values are scaled for display
    colormap(gray(2^Img_disp_nb_bit));
    colorbar; axis image;
    xlabel('x [pixel]'); ylabel('y [pixel]');zlabel('[W/m^2]');
    s = 'Light intensity on image sensor';
    title(s); set(gcf,'Name',s);

    figure
    [x,y] = meshgrid(1:camSize_xy(1),1:camSize_xy(2)); % create grid
    surf(y,x,intensity)   % Invert x y axes for imaging IRF conventions
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]'); zlabel('[W/m^2]');
    s = 'Light intensity on image sensor (3D)';
    title(s); set(gcf,'Name',s);

    figure;
    imagesc(img_bit, [min(img_bit(:)) max(img_bit(:))]); % values are scaled for d
    colormap(gray(2^Img_disp_nb_bit));
    colorbar; axis image;
    xlabel('x [pixel]'); ylabel('y [pixel]'); zlabel('[LSB]');
    s = 'Final image sensor output image';
    title(s); set(gcf,'Name',s);

    figure
    [x,y] = meshgrid(1:camSize_xy(1),1:camSize_xy(2)); % create grid
    surf(y,x,img_bit)   % Invert x y axes for imaging IRF conventions
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]');zlabel('[LSB]');
    s = 'Final image sensor output (3D)';
    title(s); set(gcf,'Name',s);

end

%*************************************************************************
%  Internal Functions
%*************************************************************************

function [psf_distance, psf_value] = PSF_shape_1D(pitch, width, shape, makeplot)
% Optical Point spread function: Compute the shape of the PSF in the 1D
% case. The output vectors should be symmetric from psf_distance = 0 !
%
% Inputs:
```

```
% - pitch        : Pitch for computing PSF values [pixel, non integer value]
%                   (1/pitch = pixel oversampling), psf_distance increments.
% - width        : Full width of the PSF [pixel]
% - shape        : shape type of the PSF
%
% Outputs:
% - psf_distance : linearly spaced distance from -x to +x [pixel]
% - psf_value    : point spread function values (not normalized, because it
%                   is done later in the 2D function anyway).

switch shape
    case 'gaussian'
        sigma = width/3 /2; % +-3*sigma interval = 99.7% energy
        half_interval = 1.0*width/2;
        psf_distance = 0:pitch:half_interval;
        psf_value = 1/(sqrt(2*pi)*sigma) * exp(-1/2*psf_distance.^2/(sigma)^2); %

    case 'rectangular'
        psf_distance = 0:pitch:width/2;
        psf_value = ones(1,length(psf_distance)); % rectangular shape

    case 'custom'

%         psf_distance = [0 0.067   0.139   0.416   0.444   0.555   0.694   0.943
%         psf_value =    [1 0.028   0.074   0.010   0.010   0.042   0.007   0.027
        % with blue filter
        psf_distance =  [0  0.178   0.296   0.533   0.651   0.888   1.421   1.895
        psf_value = [0.705882353    0.150   0.157   0.078   0.078   0.007   0.033


        d = 0:pitch:max(psf_distance);
        psf_value = interp1(psf_distance,psf_value, d, 'linear',0);
        psf_distance = d;

    otherwise
        error('Shape of PSF is unknown');
end

if makeplot
    figure
    psf_value_plot = psf_value/sum(psf_value)/pitch; % scale value to insure an ir
    plot(psf_distance,psf_value_plot,'r')
    xlabel('[pixel]');
    s = 'PSF 1D (normalized)';
```

```
    title(s); set(gcf,'Name',s);
end


%************************************************************************
function [psf_x_2D, psf_y_2D, psf_value_2D] = PSF_shape_2D(Pixel_size, pitch, widt
% Optical Point spread function in 2D. The PSF is supposed to be the same
% than the 1D PSF function but rotated over its center peak. Interpolates
% the 1D PSF
%
% Inputs:
% - Pixel_size    : [m]
% - pitch         : Pitch for computing PSF values [pixel, non integer value]
%                   (1/pitch = pixel oversampling), psf_distance increments.
% - width         : Full width of the PSF [pixel]
% - shape         : shape type of the PSF
%
% Outputs:
% - psf_x_2D      : Square matrix, linearly spaced distance from -x
%                   to +x [pixel], increment along x only (constant along y)
% - psf_y_2D      : Square matrix, linearly spaced distance from -y
%                   to +y [pixel], increment along y only (constant along x)
% - psf_value     : [1/m2] Normalized value. The integral of psf_value is 1
%                   or a specified value over psf_distance [m]. Square matrix.

[psf_distance, psf_value] = PSF_shape_1D(pitch, width, shape, makeplot);

%Generate symmetric values for the meshgrid
i=1;
if psf_distance(1)==0
    i=2;
end
psf_distance = [fliplr(-psf_distance(i:end)) psf_distance];
psf_value = [fliplr(-psf_value(i:end)) psf_value];

% Generate mesgrid and interpolate
[x,y] = meshgrid(psf_distance); % create grid for interpolating points
d = sqrt(x.^2+y.^2);    % Compute distances
z = interp1(psf_distance,psf_value, d, 'linear',0); % interpolate 1D PSF at value

% PSF / Energy scaling
if (strcmp(shape,'gaussian') || strcmp(shape,'rectangular'))
    psf_value_2D = z/sum(z(:))/(Pixel_size*pitch)^2; % scale value to insure an in
else
```

```
    ix0 = find(x(1,:)==0);   % find the 0 in the array
    iy0 = find(y(:,1)==0);
    ix = ix0 + (-EnergySquareSize/2/pitch:EnergySquareSize/2/pitch); % indexes of
    iy = iy0 + (-EnergySquareSize/2/pitch:EnergySquareSize/2/pitch);
    energy = sum(sum(z(iy,ix)))*(Pixel_size*pitch)^2;

    psf_value_2D = z/energy*EnergyRatioInSquare;    % scale to have EnergyRatioInS
end

psf_x_2D = x;
psf_y_2D = y;

disp('- Display total enery of PSF for info, should be <= 1');
E_tot_PSF = sum(psf_value_2D(:))*(Pixel_size*pitch)^2

if makeplot
    figure
    surf(psf_x_2D,psf_y_2D,psf_value_2D,'EdgeColor','none')
    colorbar; axis image;
    xlabel('[pixel]'); ylabel('[pixel]');
    s = 'PSF 2D (normalized)';
    title(s); set(gcf,'Name',s);
%
%     figure;
%     axislimits = [min(psf_x_2D(:)) max(psf_x_2D(:))];
%     imagesc(axislimits,axislimits,psf_value_2D, [0 max(psf_value_2D(:))]); % val
%     colormap(gray(2^Img_disp_nb_bit));
%     colorbar; axis image;
%     xlabel('x [pixel]'); ylabel('y [pixel]');
%     s = 'Point spread function';
%     title(s); set(gcf,'Name',s);

end
```

generateNoiseHuffman.m

```
%*************************************************************************
%
% Function description:
% Generate the noise of the image sensor according to Huffman's thesis
% model.
% Huffman considers the following noise sources: signal shot noise,
% dark current noise, background noise, quantization noise and readout
% noise.
% Missing: thermal noise.
%
% Usage:
%
% Inputs:
%
% Outputs:
%
%



% ADC_resolution = Vref_ADC/2^ADC_nb_bit; %[V/LSB]
% Nvolt = ADC_resolution * N; % [V]
% Nelectron = Nvolt* Capacitance/q_electron; % [e-]


function [noise] =  generateNoiseHuffman ...
                    (intensity, integration_time, Temperature)

% load parameters
CameraParameters; %version modified by EF of CameraParameters,

[m n] = size(intensity);
camSize_yx = [m n];

%*************************************************************************
% 1) Shot Noise: (same as Herv, Huffman's depends on Optics, which are not
% modeled yet)
%*************************************************************************
% - follow a poisson distribution on the number of generated electrons
% - the RMS value is different for each pixel depending in the light
%   intensity (number of generater electrons)
```

```
Nb_electron_avg = round(intensity*integration_time*Pixel_size^2*avgLambda*avgQEFF/
    /(h*LigthSpeed)); %[e-] average generated electrons per pixel
Nb_electron = randPoissonImage(Nb_electron_avg);

