Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Department of Electrical, Electronic and Communications Engineering

Bachelor's Degree in
Telecommunications Engineering

BACHELOR'S THESIS

AllSky-Camera system for Monitoring of Optical Satellite Downlinks

Supervisor

Prof. Dr. Miguel A. G. Laso

Student

Iker Aldasoro Marculeta

DLR Deutsches Zentrum für Luft- und Raumfahrt
German Aerospace Center

The thesis work has been carried out at the German Aerospace Center (DLR), in the Institute of Communications and Navigation (IKN) with collaboration of the German Space Operations Center (GSOC), site Oberpfaffenhofen.

DLR Supervisors

Dr. Dirk Giggenbach, IKN (Inst. of Communications and Navigation)

Dr. Marcus T. Knopp, RSC3 (Responsive Space Center)

Academic year 2023/2024

Acknowledgements

First of all, major thanks to my parents and family, for their upbringing and guiding me throughout my journey to adulthood, and for being there for me in the highs and lows. This could not have been possible without you.

I cannot fully express my gratitude to both my supervisors at the German Aerospace Center (DLR), Dr. Dirk Giggenbach and Dr. Marcus T. Knopp, for their guidance and support. I am immensely grateful for the time they dedicated to me, even outside of office hours, and giving the chance to experience this magnificent opportunity. A special thanks to Andrea for all her help with the measurements and for lending me a hand whenever I needed it.

I am also grateful to Dr. Miguel Angel Gomez Laso for his engaging lectures during my studies and his invaluable support and advice during my stay. I am also sorry for the timing of this work, which has not been optimal.

A big thanks to my friends from Pamplona, who helped me stay grounded and motivated during this long period away from home. We do not really value what we have, how close are we, although when being far apart. The longer and farther I am, the more I appreciate it.

Lastly, I would like to thanks all the students from the Institute of Communications and Navigation: Chiara, Salvatore, Davide, Jan, Aurora, Matteo, Thomas, Luka, Raphael, Guillaume, Matti, Uri... Time has passed too quickly for us to realize it. You have all helped me grown up so much as a professional, but most importantly as a person, for which I am deeply thankful. I am sure you all have bright futures ahead of you—you are all wonderful people.

To whom it may concern: thank you, I am sorry, I love you.

Abstract

Satellites widely use the Radio Frequency (RF) band for communications. However, the rapid expansion of Low Earth Orbit (LEO) in the last decade, driven by reduce costs, has led to larger volumes of data to be transmitted from these satellites (the same problem appears in deep space communications). Optical communications offer a solution, providing higher throughputs with reduced equipment volume, lower power consumption, all while avoiding of the regulatory restrictions and tariffs associated with RF. The main problem of space-to-ground links is that the need to face multiple loss-effects primarily due to three main factors: pointing-error losses caused by the satellite's pointing precision; Free-Space Losses (FSL) resulting from the orbit geometry; and atmospheric effects and visibility problems such as atmospheric attenuation, scintillation, and obstruction. This could lead to miss the downlink, being difficult to assess exactly why did that happen.
We hypothesized that a validation tool could be developed as a proof of concept to evaluate the failure point of the operation. Based on an Indium Gallium Arsenide (InGaAs) camera, the system can locate the satellite and estimating its elevation, azimuth and the received intensity at the camera compared to the expected intensity based on the link budget, all without requiring mechanical tracking. This tool is a waterproof enclosure capable of being transported anywhere with any issue. The camera is fitted with a wide-angle lens, providing a 140 degrees field of view, capturing most of the relevant hemisphere. We tested the lens to estimate the Field of View FOV, while the intensity calibration was performed using a method involving a Coarse Wavelength Division Multiplexing (CWDM) and a radio tower equipped with a 1550 *nm* laser. The entire system is controlled by a Python project named "allsky4oleodl," composed of eight different scripts. Our findings indicate that the device successfully detected the satellite on occasions when the Optical Ground Station (OGS) did not, proving the proof-of-concept successful. If further developed and tested, this tool could become a critical standard component of Optical Low Earth Orbit Downlink (OLEODL) systems in the future.

Keywords: AllSky, Azimuth Angle, Beam Spot Size, Contour, CubeSat, CWDM, Deep Space Optical Communications, Elevation Angle, Exposure Time, Filter, GUI, InGaAs, Infrared Camera, Inter-satellite Links, Laser Signal Intensity, Laser Transmitter, Lens, Link Budget, NORSAT-TD, OLEODL – Optical Low Earth Orbit Downlinks, OGS - Optical Ground Station, Optical Satellite Terminal, Optical Space-ground Link, OSIRIS, Threshold.

Contents

Abbreviations

4QD 4-Quadrant Diode. 23

API Application Programming Interface. 29, 35, 38

APK Absolute Pointing Knowledge. 25

ARTEMIS Advanced Relay and Technology MIssion Satellite. 11

CPA Coarse Pointing Assembly. 24, 25

CPF Consolidated Prediction Format. 64

CRL Communications Research Laboratory. 11

Cu Copper. 28

CubeL Laser CubeSat. 23, 24, 63, 64

CubeLCT CubeSat Laser Communication Transmitter. 23

CWDM Coarse Wavelength Division Multiplexing. v, 54

DLR Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center). , 9–11, 16, 21, 22, 57, 68