% Compute shot noise by removing average
nShotElectron = (Nb_electron - Nb_electron_avg); %[e-]
nShot = nShotElectron * q_electron/Capacitance; %[V]


%************************************************************************
% 2) Dark current noise (based on Poisson distribution)
%************************************************************************
%
% Compute Arhenius parameters:
A = V_dark_rate_avg*exp(Ea_avg*q_electron/KB/Tref_idark); %[V/s]
% Ea distribution:
Delta_Ea_max = abs(KB/q_electron*Tref_idark*log(1/(DSNU_max/100+1))); % [eV] 3 sig
Ea = Delta_Ea_max/3 *randn(camSize_yx) + Ea_avg;   % [eV], add 3 sigma variation
% Compute dark current
darkCurrentRate = A * exp(-Ea*q_electron/KB./Temperature); %[V/s] rate for each pi
V_Dark = darkCurrentRate.*integration_time;
nDark = V_Dark - mean(V_Dark(:));    % [V] dark current noise, substract...
                                  %the mean in order to get the noise only% Compu

% sigmaDark = sqrt(darkCurrentRate.*integration_time);
% lambdaDark = sigmaDark.^2;
% nDark = randPoissonImage(lambdaDark); %[V]

% value for LUPA300
% n_dark_electron = 8823; [electrons] valid at 21 C
% Value comes from excel file with comparison of all CMOS
% sensors.


%************************************************************************
% 3) Background noise
%************************************************************************
% Neglected
% Background_intensity =9.92e-13; % [W/m2] space background light intensity
% at the image sensor
nBackground = 0;

%************************************************************************
% 4) Quantization noise
```

```
%************************************************************************
ADU_max = 2^ADC_nb_bit; % ADC_nb_bit is the number of bit of the A/D
                        % converter, stored in CameraParameters.m
% fullWellPixel comes from the CameraParameters.m file
delta = fullWellPixel/ADU_max;
nQuantizationElectron = delta/sqrt(12); % [e-]
nQuantization = nQuantizationElectron * q_electron/Capacitance * ...
                randn(camSize_yx);  %[V]


%************************************************************************
% 5) Readout noise
%************************************************************************
%from the datasheet
nReadoutElectrons = nReadoutElectrons / delta; % [V]
nReadout = nReadoutElectrons * q_electron/Capacitance * ...
                randn(camSize_yx); %[electrons]

%************************************************************************
% 6) Reset / thermal /KTC noise
%************************************************************************
n_KTC_RMS = sqrt(KB*Temperature/Capacitance);   %[V]
nKTC = n_KTC_RMS * randn(camSize_yx);          %[V]



% final result
noise = nShot + nDark + nBackground + nQuantization + nReadout + nKTC; %[V]


end
```

GetStarFromSensorModel.m

```
function [est_star_mxy] = GetStarFromSensorModel(star_mxy,...
                            integration_time, Temperature)
%**************************************************************************
%
% Function description:
% Complete model of the image sensor and centroiding process. From the star
% position and magnitude, it generates the image outputted by the image
% sensor, then process it to obtain the estimated star position and
% magnitude.
% Note that is is possible to have a rotation during the integration time.
% In that case, it is necessary to provide to this function the different
% position of the star among time with steps. It is assume that the amount
% of steps is high enough in order not to have to do any interpolation.
%
% Inputs:
%   - star_mxy      : 1 x nb_step cell array containing nb_star x 3 matrix.
%                     Contains magnitude, x and y ideal position for every
%                     star at every time step.
%                     Unit is [pixel (non integer values)]
%                     If a star goes outside of the
%                     image, nb_star just get lower for that time step.
%                     - star magnitude [normalized star magnitude]
%                     - Position x=0 y=0 is the centre of the Field of view
%                       of the sensor
%                         ^
%                         |
%                         |
%                       (0,0)----->
%
%                     - A constant time step is supposed:
%                       time step = total_integration_time / nb_step.
%   - total_integration_time : [s]
%   - Temperature:    [K] Sensor temperature
%
% Outputs:
%   - est_star_mxy  : nb_found_star x 3 matrix. For the star that have
%                     been found, it contains the estimated star magnitude,
%                     x and y position [pixel]
%
%**************************************************************************
```

```
% star_mxy = {[4 20.5 20 ]...
%              [4 20.5 20 ],...
%              [4 20.5 20]...
%            };
% integration_time = 0.15; %[s]

% Parameters:
CameraParameters;

%  - Image generation:
PSF_width = 6;              %[pixel]
PSF_shape = 'SimData';
% PSF_shape = 'custom';
% NoiseImageFile = '';    % Use noise model
makeplotImageGen = 0;
makeplotImageGenOut = 0;

clipLines = 5;
% - Image Enhancement:
sigmaWindowSize = 3; %[pixel]
sigmaThreshold = 1.5;
minPixFraction = 0.2;
gaussianWindowSize = 5;       %[pixel]
makeplotFilter   = 0;
SigmaNoiseThreshold = 2.3;

% - Centroiding:
numiterCentroiding = 1;     % total number of centroiding iterations
centroidingWindowSize = PSF_width+1.5;  %[pixel]
makeplotCentroid = 0;

%************************************************************************
% Convert input star position (offset)
for nstep = 1:length(star_mxy)
    star_mxy{nstep}(:,2) = star_mxy{nstep}(:,2) + (CMOSsizeX+1)/2; % x
    star_mxy{nstep}(:,3) = (CMOSsizeY+1)/2 - star_mxy{nstep}(:,3); % y
end

% Create camera image
[img, noise] = generateImageSensorOutput(star_mxy, [CMOSsizeX CMOSsizeY],PSF_width

if makeplotImageGenOut
    figure;
    imagesc(img, [min(img(:)) max(img(:))]); % values are scaled for display
```

```
    colormap(gray(2^8));
    colorbar; axis image;
    xlabel('x [pixel]'); ylabel('y [pixel]'); zlabel('[LSB]');
    s = 'Final image sensor output image';
    title(s); set(gcf,'Name',s);

    figure
    [x,y] = meshgrid(1:size(img,2),1:size(img,1)); % create grid
    surf(y,x,img,'EdgeColor','None')   % Invert x y axes for imaging IRF conventio
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]');zlabel('[LSB]');
    s = 'Final image sensor output image (3D)';
    title(s); set(gcf,'Name',s);

    figure
    [x,y] = meshgrid(1:size(noise,2),1:size(noise,1)); % create grid
    surf(y,x,noise,'EdgeColor','None')   % Invert x y axes for imaging IRF convent
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]');zlabel('[LSB]');
    s = 'Noise (3D)';
    title(s); set(gcf,'Name',s);
end

% Zero-out the noisy top and bottom 10 lines from the image
img(1:clipLines,:) = 0;
img(size(img,1)-clipLines:size(img,1),:) = 0;

% Zero-out the noisy top and bottom 10 lines from noise
% (so that the thresold is correctly calculated)
noise(1:clipLines,:) = 0;
noise(size(noise,1)-clipLines:size(noise,1),:) = 0;
% Select threshold level
noiseThreshold = SigmaNoiseThreshold*std(noise(:))+ mean(noise(:)); % sigma value

% Filter image
%img = enhanceImageSensorOutput(img, gaussianWindowSize, noiseThreshold, sigmaWinc
img = enhanceImageSensorOutput(img, noiseThreshold, sigmaWindowSize, sigmaThreshol

% Perform Centroiding and star detection
[found_star_xy_irf, est_magn, nb_star] = centroiding(img,centroidingWindowSize,num

%Correct for optical distortion
nbStars = size(found_star_xy_irf,1);
for i = 1:nbStars
```

```
    found_star_xy_irf(i,:) = OpticalDistortionCompensation2(found_star_xy_irf(i,1)
end

% Convert star position
found_star_xy_irf(:,1) =  found_star_xy_irf(:,1) - (CMOSsizeX+1)/2;
found_star_xy_irf(:,2) =  -found_star_xy_irf(:,2) + (CMOSsizeY+1)/2;

% Output data
est_star_mxy = [est_magn,found_star_xy_irf];
```

main_detect_star_test.m