DM Deformable Mirror. 6, 7

DPC Defect Pixel Correction. 30

DSOC Deep Space Optical Communications. 11

DTE Direct To Earth. 13, 15

EDFA Erbium-Doped Fiber Amplifier. 57

EDRS European Data Relay Satellite System. 11

ESA European Space Agency. 9, 11

ETS Engineering Test Satellite. 11

FEC Forward Error Correction. 6

FLP Flying LaPtop. 21, 22, 50, 63

FOC Fiber Optic Communication. 4

FOR Field Of Regard. 25

FOV Field Of View. v, 1, 21, 27–29, 31, 47, 48, 52, 53

FPA Fine Pointing Assembly. 23, 24

FSL Free-Space Losses. 13, 15

FSM Fine Steering Mirror. 23

FSO Free Space Optical. 25

FSOC Free Space Optical Communications. 1, 3, 4, 10–12, 14

FSOL Free Space Optical Links. 1, 3, 8

FWHM Full Width at Half Maximum. 12, 20, 55

GEO GEostationary Orbit. 8, 10, 11, 14

GOPEX Galileo Optical EXperiment. 11

GSO GeoSynchronous Orbit. 5, 9

GSOC German Space Operations Center. , 9, 58, 61, 64

GUI Graphical User Interface. 35, 49

HAP High-Altitude Platform. 11

IKN Institute of Communications and Navigation. , 9, 21, 51, 55, 57, 58, 61

InGaAs INdium GAllium ArSenide. v, 1, 28, 51, 67

IR InfraRed. 1, 10, 27, 31

IRT Index-of-Refraction Turbulence. 11, 13

ISI Intensity-Scintillation Index. 17

ISS International Space Station. 5

JAXA Japan Aerospace eXploration Agency. 11

KIODO KIrari's Optical Downlink to Oberpfaffenhofen. 9

LADEE Lunar Atmosphere and Dust Environment Explorer. 11

LCRD Laser Communications Relay Demonstration. 11

LCT Laser Communication Terminal. 9

LEO Low Earth Orbit. v, 1, 3–5, 10, 11, 13–15, 27

LLCD Lunar Laser Communication Demonstration. 11

LOS Line Of Sight. 19

LUT Look-Up Tables. 30, 38, 39, 63

MDA Missile Defense Agency. 11

MFD Mode Field Diameter. 55

NASA National Aeronautics and Space Administration. 5, 11, 15

NASDA National Space Development Agency of Japan. 11

NICT National Institute of information and Communications Technology. 11

OGS Optical Ground Station. v, 1, 3, 6, 9, 11, 13, 14, 16–21, 23, 27, 49, 57, 58, 60, 63, 64, 67, 68

OGSOP Optical Ground Station in OberPfaffenhofen. 9, 10

OICETS Optical Inter-orbit Communications Engineering Test Satellite. 11

OLEODL Optical Low Earth Orbit data DownLinks. v, 1, 13, 15–17, 21, 60, 64, 65, 67

OP Oberpfaffenhofen. 16, 57, 58, 60, 63

OSIRIS Optical Space InfraRed downlInk System. 21–25, 50, 63

OSIRIS4C Optical Space InfraRed downlInk System for CubeSat. 23

OSL Optical Satellite Link. 6

PAA Point-Ahead Angle. 7

PAT Pointing, Acquisition and Tracking. 23

PCB Printed Circuit Board. 23

POE Power Over Ethernet. 34

PSI Power Scintillation Index. 17, 21

PTU Pan-Tilt-Unit. 53, 57, 58

QKD Quantum Key Distribution. 11

RF Radio Frequency. v, 3, 10–12, 14, 67

RSC3 Responsive Space Center.

SAR Synthetic Aperture Radar. 11

SILEX Semiconductor Inter satellite Link EXperiment. 11

SMF Single Mode Fiber. 54, 55

SOFA Optical Ground Station Focal Assemblyr. 57

SOTA Small Optical TrAnsponder. 11

SWIR Short-Wavelength InfraRed. 28–31

TAOGS Transportable Adaptive Optics Ground Station. 9

TLE Two-Line Elements. 64

TNO Dutch Organization for Applied Scientific Research. 64

TOGS Transportable Optical Ground Station. 9, 10

VIR Visual and InfraRed. 31

VIS Visible. 28, 31

VSWIR Visible to Short-Wavelength InfraRed. 28, 30

WFS WaveFront Sensor. 6

WOC Wireless Optical Communications. 3

1 Introduction

In recent years, wireless communications have developed at an unprecedented rate. From cellular networks to satellite links, every telecommunication branch has improved significantly. Free Space Optical Communications (FSOC) has emerged as a disruptive technology. Besides other major advantages discussed later, it unlocks the use of new technologies such as quantum communications or adaptive optics, and the use of smaller and lighter antennas, promising to revolutionize satellite communications.

All major space agencies have been testing Free Space Optical Links (FSOL) with Low Earth Orbit (LEO) satellites. One of the main challenges is diagnosing which element of the system is responsible for a failed link acquisition process, as the cause is often unknown. These failures can be due to misalignment, incorrect satellite's orbit data, or faulty equipment in the Optical Ground Station (OGS): The satellite itself could be the reason if its payload is malfunctioning. While visibility and atmospheric conditions significantly impact acquisition success, they are less critical, as a link is attempted only under favourable conditions. Correctly assessing the issue would enhance the understanding of data and lead to improvements in the entire process.

Our hypothesis was that a validation tool could assess the failure point of the entire operation. We propose a monitoring device based on an AllSky-Camera approach to verify the optical signal at the location of the optical ground station. Using an Indium Gallium Arsenide (InGaAs) Infrared (IR) camera operating at the same wavelength as the payload (typically 1550nm) should, if capable of observing most of the hemisphere: track, position and assess the quality of the satellite link throughout the whole acquisition without needing any tracking module. It must also acquire the necessary frames for the posterior analysis while excluding the background light effects. Everything must be built using standard hardware, keeping costs low.

Inspired by cloud coverage systems [1], [2], [3], this work presents a novel approach, offering a compact, portable and affordable solution capable of working as a secondary OGS. Although similar systems have been developed before, they are limited to tracking [4], [5], or narrow fields of view [6], [7]; none can measure the intensity and position of the satellite without a tracking module.

This thesis serves as a Proof of Concept to determine the feasibility of such a device, which has been developed from the ground up through the design, implementation, and evaluation phases. We went through several key phases: requirement analysis of the system sensitivity (link budget), Field Of View (FOV) and exposure times compared to the movement of the objects; component analysis and evaluation; software design and housing the components within a robust, autonomous mechanical setup controlled via Ethernet; testing, verification and analysis of Optical Low Earth Orbit data DownLinks (OLEODL); and documentation of the setup, experiments and their analysis.

The research is structured into five chapters, beginning with this introduction. The second chapter provides a review of the literature, primarily on OLEODL. The third chapter details the materials and methods used, including the system requirements, component selection, experiments and the controlling python software. The fourth chapter presents the results of the experiments. Finally in the fifth summarizes the thesis and offers an outlook for future work.

2 Literature Review

This chapter describes the nature of Free Space Optical Communications (FSOC) and its advantages and disadvantages compared to Radio Frequency (RF) communications. It also explores the use of Free Space Optical Links (FSOL) for Low Earth Orbit (LEO) satellite-to-ground communications, alongside the role of Optical Ground Stations (OGS).