```
CameraParameters;

%  - Image generation:
PSF_width = 7;              %[pixel]
Temperature = 21 + 273.15;    %[K] Sensor temperature
PSF_shape = 'SimData';
% PSF_shape = 'custom';
% NoiseImageFile = '';    % Use noise model
NoiseImageFile = 'dark_+21.0deg_Tint100ms.bmp';     % Use noise image
makeplotImageGen = 0;
makeplotImageGenOut = 0;

clipLines = 5;
% - Image Enhancement:
sigmaWindowSize = 3; %[pixel]
sigmaThreshold = 1.5;
minPixFraction = 0.2;
gaussianWindowSize = 5;       %[pixel]
makeplotFilter   = 0;
SigmaNoiseThreshold = 3;

% - Centroiding:
numiterCentroiding = 3;     % total number of centroiding iterations
centroidingWindowSize = PSF_width+1.5;  %[pixel]
makeplotCentroid = 0;

%************************************************************************
% Convert input star position (offset)
% for nstep = 1:length(star_mxy)
%     star_mxy{nstep}(:,2) = star_mxy{nstep}(:,2) + (CMOSsizeX+1)/2; % x
%     star_mxy{nstep}(:,3) = (CMOSsizeY+1)/2 - star_mxy{nstep}(:,3); % y
% end

% Create camera image
%[img, noise] = generateImageSensorOutput(star_mxy, [CMOSsizeX CMOSsizeY],PSF_widt
%read star and noise data from image files
img = imread('ND70_+21.0deg_Tint100ms.bmp');
noise = imread('dark_+21.0deg_Tint100ms.bmp');
img=double(img(:,:,1));
noise=double(noise(:,:,1));
```

```
if makeplotImageGenOut
    figure;
    imagesc(img, [min(img(:)) max(img(:))]); % values are scaled for display
    colormap(gray(2^8));
    colorbar; axis image;
    xlabel('x [pixel]'); ylabel('y [pixel]'); zlabel('[LSB]');
    s = 'Final image sensor output image';
    title(s); set(gcf,'Name',s);

    figure
    [x,y] = meshgrid(1:size(img,2),1:size(img,1)); % create grid
    surf(y,x,img,'EdgeColor','None')   % Invert x y axes for imaging IRF conventio
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]');zlabel('[LSB]');
    s = 'Final image sensor output image (3D)';
    title(s); set(gcf,'Name',s);

    figure
    [x,y] = meshgrid(1:size(noise,2),1:size(noise,1)); % create grid
    surf(y,x,noise,'EdgeColor','None')   % Invert x y axes for imaging IRF convent
    colorbar; axis image;
    xlabel('y [pixel]'); ylabel('x [pixel]');zlabel('[LSB]');
    s = 'Noise (3D)';
    title(s); set(gcf,'Name',s);
end

% Zero-out the noisy top and bottom 10 lines from the image
img(1:clipLines,:) = 0;
img(size(img,1)-clipLines:size(img,1),:) = 0;

% Zero-out the noisy top and bottom 10 lines from noise
% (so that the thresold is correctly calculated)
noise(1:clipLines,:) = 0;
noise(size(noise,1)-clipLines:size(noise,1),:) = 0;
% Select threshold level
noiseThreshold = SigmaNoiseThreshold*std(noise(:))+ mean(noise(:)); % sigma value

% Filter image
%img = enhanceImageSensorOutput(img, gaussianWindowSize, noiseThreshold, sigmaWind
img = enhanceImageSensorOutput(img, noiseThreshold, sigmaWindowSize, sigmaThreshol

% Perform Centroiding and star detection
[found_star_xy_irf, est_magn, nb_star] = centroiding(img,centroidingWindowSize,num
```

```
%Correct for optical distortion
nbStars = size(found_star_xy_irf,1);
for i = 1:nbStars
    found_star_xy_irf(i,:) = OpticalDistortionCompensation2(found_star_xy_irf(i,1)
end

% Convert star position
found_star_xy_irf(:,1) =  found_star_xy_irf(:,1) - (CMOSsizeX+1)/2;
found_star_xy_irf(:,2) =  -found_star_xy_irf(:,2) + (CMOSsizeY+1)/2;

% Output data
est_star_mxy = [est_magn,found_star_xy_irf]
```

main_image_sensor_test.m

```
%*************************************************************************
%
% Function description:
% Main file script for simulating the image generation process of the image
% sensor and the centroiding / image processing phases to find the stars on
% the image.


close all
clear
% clear all

% star_mxy = {[3 10 10; 3 9 9 ],...
%             [3 11 11; 3 10 10 ],...
%             [3 12 12; 3 11 11],...
%             [3 13 13; 3 12 12],...
%             [3 14 14; 3 13 13],...
%             };
star_mxy = {[1 -1 -1; 2 100 10],...
            [1 -3 -3; 2 105 10]

           };

integration_time = 0.15;     %[s]

%*************************************************************************
% Create camera image

[est_star_mxy] = GetStarFromSensorModel(star_mxy, integration_time)
found_star_xy_irf = est_star_mxy(:,2:3);

if size(est_star_mxy,1) > size(star_mxy{1},1)
    % Note that this test does not detect all false stars!
    warning('False star detected')
end

%*************************************************************************
% Analyse performances:

% Sort detected star to compute error
nb_step = length(star_mxy);
```

```
[nb_star m] = size(star_mxy{1});
avg_star_xy_irf = zeros(nb_star,2);
for step = 1:nb_step    % compute average position over time
    avg_star_xy_irf = avg_star_xy_irf + star_mxy{step}(:,2:3)/nb_step;
end

star_xy_sorted = sortrows(avg_star_xy_irf);
% found_star_xy_sorted = sortrows(found_star_xy_irf); % this step can be skipped

centroiding_error_inPixel = abs(star_xy_sorted-found_star_xy_irf)   %[pixel]
avg_error = mean(centroiding_error_inPixel(:))

% error_radius_inPixel = zeros(nb_star,1);
% for i = 1:nb_star
%     error_radius_inPixel(i) = norm(centroiding_error_inPixel(i,:));
% end
% error_radius_inPixel
```

main_investigate_performances.m

```
%*************************************************************************
%
% Function description:
% Main file script for simulating the image generation process of the image
% sensor and the centroiding / image processing phases to find the stars on
% the image.


% close all
clear all


star_magn = 1.7:0.1:8;
Temp = (-20:2:40)+273.15; %[s]
int_time = [0.05 0.1 0.15];% 0.25 0.4]; %[K] Sensor temperature

PSF_width = 9;              %[pixel]
PSF_shape = 'custom';

SNRmin = 2;
niter = 2;              % number of iterations for averaging result

signal_window_size = PSF_width-1;

star_xy = [25 25];        %[pixel]
camSize_xy  = [50 50];    %[pixel]

filterWindowSize = 5;       %[pixel]
centroidingWindowSize = PSF_width+1.5;  %[pixel]
numiterCentroiding = 3;     % total number of centroiding iterations
makeplotImageGen = 0;
makeplotFilter   = 0;
makeplotCentroid = 0;

%*************************************************************************


SNR = zeros(length(Temp),length(int_time));
minDetectStarCat = zeros(length(Temp),length(int_time));
j=0;
for integration_time = int_time
```

```
j=j+1;
i=0;
for Temperature = Temp
    i = i+1;
    k=0;
    while ~k || (SNR(i,j) >= SNRmin && k < length(star_magn))
        k=k+1;
        star_mxy = {[star_magn(k) star_xy]}; % create star cell
        SNR(i,j) = 0;

        for n=1:niter
            % Create camera image
            [img, noise] = generateImageSensorOutput(star_mxy,...
                camSize_xy,PSF_width, PSF_shape,integration_time,...
                Temperature,'', makeplotImageGen);

            % Filter
            img = filterImage(img, filterWindowSize, makeplotFilter);
            noise = filterImage(noise, filterWindowSize, makeplotFilter);

            % Select threshold level
            N = 3*std(noise(:)); % [LSB] 3sigma value, Noise level

            threshold = N; %

            %get X and Y coordinates of the window
            x_indexes = round(star_mxy{1}(:,2)-signal_window_size/2):round(sta
            y_indexes = round(star_mxy{1}(:,3)-signal_window_size/2):round(sta
            %the signal will be the average value of the window pixels
            S = mean(mean(img(y_indexes,x_indexes)));
            %SNR is the SNR from the previous iterations, plus the
            %value for the current iteration
            SNR(i,j) = SNR(i,j) + S/N/niter;    % compute average SNR over nit

            %             figure
            %             hold all
            %             plot(img(25,1:end))
            %             plot([1 50],[N N])
            %             plot([20 30], [S S])
            %             xlabel('[pixel]')
            %             ylabel('image sensor output [LSB]')
            %             legend('Total signal', 'N','S')

        end
```

```
        end

        if k == 1 && SNR(i,j) < SNRmin
            minDetectStarCat(i,j) = NaN;
        else
            minDetectStarCat(i,j) = star_magn(k);
        end
    end
end


Temp =  Temp-273.15;

figure
hold on
plot(Temp'*ones(1,length(int_time)),SNR)
plot([Temp(1) Temp(end)]', [1 2;1 2],'k:')
axis tight
legend(num2str(int_time'))
xlabel('Temperature [ C ]'); ylabel('SNR');
s =['PSF\_width=',num2str(PSF_width),', ',PSF_shape,', star magn=',num2str(star_mx
title(s);

figure
plot(Temp'*ones(1,length(int_time)),minDetectStarCat)
axis tight
legend(num2str(int_time'))
xlabel('Temperature [ C ]'); ylabel('min Detect Star Category');
s =['PSF\_width=',num2str(PSF_width),', ',PSF_shape];
title(s);




% % Filter image
% img = enhanceImageSensorOutput(img, filterWindowSize, threshold, makeplotFilter)
%
%
%
% % Perform Centroiding and star detection
% [found_star_xy_irf, est_magn, nb_star] = centroiding(img,centroidingWindowSize,r
%
% %*************************************************************************
% % Analyse performances:
```

```
%
% % Sort detected star to compute error
% nb_step = length(star_mxy);
% [nb_star m] = size(star_mxy{1});
% avg_star_xy_irf = zeros(nb_star,2);
% for step = 1:nb_step    % compute average position over time
%     avg_star_xy_irf = avg_star_xy_irf + star_mxy{step}(:,2:3)/nb_step;
% end
%
% star_xy_sorted = sortrows(avg_star_xy_irf);
% % found_star_xy_sorted = sortrows(found_star_xy_irf); % this step can be skipped
%
% centroiding_error_inPixel = abs(star_xy_sorted-found_star_xy_irf)   %[pixel]
% avg_error = mean(centroiding_error_inPixel(:))
%
% % error_radius_inPixel = zeros(nb_star,1);
% % for i = 1:nb_star
% %     error_radius_inPixel(i) = norm(centroiding_error_inPixel(i,:));
% % end
% % error_radius_inPixel
```

main_plot_noise_level.m

```
%*************************************************************************
%
% Function description:
% Main file script for simulating the image generation process of the image
% sensor and the centroiding / image processing phases to find the stars on
% the image.


% close all
clear all


NoiseImageFile = '';

star_magn = 4;
Temp = (-20:2:40)+273.15; %[K] Sensor temperature
int_time = [0.05 0.1 0.15 0.25 0.4]; %[s]

PSF_width = 9;              %[pixel]
PSF_shape = 'custom';

SNRmin = 2;
niter = 2;              % number of iterations for averaging result

signal_window_size = PSF_width-1;

star_xy = [25 25];        %[pixel]
camSize_xy  = [50 50];    %[pixel]

filterWindowSize = 5;       %[pixel]
centroidingWindowSize = PSF_width+1.5;  %[pixel]
numiterCentroiding = 3;     % total number of centroiding iterations
makeplotImageGen = 0;
makeplotFilter   = 0;
makeplotCentroid = 0;

%*************************************************************************


N = zeros(length(Temp),length(int_time));
j=0;
```

```
for integration_time = int_time
    j=j+1;
    i=0;
    for Temperature = Temp
        i = i+1;

        star_mxy = {[star_magn star_xy]}; % create star cell

        % Create camera image
        [img, noise] = generateImageSensorOutput(star_mxy, camSize_xy,...
                    PSF_width, PSF_shape,integration_time,Temperature,...
                    NoiseImageFile, makeplotImageGen);



%         % Filter
%         img = filterImage(img, filterWindowSize, makeplotFilter);
%         noise = filterImage(noise, filterWindowSize, makeplotFilter);

        % Noise
        N(i,j) = 3*std(noise(:)); % [LSB] 3sigma value, Noise level

    end
end

CameraParameters;
ADC_resolution = Vref_ADC/2^ADC_nb_bit; %[V/LSB]
Nvolt = ADC_resolution * N; % [V]
Nelectron = Nvolt* Capacitance/q_electron; % [e-]

Temp =  Temp-273.15;


figure
semilogy(Temp'*ones(1,length(int_time)),N)
grid
legend(num2str(int_time'))
xlabel('Temperature [ C ]'); ylabel('Noise level [LSB]');
s ='Noise level for different integration time [s] and temperature';
title(s);

figure
semilogy(Temp'*ones(1,length(int_time)),Nelectron)
grid
```

```
legend(num2str(int_time'))
xlabel('Temperature [ C ]'); ylabel('Noise level [e-]');
s ='Noise level for different integration time [s] and temperature';
title(s);
```

OpticalDistortionCompensation.m

```
% Compensates for shift in star centroid position caused by distortion
% caused by the lenses. The amount of distortion is correlated to the
% distance from the center of the sensor/optics. The function uses the
% calculated centroids of 7 PSFs given by Drewis as a measure of this
% distortion.
% In PSFpixels function, the expected optical pattern is generated based on
% the nearest available PSF. Therefore, the distortion in the simulation
% has only 7 possible values. In practise the distortion should vary
% as a continuous function of distance from the centre.
% Inputs:
%   X - Horizontal position (in pixels) of the detected centroid of the star
%   Y - Vertical position (in pixels) of the detected centroid of the star
%   method - Interpolation method to be used to calculate the distortion
%   for a particular detected star. 'nearest' should be used for simulation
%   purposes. In practise, 'cubic' and 'spline' are good approximations of
%   the actual distortion.
% Assumptions:
%   Since all the PSF's are lined up along a vertical line bisecting the
%   sensor, the distortion in the PSFs is only along the vertical (Y) axis.
%   Therefore, the relatively small variations in centroids along the X
%   axis are ignored.
function [correctedPosnXY] = OpticalDistortionCompensation(X, Y, method)
CMOSpixelSize = 9.9*10^-6; %[m]
CMOSsizeX = 480;
CMOSsizeY = 480;
CenterPixelX = CMOSsizeX/2;
CenterPixelY = CMOSsizeY/2;
PSFPositionY = [0 -0.783 -1.175 -1.567 -1.762 -1.9975 -2.35].*10^-3; % Positions o
PSFPixelPositionR = -1*(PSFPositionY / CMOSpixelSize);
%PSFPositionX = [0 0 0 0 0 0 0];
%PSFPixelPositionX = PSFPositionX / CMOSpixelSize + CenterPixelX;
PSFDataSizes = [1024 1024 1024 2048 2048 2048 2048]; % Number of data points along
%PSFCenterX = [512 512 512 1024 1024 1024 1024]; % Center data point in the PSF. P
PSFCenterY = [513 513 513 1025 1025 1025 1025];
PSFDataWidth = [101.279 100.455 99.387 138.661 137.311 135.408 132.020 ].*10^-6; %
%PSFCentroidsX = [511.802093505859, 512.175842285156, 512.099731445313,...
%    1022.19702148438, 1022.53417968750, 1026.86340332031, 1027.65258789063];
PSFCentroidsY = [513.060241699219, 467.787414550781, 448.448699951172,...
    883.361938476563, 869.465087890625, 851.833190917969, 835.838012695313];

PSFPointWidth = PSFDataWidth ./ PSFDataSizes;
```

```
%DifferenceX = (PSFCentroidsX - PSFCenterX) .* PSFPointWidth;
DifferenceY = (PSFCentroidsY - PSFCenterY) .* PSFPointWidth;
%CentroidPositionsX = PSFPositionX + DifferenceX;
%CentroidPositionsY = PSFPositionY + DifferenceY;