This is an incredibly vast array of topics. Explaining them in its entirety would require a dedicated dissertation, which is beyond the scope of this work. Therefore, we will focus on the key aspects that directly influence this study.

2.1 Free Space Optical Communications

Free Space Optical Communications (FSOC), a branch of the Wireless Optical Communications (WOC), are classified into terrestrial and space systems, as shown by Fig. 2.1.

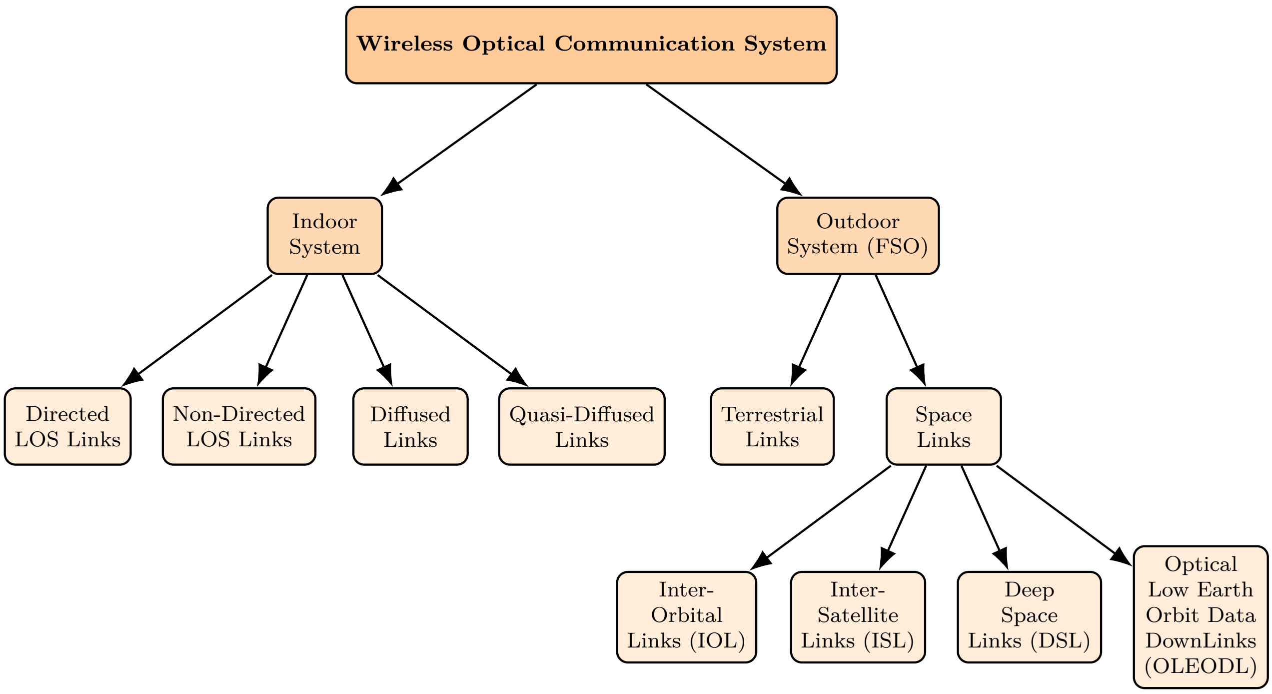


Figure 2.1: Classification of wireless optical communication systems. [8].

The concept of transmitting information through light can be traced back to ancient times, when humans used smoke signals for communication. Alexander Graham Bell, alongside his assistant Charles Sumner Tainter laid the groundwork for modern optical communication patenting the photophone in 1880 [9], predating Guglielmo Marconi's radio communication system [10]. The photophone transmitted sound by modulating light and then extracting the sound signal using materials sensitive to light [11].

Nowadays, a typical free space optical communication system consists of the following components: an electronic data input; a small but powerful light source which can be modulated; emitter optics which shape the emitted beam into a highly directed beam; the atmosphere as the transmission medium; detector optics which receive the transmitted light and focus it onto a photodetector; and an electronic amplifier, as the data output. This setup is illustrated in Fig. 2.2.

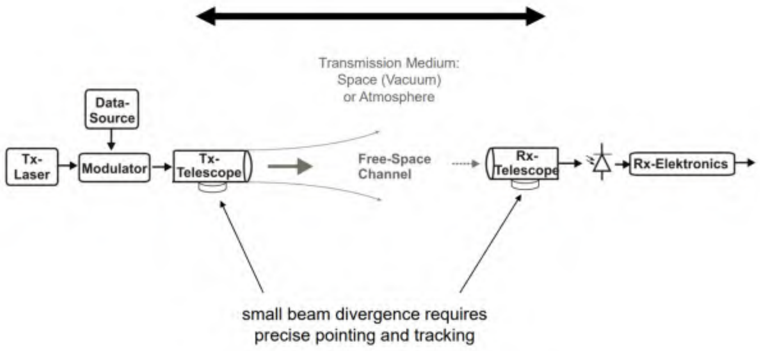


Figure 2.2: Directed Point-to-Point FSOC Link by Modulated Laser Beams. Figure taken from [12].

2.1.1 Free Space Optics vs. Fiber Optics Communications

Both these communications methods rely on a light emitting diode or laser as a point source for data transmission. The main difference is that Fiber Optic Communication (FOC) guides an energy beam through an optical cable, while free space optic communication guides an energy beam through free space. FSOC is particularly useful where physical connections via fiber optical cable are impractical or non-feasible, being a viable solution for "Last Mile Connectivity" [13]. However, it still has limitations.

FSOC greatly suffers from weather conditions, line of sight requirement and range limitation. Weather dependence is particularly troublesome, as attenuation values can reach up to 300 dB/km under adverse conditions [14]. These disadvantages compared with fiber optics, have resulted in its limited adoption.

2.2 Low Earth Orbit Satellites

Almost 8000 active satellites are currently orbiting Earth [15]. This increase is primarily driven by the Low Earth Orbit (LEO) industry, which accounts for close to 90% of the total satellites. Technological advancements have made LEO satellites more cost-effective, easier to launch, and simpler to manage [16].

Low Earth orbit satellites are relatively low in altitude, typically orbiting between 350 and 2000 km above the Earth's surface [17]. They work in interconnected constellations, communicating with ground-based stations to transmit and receive data—enabling various applications such as global communications, Earth observation, and navigation.