%PixelDifferenceX = DifferenceX / CMOSpixelSize;
PixelDifferenceY = DifferenceY / CMOSpixelSize;
PixelDifferenceR = PixelDifferenceY;
PixelDifferenceR(1) = 0;

if X==CenterPixelX && Y==CenterPixelY
    correctedPosnXY = [X Y];
    return;
end

[centerAngle, centerDistance] = cart2pol(X-CenterPixelX, Y-CenterPixelY);
correction = interp1(PSFPixelPositionR, PixelDifferenceR, centerDistance, method,
if isnan(correction)==1
    disp('WARNING!: Interpolated value is a NaN, returning original coordinates');
    correctedCenterDistance = centerDistance;
else
    correctedCenterDistance = centerDistance + correction;
end
[correctedPosnXY(1) correctedPosnXY(2)] = pol2cart(centerAngle, correctedCenterDis
correctedPosnXY = correctedPosnXY + [CMOSsizeX/2, CMOSsizeY/2];
```

PSFpixels.m

```
function [pixels] = PSFpixels(starX, starY)
%************************************************************************
% INPUT = x and y positions [m] of the star in real world coordinates with
% origin at the center of the sensor.
% OUTPUT = Normalized pixel data in a matrix of the size of the sensor.

% ----Assumptions and Notes----
% 1. PSF is an n-by-n matrix one data point maps as a square in the real
% world, otherwise the summations along X and Y directions are different
% and the rotation will be much more complex.
%
% 2. CMOS pixels are also square.
%
% 3. There is no distance in between the sensor pixels, otherwise some of
% the PSF elements should be ignored while accululating into pixels
% according to the overlap of the data with the real world pixel array.
%
% 4. Does sensor pixel sensitivity vary with where inside the pixel the
% photon falls? The datasheet seems to say nothing of this.
% Its assumed that it doesn't or is insignificant.
%
% 5. The PSF data contains little or no energy outside a radius of
% datasize/2 about its center. Rotation clips off part of the matrix.
% This is a safe assumption. If significant energy is actually present
% outside this radius, then the given PSF data should be larger, because in
% that case, the input matrix clips off a significant amount of the PSF's
% energy anyway For 45 degree rotation, the energy loss was found to be:
% 0.16%, 0.29% 0.54%, 0.12%, 0.15%, 0.24%, 0.83% for fields 1-7
% respectively. For 90 degree rotation, the loss was less than 45 degree.
%
% 6. The actual 7 PSF data matrices are loaded from "PSFsSingle.mat" file.
% This file should exist in the current directory of matlab. The function
% loads the PSFs as a persistent variable to avoid wastage of time for
% loading from file in subsequent runs. 'clear <function name>' (in this
% case,  'clear PSFpixels') will clear out any persistent variables used by
% that function. It is recommended to do this once this function is no
% longer in use during the simulation as it will free up a lot of memory.
%
% 7. If the given star position is so far from the center such that no data
% point of the PSFs overlaps with the sensor area, a zeros matrix is
% returned immediately.
```

```
%
% 8. The origin is at the center of the pixel array and the X and Y
% coordinates are as in this figure:
%                                   -Y
%                                   ^
%                                   |
%                                   |
%                           -X<-----.----->+X
%                                   |
%                                   |
%                                   v
%                                   +Y
% 9. Rotation of PSF is not performed for the center PSF, because it is
%    assumed to be symmetrical.
%
% ----PSF and Sensor Data----
persistent PSFs; % make PSFs persistent so they remain in memory even after the fu
% If the PSF data doesn't exist, load it.
if isempty(PSFs) == 1
    % The PSFsSingle contains single precision floating point PSFs, all
    % susequent calculations here generate single precision results.
    load('PSFsSingle.mat');
end
CameraParameters; % Load parameters
CMOSpixelSize = Pixel_size; %[m]
% CMOSsizeX = 480;
% CMOSsizeY = 480;

% Following are the test matrices to act as PSFs
%PSFs = {magic(1024) magic(1024) magic(1024) magic(2048) magic(2048) magic(2048) m
%gallery('triw',[1024 1024],1,512)
%upper = gallery('triw',[1024 1024],1,1024);
%upper2 = gallery('triw',[2048 2048],1,2048);
%PSFs = {upper upper upper upper2 upper2 upper2 upper2};
%ones1 = ones(1024);
%ones2 = ones(2048);
%PSFs = {ones1 ones1 ones1 ones2 ones2 ones2 ones2};

PSFPositionY = [0 -0.783 -1.175 -1.567 -1.762 -1.9975 -2.35].*10^-3; % Positions o
PSFPositionX = [0 0 0 0 0 0 0];
PSFCenterX = [512 512 512 1024 1024 1024 1024]; % Center data point in the PSF. Ro
PSFCenterY = [513 513 513 1025 1025 1025 1025];
PSFDataWidth = [101.279 100.455 99.387 138.661 137.311 135.408 132.020 ].*10^-6; %
```

```
% ----Input Validation----
% If the given star position is so far from the center such that no light
% falls on the center, just return zeros.
[~, rhoStar] = cart2pol(starX, starY);
[~, rhoPSFPositions] = cart2pol(PSFPositionX, PSFPositionY);
if rhoStar > max(CMOSsizeX, CMOSsizeY)*CMOSpixelSize + PSFDataWidth(end)
    pixels = cast(zeros(CMOSsizeY, CMOSsizeX), 'single');
    return;
end


% ----Matching and Rotation----
% find the PSF closest to the input star's coordinates
[~, matchingField] = min(abs(rhoPSFPositions - rhoStar));

if matchingField == 1 % Do not do rotation for center PSF because it is symmetrica
    rotatedField = PSFs{matchingField};
else
    % Rotate the PSF according to the input star position
    rotatedField = rotateMatrix(PSFs{matchingField}, PSFCenterX(matchingField), PS
end
PSFPointSize = PSFDataWidth(matchingField) / size(PSFs{matchingField},1); % [m] Th

% PSFposition has the x,y pairs of starting point of the PSF in the FOV
PSFpositionCenter =  [(starX + (1 - PSFCenterX(matchingField))*PSFPointSize) (star

% So far the origin has been at the center of the FOV, now shifting it to the left
%rotatedField = PSFs{matchingField};
% extra pixels as padding around the size of the CMOS to account for stars
% actually positioned outside the sensor's area but still some of their
% light might fall on the sensor.
extraPixels = 2*ceil(PSFDataWidth(end)/CMOSpixelSize);
PSFposition(1) = PSFpositionCenter(1) + (CMOSpixelSize*CMOSsizeX/2) + CMOSpixelSiz
PSFposition(2) = PSFpositionCenter(2) + (CMOSpixelSize*CMOSsizeY/2) + CMOSpixelSiz

paddedPixels = single(zeros(CMOSsizeY+extraPixels, CMOSsizeX+extraPixels));
for i = 1:size(PSFs{matchingField},2)
    positionX = PSFposition(1) + PSFPointSize*(i-1);
    currentPixelX = floor(positionX/CMOSpixelSize);
    for j = 1:size(PSFs{matchingField},1)
        positionY = PSFposition(2) + PSFPointSize*(j-1);
        currentPixelY = floor(positionY/CMOSpixelSize);
        paddedPixels(currentPixelY,currentPixelX) = paddedPixels(currentPixelY,cur
    end
```

```
end
pixels = paddedPixels(extraPixels/2:(extraPixels/2+CMOSsizeY-1), extraPixels/2:(ex

% Normalize output energy
pixels = pixels / (sum(pixels(:))* CMOSpixelSize.^2);
```

randPoisson.m

```
function [y] = randPoisson(lambda)
%************************************************************************
%
% Function description:
% Generate random numbers accoding to poisson distribution.
%
% Inputs:
% - lambda:     mean of the distribution
% - matrixSize: 1x2 vector, size of the matrix to be returned
%
% Reference: Wikipedia
% algorithm poisson random number (Knuth):
%     init:
%          Let L ? e??, k ? 0 and p ? 1.
%     do:
%          k ? k + 1.
%          Generate uniform random number u in [0,1] and let p ? p   u.
%     while p ? L.
%     return k ? 1.