Figure 2.3: Different satellite orbits classified by altitude. Figure taken from [18].

2.2.1 History of Low Earth Orbit Satellites

The Earth orbit is currently dominated by Low Earth orbit satellites, but this was not always the case. GeoSynchronous Orbit (GSO) satellites were the preferred method for observing the Earth until recently. The first GSO satellite, Syncom II launched in 1963 [19], was the world's first geosynchronous communications satellite and set the standard for 30 years.

Sputnik I, the first LEO satellite was launched by the Soviet Union in 1957, marking the beginning of the Space Race against the United States [20]. Sadly, low Earth orbit technology stagnated until the early 1990s, with its resurgence by the launch of the IRIDIUM system, a constellation of satellites aimed to provide global communication [21]. However, many LEO missions struggled to meet demands due to their high costs. This would be solved in the 2000s with the development of the CubeSats: a class of nanosatellites using a standard size and form factor. Originally developed in 1999 by California Polytechnic State University and Stanford University [22], The National Aeronautics and Space Administration (NASA) Ames launched its first CubeSat, GeneSat, in December 2006 [23]. The 2010s saw a shift with the rise of commercial companies like SpaceX, and its Starlink project [24]. The increase in LEO satellite deployments has raised concerns about space traffic management and orbital debris, potentially leading into what is known as the "Kessler syndrome", rendering the entire Low Earth Orbit unusable [25], [26].

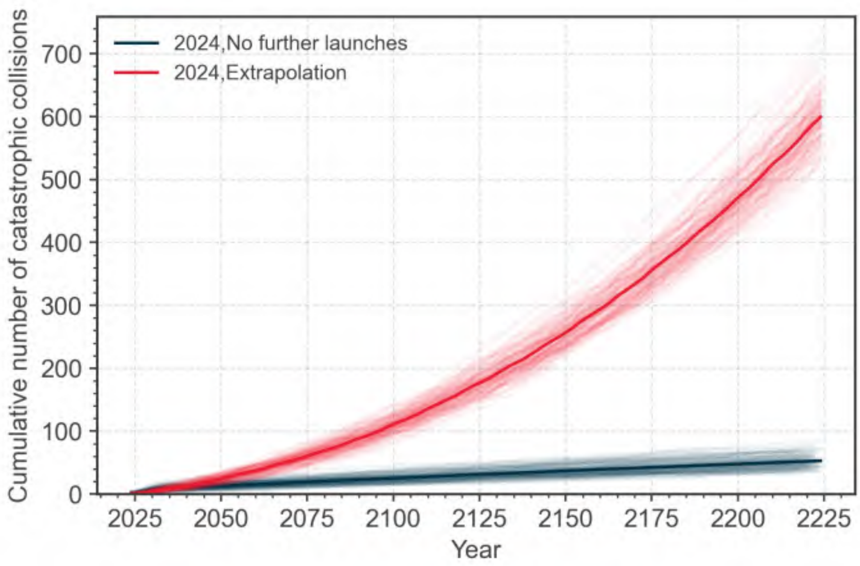


Figure 2.4: The future number catastrophic collisions in Earth orbit. Figure taken from [27].

2.2.2 Advantages and Disadvantages of Low Earth Orbit Satellites

Being closer to the Earth's surface offers different advantages compared to satellites in higher orbits: lower latency and higher bandwidth; enhanced image resolution; faster orbiting speeds (traveling at 7.8 km/s it takes just over 90 minutes to complete an orbit, which means circling around Earth approximately 16 times per day); greater path flexibility, as tilting the orbital plane relative to the Equator allows for more route options; and reduced energy requirements for reaching the final orbit, which is why the International Space Station (ISS)—scheduled for deorbit in 2030 [28]—is positioned at 415 km altitude, enabling quicker and more cost-effective access for spacecrafts.

The main problem of the low Earth orbit is the overcrowding of space debris as the number of launches increases. This problem is aggravated by the fact that LEO satellites, due to their lower altitudes, suffer from a higher rate of atmospheric drag, requiring more power and limiting their lifespan to 7-10 years. Additionally, LEO satellites cannot function effectively alone for communication purposes as they are hard to track, requiring full constellations.

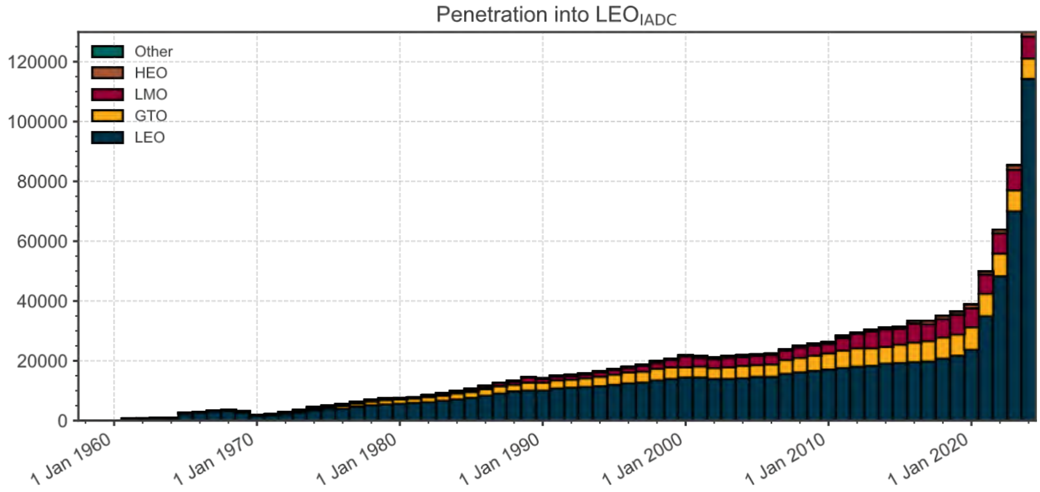


Figure 2.5: Evolution of absolute area residing in or penetrating LEO_{IADC}. Figure taken from [27].

2.3 Optical Ground Stations

To establish an Optical Satellite Link (OSL), an Optical Ground Station (OGS) is required to communicate with the optical terminal. Fig. 2.6 illustrates the main components of the ground segment for a very-high throughput communication system.

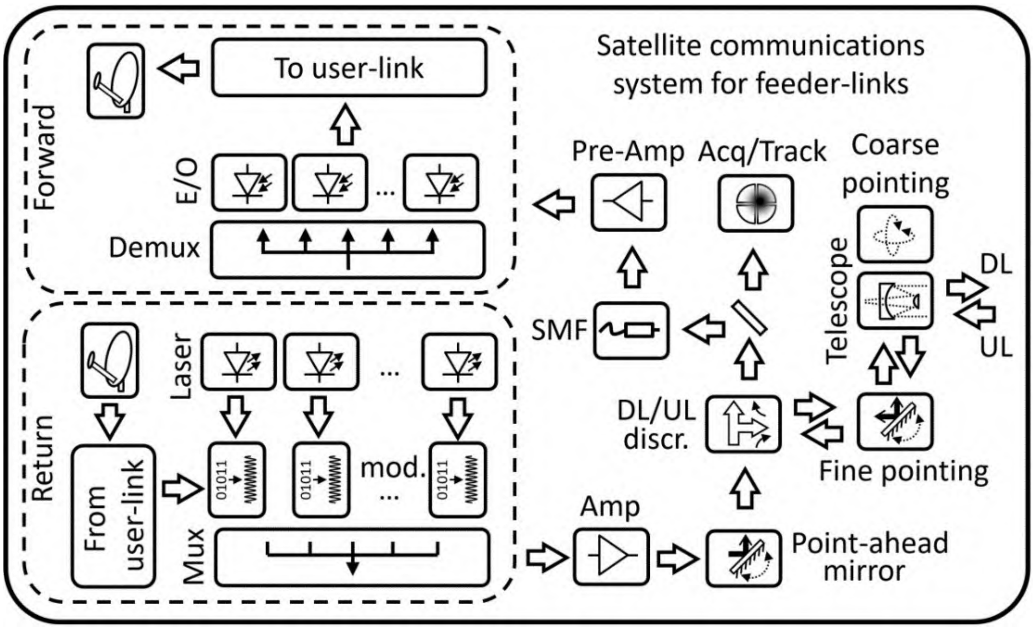


Figure 2.6: Main components of the space terminal of a very-high throughput communications satellite based on optical feeder link. Figure taken from [29].

The telescope transmits and receives data to and from the satellite. The coarse-pointing system points towards the satellite, maintaining a small pointing error to compensate for the signal's angle-of-arrival. The WaveFront Sensor (WFS) and Deformable Mirror (DM), components of the adaptive optics system, compensate phase distortions caused by atmospheric turbulence. Both the adaptive optics and pointing systems are employed in both link directions: in the downlink for fiber coupling and in the uplink for pre-compensation of beam wander and phase distortions.

In the downlink process, the light is coupled into a single-mode fiber, pre-amplified, demultiplexed, and converted to the electrical domain for Forward Error Correction (FEC) and data processing before sending it to the network. Using a large telescope benefits the link budget, as the receiver gain increases with the diameter of the receiver.

For the uplink, data from the network is converted into the optical feeder-link format, modulated onto each laser carrier, multiplexed, amplified, and after compensating for the point-ahead angle, coupled into the telescope system to be transmitted towards the satellite. The transmitter size is constrained by atmospheric turbulence and pointing accuracy, which is limited by the beam wander.

Certain systems are required at ground stations to ensure reliable and stable satellite-to-ground communications. Given their complexity, we will provide only a brief overview of these systems.

2.3.1 Adaptive Optics

To utilize components such as low-noise amplifiers or multiplexers, the light collected by the telescope must be coupled into a single-mode optical fiber. This coupling efficiency is compromised due to wavefront distortions caused by atmospheric turbulence. These phase distortions increase with the telescope's diameter, as larger apertures capture more phase aberrations—amount defined by the ratio between the aperture's diameter and the fried parameter D/r_o. This is a key constraint in increasing telescope diameter [30].

Adaptive optics systems can partially correct for these phase distortions. A typical adaptive optics setup includes a tip-tilt mirror, which compensates for angle-of-arrival fluctuations caused by atmospheric turbulence and tracking errors; a deformable mirror, composed of a set of actuators, which receives the beam and compensates its phase distortion; and a wavefront sensor, which estimates the phase of the received wavefront and computes the signals to drive the DM.

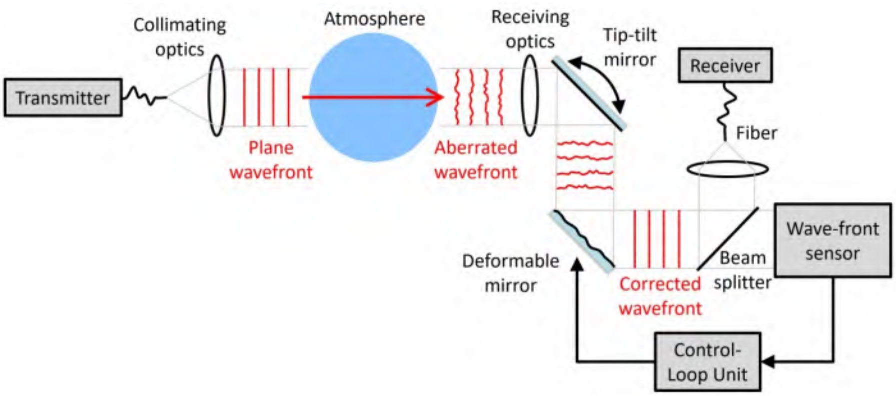


Figure 2.7: Block diagram of an ideal adaptive-optics system. Figure taken from [29].

2.3.2 Point-ahead Angle and References for Uplink Pre-correction

As previously discussed, the tip-tilt mirror is necessary to compensate for angle-of-arrival fluctuations caused by atmospheric turbulence. In the meantime, the finite speed of light delays the uplink signal reaching the satellite, resulting in an angular separation between the uplink and downlink directions. This separation is referred to as the Point-Ahead Angle (PAA), as shown in Fig. 2.8.