if lambda <0
    error('Lambda must be positif');
elseif lambda == 0
    y = 0;
elseif lambda < 100 % use simple poisson algorithm
    L = exp(-lambda);

    k = 0;
    p=1;

    while p>=L
        k=k+1;
        p=p*rand(1);
    end

    y=k-1;

else % Use gaussian distribution approximation for big enough lambda
    y = round(randn(1) * sqrt(lambda) +lambda);
```

```
    index = y<0; % supress negative values if some are present (very unlikely)
    y(index) = 0;
end



% Matrix version of the function:
%**************************************************************************
% function [y] = randPoisson(lambda,matrixSize)
% if lambda <0
%     error('Lambda must be positif');
% elseif lambda == 0
%     y = zeros(matrixSize);
% elseif lambda < 100 % use simple poisson algorithm
%     L = exp(-lambda);
%
%     y = zeros(matrixSize);
%
%     for m = 1:matrixSize(1)
%         for n = 1:matrixSize(2)
%             k = 0;
%             p=1;
%
%             while p>=L
%                 k=k+1;
%                 p=p*rand(1);
%             end
%
%             y(m,n)=k-1;
%
%         end
%     end
%
% else % Use gaussian distribution approximation for  for big enough lambda
%     y = round(randn(matrixSize) * sqrt(lambda) +lambda);
%
%     index = y<0; % supress negative values if some are present (very unlikely)
%     y(index) = 0;
% end
```

randPoissonImage.m

```
function [y] = randPoissonImage(lambda)
%************************************************************************
% Function description:
% Generate random numbers accoding to poisson distribution for different
% lambda values for every pixel.
%
% Inputs:
% - lambda:     mxn matrix, mean of the distribution
%
% Reference: Wikipedia
% algorithm poisson random number (Knuth):
%     init:
%          Let L ? e??, k ? 0 and p ? 1.
%     do:
%          k ? k + 1.
%          Generate uniform random number u in [0,1] and let p ? p   u.
%     while p ? L.
%     return k ? 1.

[m n] = size(lambda);
lambda = lambda(:);

y = zeros(size(lambda));
indexes = find(lambda<50 & lambda >0); % Cases where pixel intensities are less th
if (~isempty(indexes))
    g = exp(-lambda(indexes));
    em = -ones(size(g));
    t = ones(size(g));
    indexes2 = (1:length(indexes))';
    while ~isempty(indexes2)
        em(indexes2) = em(indexes2) + 1;
        t(indexes2) = t(indexes2) .* rand(size(indexes2));
        indexes2 = indexes2(t(indexes2) > g(indexes2));
    end
    y(indexes) = em;
end

% Use gaussian distribution approximation for big enough lambda
indexes = find(lambda >= 50);
if (~isempty(indexes))
    y(indexes) = round(lambda(indexes) + sqrt(lambda(indexes)) .* randn(size(index
```

```
end

y = reshape(y,[m n]);
```

readSensorNoise.m

```
function noiseArray = readSensorNoise(integration_time, Temperature)
%************************************************************************
%
% Function description:
% Read noise produced in the e2v detector from dark images. When there is
% no experimental data available, dark current is interpolated using
% griddata function
%
% Inputs:
%
%   - integration_time  : [ms] camera integration time
%   - Temperature       : [C] Sensor temperature

%
% Outputs:
%   - noise           : dark current in the image sensor [DN]
%************************************************************************

integration_time = integration_time*1e3; %[ms]

%%check input parameters
if (Temperature<-7.9 || Temperature>42.7)
    error('No experimental data available for that temperature')
elseif (integration_time<10 || integration_time>100)
    error('No experimental data available for that integration time. Integration t
end

%% read all dark images and process noise std. dev. and mean. Results are
%% saved in dark_images_data.mat
% Temps = [-7.9 3 12.8 22.6 32.2 42.7]; %[deg]
% Tints = [10 20 30 40 50 60 70 80 90 100]; %[ms]
% vecTemp = zeros(length(Temps)*length(Tints),1); %[deg]
% vecTint = zeros(length(Temps)*length(Tints),1); %[ms]
% noise=zeros(640-2,844-2,length(Temps)*length(Tints));
% noise3=zeros(640,844,3);
% Nstd=zeros(length(Temps)*length(Tints),1);
% Nmean=zeros(length(Temps)*length(Tints),1);
% extension = 'bmp';
% dir = strcat(pwd,'\NoiseImages\');
% j=0;
%
```

```
% for indexTemp = Temps
%     for indexTint = Tints
%         j = j+1;
%         if (indexTemp>0)
%             file = sprintf('dark_+%04.1fdeg_Tint%03.0fms.%s', indexTemp, indexTi
%             file = strcat(dir, file)
%         else
%             file = sprintf('dark_%05.1fdeg_Tint%03.0fms.%s', indexTemp, indexTin
%             file = strcat(dir, file)
%         end
%         noise3=imread(file);
%         noise(:,:,j) = double(noise3(2:end-1,2:end-1,1));
%
%         %temporal variable
%         noiseTemp = noise(:,:,j);
%
%         % Noise computation
%         Nstd(j) = std(noiseTemp(:)); % [LSB] 3sigma value, Noise level
%         Nmean(j)= mean(noiseTemp(:)); % [LSB] 3sigma value, Noise level
%         %one row for each temperature, one colum for each integration time
%
%         vecTemp(j) = indexTemp;
%         vecTint(j) = indexTint;
%     end
% end

%% read the data stored in .mat file
%it loads variables: Nmean, Nstd, Temps, Tints, vecTemp, vecTint
load dark_images_data.mat

%preallocation
%noiseArray=zeros(640,844);

%% compute the noise for each pixel
%noiseArray = griddata(vecTemp,vecTint,noise(i,j,:),Temperature,integration_time);
noiseSTD = griddata(vecTemp,vecTint,Nstd,Temperature,integration_time);
noiseMean = griddata(vecTemp,vecTint,Nmean,Temperature,integration_time);

%Generate values from a normal distribution with mean noiseMean and
%standard deviation noiseSTD:
noiseArray = noiseMean + noiseSTD.*randn(640, 844);
```

SigmaFilter.c

```
#include <math.h>
#include "mex.h"

double mean(double *data, unsigned int size) {
    double total=0;
    unsigned int i=0;

    while (i<size) {
        total+= data[i++];
    }
    return (total/size);
}

double standardDeviation(double *data, double average, unsigned int size) {
    double variance=0, diff;
    unsigned int i=0;

    while (i<size) {
        diff = data[i++] - average;
        variance += diff*diff;
    }
    variance /= (size-1);
    return (sqrt(variance));
}


double sigmaKernel(double *kernel, unsigned int kernelSize, double sigma, double m
    unsigned int i=0;
    double val, stDev, average, sigmaRange, sigmaTop, sigmaBottom, total=0, count=

    average = mean(kernel, kernelSize);
    stDev = standardDeviation(kernel, average, kernelSize);
    sigmaRange = sigma * stDev;
    sigmaTop = average + sigmaRange;
    sigmaBottom = average - sigmaRange;
    while (i<kernelSize) {
        val = kernel[i];
        if (val>=sigmaBottom && val<=sigmaTop) {
            total += val;
            count++;
        }
```