This angle Φ can be calculated using equation (2.1), where v_t is the tangential velocity of the satellite and c denotes the speed of light.

$$\Phi = \frac{2v_t}{c} \tag{2.1}$$

The pointing direction of the uplink will fluctuate due to atmospheric turbulence generating intensity fluctuations; this phenomenon is known as beam wander. Ideally, the uplink could be compensated for beam wander using the same measurements from the tip-tilt mirror as a form of pre-compensation. A key challenge is that atmospheric effects are correlated within a certain cone, known as the isoplanatic angle [31] (as illustrated in Fig. 2.8). If the point-ahead angle exceeds the isoplanatic angle, the system will move outside this cone. Exiting the coherent cone leads to increased decorrelation between both paths, resulting in greater beam wander and, ultimately, more significant intensity fluctuations at the satellite.

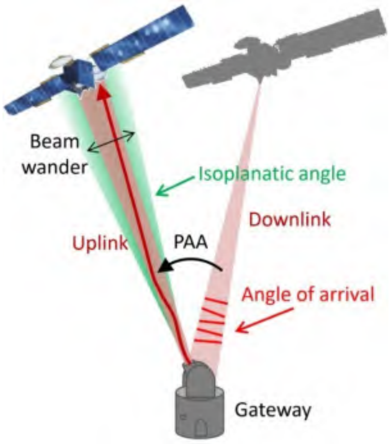


Figure 2.8: Block diagram of an ideal adaptive-optics system. Figure taken from [29].

The most straightforward solution to reduce beam wander is to increase the divergence. The main drawback is the reduction in mean power received at the satellite. Other solutions are possible, such as laser guide stars based on Rayleigh scattering or adaptive optics, exist but they will not be covered in this discussion.

2.3.3 Spatial Diversity for Turbulence Mitigation

Atmospheric turbulence can cause a significant performance degradation in Free Space Optical Links (FSOL). Spatial diversity is applied on both the transmitter and receiver, alongside adaptive optics, to minimize its impact.

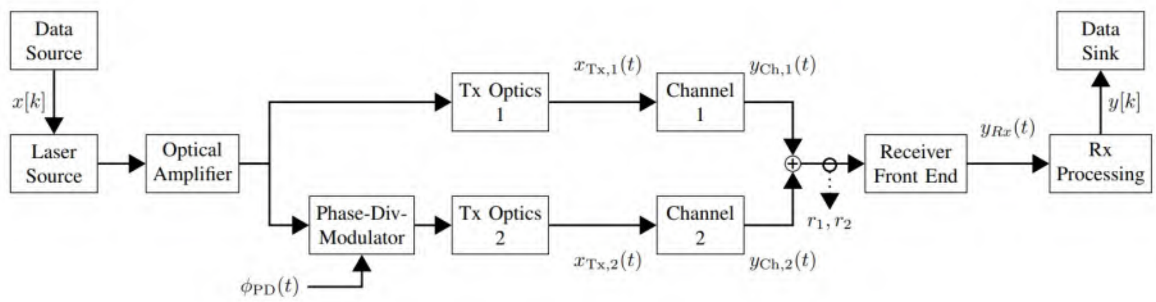


Figure 2.9: Block diagram of transmitter diversity system with Phase-Division in Bit-Time for two transmitters and a single receiver. Figure taken from [32].

At the transmitter, size N uncorrelated beams are transmitted to reduce the scintillation at the receiver side by an equal factor [33], limiting power fluctuation. N independent sources might be used to obtain these N beams, although this can be circumvented using on-off keying data modulation [34]. Each beam is located at a certain distance respect to the others, to ensure that the turbulence crossed by each one is mutually uncorrelated—as the atmospheric paths are assumed to be uncorrelated when they are half a meter apart. The more beams we can use, the lower the scintillation will become; however, this technique is mostly used for Geostationary Orbit (GEO) satellite links. Experimental results can be found in [35], [36].

Data can be modulated on the laser carrier, but careful consideration is required, as partial bandwidth overlap can result in strong interference. Techniques like polarization or wavelength separation can mitigate this, although new methods, such as employing multiple signal sidebands, are currently being researched [37].

At the receiver, diversity can be achieved using an array of telescopes, beneficial in avoiding monolithic mirrors in deep space scenarios while achieving similar performance.

2.3.4 Examples of Optical Ground Stations

There are over 30 OGSs operated by various international organizations worldwide, with more planned for the future. The European Space Agency (ESA)-OGS, located at the *Observatorio del Teide* in Tenerife, Canary Islands (Spain) [38], was built in 2001 to support ground-to-GeoSynchronous Orbit (GSO) satellite links as part of the SILEX project [39].

The Transportable Adaptive Optics Station (TAOGS), developed by the Deutsches Zentrum für Luft- und Raumfahrt (DLR) and Tesat Spacecom, was designed to support Tesat's spaceborne Laser Communication Terminals (LCTs). The key features of this OGS are its adaptive optics system and its portability, with all equipment housed within a container [40].

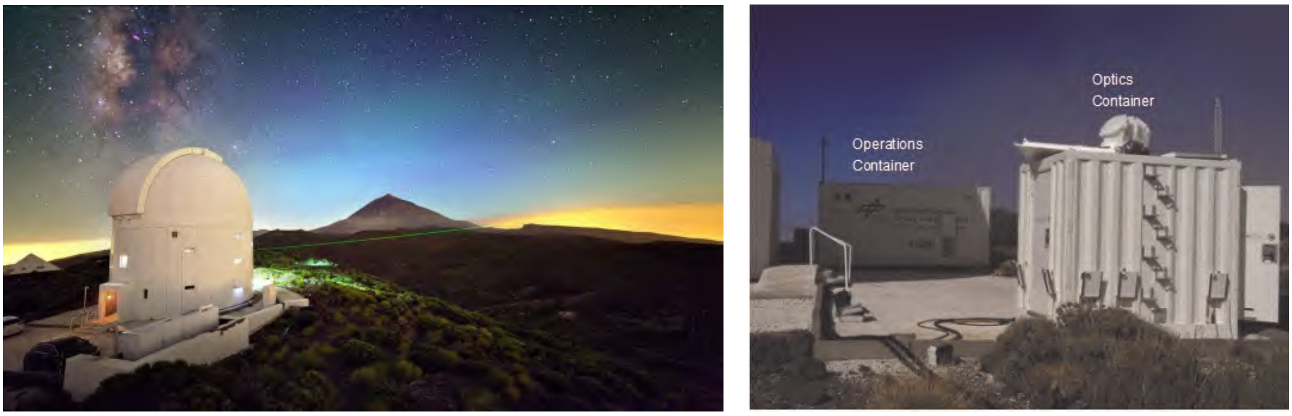


Figure 2.10: ESA Optical Ground Station in Tenerife, Canary Islands (left). Figure taken from [41]. Photograph of the TAOGS (right). Figure taken from [42].

The DLR Optical Ground Station in OberPfaffenhofen (OGSOP), originally implemented for the KIrari's Optical Downlink to Oberpfaffenhofen (KIODO) experiment in 2006 [43], equipped with a 40-cm Ritchey–Chrétien telescope, a similar version of this telescope is now used at the German Space Operations Center (GSOC). In 2021, this telescope was replaced with an 80-cm telescope featuring a Coudé path [44].

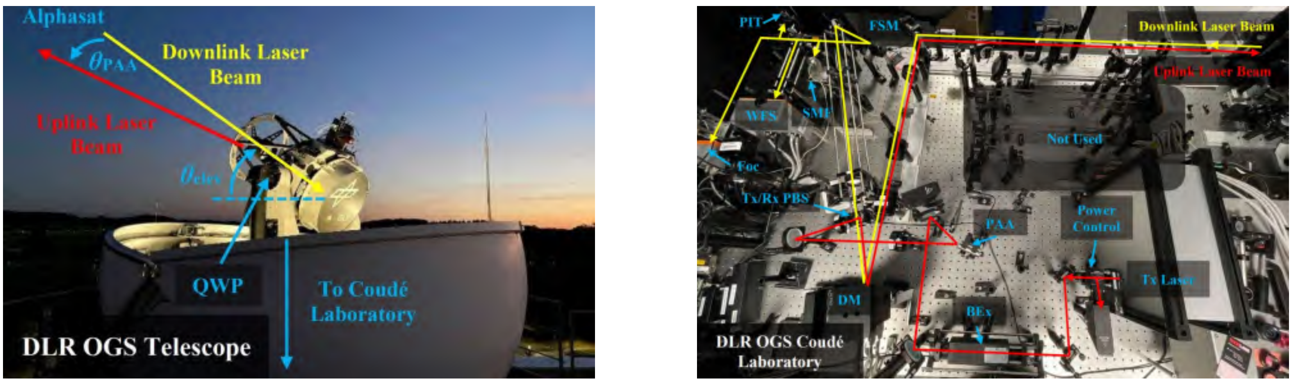


Figure 2.11: DLR Oberpfaffenhofen OGS's 80-cm telescope (left). Pre-distortion Adaptive Optics experimental setup (right). Figure taken from [45].

In addition to the OGSOP, the the Institute of Communications and Navigation (IKN) at DLR developed the Transportable Optical Ground Station (TOGS) in 2010, shown in Fig. 2.12. The TOGS is a versatile and modular OGS designed for experimental optical uplink and downlink scenarios, as well as for measuring the atmospheric optical channel. Equipped with a 60cm Ritchie-Chrétien, it was intended for rapid deployment and determination of position and attitude, as required for alignment with a known target.

Figure 2.12: DLR Transportable Optical Ground Station. Figure taken from DLR Media.

Both the OGSOP and the TOGS transmit and receive through different apertures (The OGSOP can also use the same for transmission and reception). As a result, beam wander cannot be minimized, forcing to increase divergence to ensure that the uplink reaches the satellite most of the time despite the increased beam wander, as shown in equation (2.2). The root-mean-squared value of the beam wander from GEO links are in the order of tens of microradians, however for LEO satellite links decreases to a few microradians, making the impact of beam wander due to turbulence negligible in that case.

$$\sqrt{\sigma^2_{\text{pointing}}} = 0.73 \left(\frac{\sqrt{2}\lambda}{D_T} \right) \left(\frac{D_T}{\sqrt{2} r_0} \right)^{5/6} \tag{2.2}$$

2.4 Optical Low Earth Orbit Data DownLinks

As the amount of data we gather and transmit to Earth increases, we need solutions to accommodate this need. Free Space Optical Communications (FSOC) have found their niche in satellite to ground, inter-satellite and deep space communications, addressing the limitations of Radio Frequency (RF) communications.

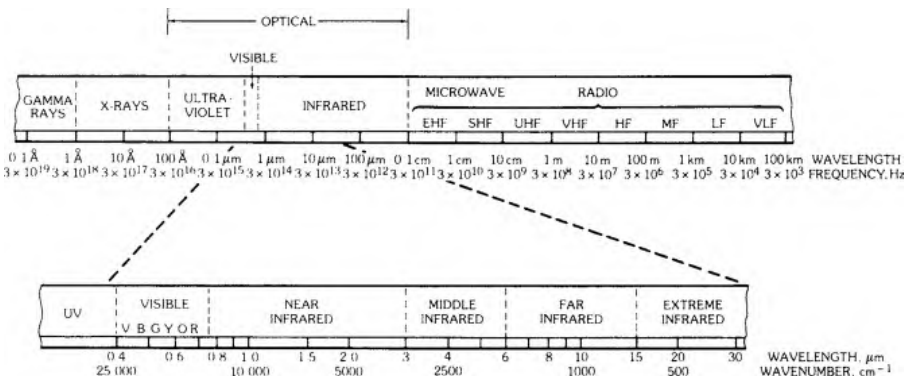


Figure 2.13: The electromagnetic spectrum. Figure taken from [46].

Although compatible [47], FSOC is establishing itself as an alternative over RF in many application scenarios due to its reduced weight and volume of the transmitter and receiver equipment (up to 50% less), lower power consumption (25% less power), and avoidance of the tariffs or regulatory restrictions associated with RF usage. The InfraRed (IR) light packs data into tighter waves, allowing ground stations to receive more data at once. While laser communications do not always provide higher throughput, that is the goal as more data can be transmitted in one downlink, increasing bandwidth by 10 to 100 times compared with radio frequency systems (Data rates up to Terabit/s are possible).

Main challenges such as link blocking by clouds and fog, signal scintillation by Index-of-Refraction Turbulence (IRT) and precise pointing and tracking for the link acquisition, will remain.

2.4.1 Historical overview of Free Space Optics in space communications

Using free space optical communications for space communication is not new. Developed in 1965 by the United States for deep space exploration, it would take until 1975 to complete an inter-satellite communication system. Despite this progress, FSOC development stalled for two decades due to atmospheric effects with optical signals.