```
        i++;
    }

    if (count >= minPixels) {
        return (total/count);
    }
    else {
        return (average);
    }
}

void sigmaFilter(double *imageIn, double *imageOut, double *kernel, unsigned int h
    int border = kernelWidth/2, kernelSize = kernelWidth*kernelWidth;
    unsigned int minPixels = (unsigned int)(kernelSize * minPixFraction);
    int i, j, k, ki, kj, tt=0;
    for (i=0; i<width; i++) {
        for (j=0; j<height; j++) {
            k = 0;
            tt++;
            for (ki=(i-border); ki<=(i+border); ki++) {
                for (kj=(j-border); kj<=(j+border); kj++) {
                    kernel[k++] = (ki<0 || kj<0 || ki>=width || kj>=height) ? 0 :
                }
            }
            imageOut[i*height + j] = sigmaKernel(kernel, kernelSize, sigma, minPix
        }
    }
}

void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {
    double *imageIn, *imageOut, *kernel, minPixFraction, sigma;
    unsigned int height, width, kernelWidth;
    width = mxGetN(prhs[0]);
    height = mxGetM(prhs[0]);
    plhs[0] = mxCreateDoubleMatrix(height, width, mxREAL);
    imageOut = mxGetPr(plhs[0]);
    imageIn = mxGetPr(prhs[0]);
    kernelWidth = (unsigned int)mxGetScalar(prhs[1]);
    sigma = mxGetScalar(prhs[2]);
    minPixFraction = mxGetScalar(prhs[3]);

    kernel = mxMalloc(kernelWidth*kernelWidth*sizeof(double));
    sigmaFilter(imageIn, imageOut, kernel, height, width, kernelWidth, sigma, minP
    mxFree(kernel);
```

```
}
```

SigmaFilterMatlab.m

```
% Applies the Sigma filter described in the paper Digital Image Smoothing
% and the Sigma Filter - Jong-Sen Lee.
% Inputs:
%  imageIn - The input image
%  kernelSize - Size of the square area where the sigma filter is applied,
%               also called a kernel or window
%  sigma - number of multiples of standard deviation within one kernel to
%          accept (above and below the average). If a pixel in this
%          window lies outside this range, it is not considered.
%  minPixFraction - fraction of pixels that must lie within the sigma
%                   range. If not enough pixels lie within this range,
%                   a simple average is taken.
% Note: One possible enhancement over this is to assign weights to
% neighbors like a gaussian filter.
function [imageOut] = SigmaFilterMatlab(imageIn, kernelSize, sigma, minPixFraction
[imgHeight imgWidth] = size(imageIn);
border = floor(kernelSize/2);
img = zeros(imgHeight+border*2, imgWidth+border*2);
img(border+1:imgHeight+border, border+1:imgWidth+border) = imageIn;

% We can either propagate the edges to the extra border or leave the border
% to zeros. In this case, the edges largely contain noise, so the borders
% are zeros.
% for y = 1:border
%     img(y,:) = img(border+1,:);
%     img(y+imgHeight+border,:) = img(imgHeight+border,:);
% end
% for x = 1:border
%     img(:,x) = img(:,border+1);
%     img(:,x+imgWidth+border) = img(:,imgWidth+border);
% end

minPixels = floor(minPixFraction*kernelSize*kernelSize);
imageOut = zeros(imgHeight, imgWidth);
for x = 1:imgWidth
   for y = 1:imgHeight
       imageOut(y,x) = SigmaWindow(img(y:y+kernelSize-1, x:x+kernelSize-1), sigma,
   end
end

% Applies one window of the Sigma filter described in the paper Digital
```

```
% Image Smoothing and the Sigma Filter - Jong-Sen Lee.
% Inputs:
%  kernel - The input kernel
%  sigma - number of multiples of standard deviation within one kernel to
%          accept (above and below the average). If a pixel in this
%          window lies outside this range, it is not considered.
%  minPixels - number of pixels that must lie within the sigma range.
%              If not enough pixels lie within this range, the output pixel
%              is the average of all the pixels in the kernel.
function [val] = SigmaWindow(kernel, sigma, minPixels)
stDev = std(kernel(:));
average = mean(kernel(:));
sigmaRange = sigma * stDev;
sigmaTop = average + sigmaRange;
sigmaBottom = average - sigmaRange;
total = 0;
count = 0;
for x = 1:size(kernel,2)
    for y = 1:size(kernel,1)
        val = kernel(y,x);
        if val>=sigmaBottom && val<=sigmaTop
            total = total + val;
            count = count + 1;
        end
    end
end

if count >= minPixels
    val = total / count;
else
    val = average;
end
```

rotateMatrix.m

```
function [matOut] = rotateMatrix(matIn, centerInX, centerInY, srcX, srcY, destX, d

[thetaSrc, rhoSrc] = cart2pol(srcX, srcY);
[thetaDest, rhoDest] = cart2pol(destX, destY);
angle = thetaSrc - thetaDest;

% rotation matrix
mRot = [cos(angle) sin(angle); -sin(angle) cos(angle)];

% Compute range of coordinates
[dimY, dimX] = size(matIn);
minX = -centerInX;
maxX = dimX - centerInX - 1;
minY = -centerInY;
maxY = dimY - centerInY - 1;

% if mod(dimX,2) == 0
%     minX = -dimX/2;
%     maxX = dimX/2-1;
% else
%     minX = -idivide(dimX, 2,'floor');
%     maxX = idivide(dimX, 2,'ceil');
% end
% if mod(dimY,2) == 0
%     minY = -dimY/2;
%     maxY = dimY/2-1;
% else
%     minY = -idivide(dimY, 2,'floor');
%     maxY = idivide(dimY, 2,'ceil');
% end

% Create of grid with the computed coordinates
[X, Y] = meshgrid(minX:maxX, minY:maxY);

% rotate co-ordinates
vRot = [X(:) Y(:)]*mRot;

XI = reshape(vRot(:,1), [dimX, dimY]);
YI = reshape(vRot(:,2), [dimX, dimY]);

% check rotation co-ordinates
```

```
%plot(X, Y, '.', XI, YI, 'x')

matOut = interp2(X, Y, matIn, XI, YI);
matOut(isnan(matOut)) = 0;
```

test_GetStarFromSensorModel.m

```
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% clear all
% %% test generateImageSensorOutput function
% star_mxy = {[5 200 300 ]...
%             [4 400 300]...
%             [3 600 300]}; %...
% %             [4 20.5 20 ],...
% %             [4 20.5 20]...
% %            };
% camSize_xy=[844 640];
% PSF_width = [6 6];
% PSF_shape = 'SimData';
% integration_time = 100;
% Temperature = 0;
% makeplot=0;
%
% [img_bit, noise_bit] = generateImageSensorOutput(star_mxy, camSize_xy,...
%     PSF_width, PSF_shape,integration_time, Temperature,makeplot);
% figure, imagesc(img_bit)
% figure, imagesc(noise_bit)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% test GetStarFromSensorModel function
clear all
star_mxy = {[4 0 0]...
            [3 -200 0]...
            [5 200 0]...
            [6 0 200]};
integration_time = 100e-3; %[s]
Temperature = 0;
[est_star_mxy] = GetStarFromSensorModel(star_mxy,...
                            integration_time, Temperature);
```

detectStars.m

```
%*************************************************************************
% Created by:           Emilio Fraile - EF
% Date:                 March 2012
% Description:          correlates input image to a gaussian surface and
%                       returns the position of the maximums (cadidate
%                       stars)
% Inputs:               imgIn - filtered sky image w/o noise [DN]
%                       windowSize - window size for the gaussian surface
%                       [pixels]
% Outputs:              starPositions - [x y] positions of the stars on the
%                       sensor
%                       numberDetectedStars - number of detected stars
%                       correlation - correlation matrix of the gaussian
%                       window throughout the sky image
% Modified:
%*************************************************************************

function [starPositionsTemp, numberDetectedStarsTemp, correlation] =...
                                    detectStars(imgIn, windowSize)

%% initialize and allocate output parameters
starPositions = [];
numberDetectedStars = 0;
correlation = zeros(FNConstants.CMOSsizeY,FNConstants.CMOSsizeX);
starPositionsTemp = [];

%% create gaussian surface
gsize=[windowSize windowSize];
sigmax = windowSize/3;
sigmay = windowSize/3;
theta = 0; %inclination
offset = 0;
factor = 1;
center = [0 0];
gaussSurf = gauss2D(gsize, sigmax, sigmay, theta, offset, factor, center);

%% correlation of the gaussian window and the imageIn
numberDetectedStarsTemp = 0;

%correlation loop
for row=1:FNConstants.CMOSsizeY-windowSize-1 % r is row in the image matrix
```

```
    for col=1:FNConstants.CMOSsizeX-windowSize-1 % y is column in the image matrix
        %crop the image to the size of the window
        imgInCrop = imgIn(row:row+windowSize-1, col:col+windowSize-1);
        if max(max(imgInCrop))>0 %if it's not only zeros
            %compute correlation
            correlation(row+round(windowSize/2),col+round(windowSize/2)) = corr2(i
            if (max(max(correlation(row+round(windowSize/2),col+round(windowSize/2
                numberDetectedStarsTemp = numberDetectedStarsTemp + 1;
                starPositionsTemp(numberDetectedStarsTemp,:) = [row col]; %store p
            end
        end
    end
end

%for debugging purposes
% figure,imagesc(correlation), title('Correlation Matrix from detectStars')

if ~isempty(starPositionsTemp)
    %current star coordinates are the first window pixel. Convert them to the
    %center of the window
    starPositionsTemp(:,1) = starPositionsTemp(:,1) + round(windowSize/2);
    starPositionsTemp(:,2) = starPositionsTemp(:,2) + round(windowSize/2);
else
    warning('detectStars: No stars detected')
end
```

Bibliography

[1] Acuna and Mario H. Space-based magnetometers. *IEEE Review of Scientific Instruments*, Vol 73(Issue 11), 2002. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4998945.

[2] U. San Bitan Y. Winetraub and A.B. Heller. Attitude determination - advanced sun sensors for pico-satellites. *Handasaim School, Tel-Aviv University, Israel.* URL http://www.agi.com/downloads/corporate/partners/edu/advancedSunSensorProject.pdf.

[3] A. et al. Trebi-Ollennu. Design and analysis of a sun sensor for planetary rover absolute heading detection. *IEEE Robotics and Automation*, Vol 17(Issue 6), 2001. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=976028.

[4] Kara M. Huffman. Designing star trackers to meet micro-satellite requirements. *Massachusetts Institute of Technology*, May 2006. URL http://dspace.mit.edu/handle/1721.1/36170.

[5] Albert Theuwissen et al. Johan Leijtens. Active pixel sensors: the sensor of choice for future space applications. *Proceedings of the SPIE*, Vol. 6744, pp. 67440V, 2007.

[6] Ali Mohammadzadeh Gordon R. Hopkinson and Reno Harboe-Sorensen. Radiation effects on a radiation tolerant CMOS active pixel sensor. *IEEE Transactions on Nuclear Science*, Vol. 51 Issue 5, Oct 2004.

[7] Shy Hamami et al. CMOS image sensor employing 3.3 v 12 bit 6.3 ms/s pipelined adc. *19th International Conference on Micro Electro Mechanical Systems*, 2006.

[8] Nasa program and project management processes and requirements. March 2007. URL http://nodis3.gsfc.nasa.gov/displayDir.cfm?Internal_ID=N_PR_7120_005D_.

[9] K. Marshall. Data verification and validation. *Palms Laboratory*, 2004. URL http://www.epa.gov/region6/qa/qadevtools/mod4references/othertools/29palms_data_verify.pdf.

[10] NASA. Space systems engineering handbook. Verification Module, v1.0.

[11] O. Yadid-Pecht et al. CMOS active pixel sensor star tracker with regional electronic shutter. *IEEE Journal Of Solid-State Circuits*, Vol 32, Feb 1997. URL 2.

[12] W.H. Steyn et al. A high performance star sensor system for full attitude determination on a microsatellite. *Department of Electronic Engineering, University of Stellenbosch.*

[13] European Space Agency. The hipparcos space astrometry mission. consulted Jul 2012. URL http://www.rssd.esa.int/index.php?project=HIPPARCOS.

[14] Encyclopedia of Astronomy and Astrophysics. The tycho-2 catalogue. consulted Jul 2012. URL http://www.astro.ku.dk/~erik/Tycho-2/.

[15] P. et al Gemeiner. Real-time slam with a high-speed CMOS camera. *Image Analysis and Processing. ICIAP 2007*, 14th International Conference, Sep 2007.

[16] K. et al Shinoda. A ccd chip for efp and colour tv camera applications. *Broadcasting Convention*, IBC, Sep 1998.

[17] Ji Soo Lee et al. Analysis of CMOS photodiodes, i. quantum efficiency. *IEEE Transactions on Electron Devices*, Vol 50(Issue 5):p. 1233–1238, May 2003.

[18] G. Prigozhin et al. Ccd charge injection structure at very small signal levels. *IEEE Transactions On Electron Devices*, Vol 55(Issue 8), Aug 2008.

[19] D. Ditwiller. CMOS vs ccd: Maturing technologies, maturing markets. *Photonics Spectra, Lauren Publishing*, 2005.

[20] Centre Suisse d'électronique et de microtechnique. Fill factor enhancement for cmos and ccd sensors. URL http://www.suss-microoptics.com/downloads/Fill_Factor_Enhancement_for_CMOS-CCD-Sensor.pdf.

[21] E.C. et al Teixeira. High fill factor CMOS aps sensor with extended output range. *IEEE Electronics Letters*, Vol 46(Issue 25):p. 1658–1659, Dec 2010.

[22] B. Hooberman M. Battaglia. Back-thinning of CMOS monolithic active pixel sensors. *SNIC Symposium, Stanford, California*, Apr 2006.

[23] Kenneth J. Mighell. Algorithms for ccd stellar photometry. *Astronomical Data Analysis Software and Systems VIII*, ASP Conference Series, National Optical Astronomy Observatories, Vol 172, 1999.

[24] Shinhak Lee. Pointing accuracy improvement using model-based noise reduction method. *Jet Propulsion Laboratory, California Institute of Technology.*

[25] Kara M. Huffman. Designing star trackers to meet micro-satellite requirements. *Massachusetts Institute of Technology*, May 2006.

[26] Hui Tian. Noise analysis in CMOS image sensors. *Stanford University*, Aug 2003.

[27] Domenico Accardo Giancarlo Rufino. Enhancement of the centroiding algorithm for star tracker measure refinement. *Acta Astronautica 53*, Department of Space Science and Engineering, University of Naples, May 2003.

[28] O. Avner, E. ; Yadid-Pecht. Low-power global/rolling shutter image sensors in silicon on sapphire technology. *IEEE International Symposium on Circuits and Systems*, Vol 1:p. 580–583, May 2005.

[29] A. Chiang W. Yang. A full fill-factor ccd imager with integrated signal processors. *Solid-State Circuits Conference 37th ISSCC, IEE International*, 1990.

[30] Wiley J. Larson. The art and science of systems engineering. *NASA*.

[31] Incose handbook of system engineering. consulted in 2012.

[32] Ben Dryer et al. Gamma radiation damage study of 0.18 um process cmos image sensors. Open University and E2V technologies.

[33] M. Battaglia et al. Back-thinning of cmos monolithic active pixel sensors for the ilc at lbn. *SNIC Symposium*, 2006.

[34] J. Vaillant et al. High performance uv anti-reflection coating for backthinned ccd and cmos image sensors. *International Conference on Space Optics*, 2010.

[35] Jong-Sen Lee, Jong-Sen Lee, Jen-Hung Wen, Thomas L. Ainsworth, Kun-Shan Chen, and Abel J. Chen. Improved sigma filter for speckle filtering of sar imagery. *IEEE T. Geoscience and Remote Sensing*, 47(1-2):202–213, 2009.

[36] European Space Agency. The hipparcos space astrometry mission. consulted in June 2012. URL http://www.rssd.esa.int/index.php?project=HIPPARCOS.

[37] Encyclopedia of Science. Hipparcos. *Satellites and Space Probes*, consulted in June 2012. URL http://www.daviddarling.info/encyclopedia/H/Hipparcos.html.

[38] ESCIES. Estec co-60 facility. Consulted in June 2012. URL https://escies.org/webdocument/showArticle?id=251.

[39] ESA. Dose rate to area calculator (for estec co-60). 2008. URL https://escies.org/webdocument/showArticle?id=262&groupid=6.

[40] Karen C. Kajder K. Russell DePriest and Curtis D. Peters. Determination of late-time γ-ray (co60) sensitivity of single diffusion lot 2n2222a transistors. *Sandia National Laboratories*, Consulted in June 2012.

[41] Magali Estribeau Vincent Goiffon and Pierre Magnan. Overview of ionizing radiation effects in image sensors fabricated in a deep-submicrometer cmos imaging technology. *IEEE*, 2009.