In 1992, the National Aeronautics and Space Administration (NASA) performed the Galileo Optical EXperiment (GOPEX)—they emitted megawatts power, 532nm pulses from Earth, detecting them with a camera onboard Galileo probe, up to 6 million km away [48]. In 1994, the Communications Research Laboratory (CRL) carried out the first laser-based ground-to-space downlink communication using the Japanese Engineering Test Satellite VI (ETS-VI), launched by the NAtional Space Development Agency of Japan (NASDA) [49]. The 21st century is witnessing a series of significant milestones: the first inter-satellite link between the French SPOT4 Low Earth Orbit (LEO) satellite and the European Space Agency (ESA) Advanced Relay and TEchnology MISsion (ARTEMIS) GEostationary Orbit satellite (GEO), was achieved by the Semiconductor Inter satellite Link EXperiment (SILEX) experiment in 2001 [50]; the first optical Gbit/s High-Altitude Platform (HAP) to ground downlink by the Deutsches Zentrum für Luft- und Raumfahrt (DLR) in 2005 [51]; the first LEO-to-ground optical communication link—between the "Kirari" Optical Inter-orbit Communications Engineering Test Satellite (OICETS) and the Optical Ground Station (OGS) developed by the National Institute of Information and Communications Technology (NICT)—was performed in collaboration with the Japan Aerospace Exploration Agency (JAXA) in 2006 [52]; the first LEO-to-LEO link between the US Missile Defense Agency (MDA) experimental satellite and TerraSAR-X, a German commercial Synthetic Aperture Radar (SAR) satellite—utilizing a secondary laser communication payload built by Tesat-Spacecom—in 2007 [53]; the first duplex laser communication between a satellite in lunar orbit, the Lunar Atmosphere and Dust Environment Explorer (LADEE), and ground stations on the Earth, performed by NASA as the Lunar Laser Communication Demonstration (LLCD) in 2013 [54]; the first LEO-to-ground optical communications using a Small Optical TrAnsponder (SOTA) and Quantum Key Distribution (QKD) by NICT in 2014 [55]; the first optical communications link through the atmosphere with a throughput over 1 Tbit/s under a realistic turbulence environment by DLR in 2016 [56]; the first long distance quantum-entanglement distribution experiment using the Chinese quantum science experiments LEO satellite, Micius, in 2017 [57]; the first system to provide data relay services to the LEO satellites from GEO orbit by means of optical and RF bands in real-time and at a rate of 1.8 Gbit/s, European Data Relay Satellite System (EDRS), developed, manufactured and tested by OHB System AG in 2019 [58]; the inclusion of the Laser Communications Relay Demonstration (LCRD) in the previously discussed LLCD to prove that optical communications can meet needs for higher data rates, by NASA in 2022 [59] and the first mission using the Deep Space Optical Communications (DSOC) system, PSYCHE, by NASA in 2023 [60].

2.4.2 Pointing, acquisition and tracking

The fundamental parameters involved in the pointing of a lasercom system are the spot size and the divergence: the former estimates how well the system can focus the received laser signal, the latter estimates how narrowly can it transmit a laser beam. Both can be studied through the concept of the diffraction limit, given by the equation (2.3).

$$\frac{I(\theta)}{I(0)} = \left[2\frac{J_1(\frac{\pi D}{\lambda}\sin(\theta))}{\frac{\pi D}{\lambda}\sin(\theta)} \right]^2 \tag{2.3}$$

Here, the angular variation of the intensity of the radiation $\frac{I(\theta)}{I(0)}$ depends on: the aperture's diameter D;

the wavelength λ; and the Bessel function of the first kind of x, $J_1(x)$. By applying the first-minimum criterion and the approximation $\sin(\theta) \approx \theta$, the diffraction limit of a telescope θ can be approximated by equation (2.4). The graph shown in Fig. 2.14 deviates from the practical application, as we can only detect intensity levels down to -10/-15 dB—one will not detect much more than the Full Width at Half Maximum (FWHM) as the sidelobes are too faint.

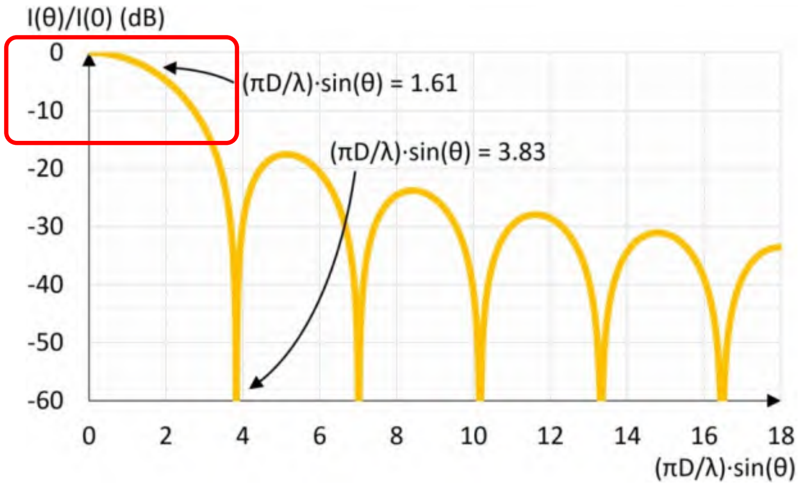


Figure 2.14: Intensity of radiation as a function of the wavelength λ, the aperture diameter D and the angular width θ. The detectable part is marked in red. [29].

$$\theta = 1.22\frac{\lambda}{D} \tag{2.4}$$

From the transmitter perspective, equation (2.4) demonstrates that shorter wavelengths produce narrower, minimal divergence beams: One of the main advantages of FSOC over RF. Low divergence is crucial for long distances as it enhances directivity, but it also demands higher pointing accuracy—in radio frequency communications pointing accuracy is in the order of milliradians, whereas deep-space laser communications need sub-microradian precision.

To keep a stable line, the satellite requires a reference point: celestial bodies in deep space or a laser beacon transmitted from the ground if the satellite is near Earth. As discussed in subsection 2.3.2, the beacon emits a signal with a divergence matching the uncertainty zone where the satellite is predicted to be. The satellite tracks this beacon and starts the downlink at a different wavelength or polarization. The complete process is illustrated in Fig. 2.15.

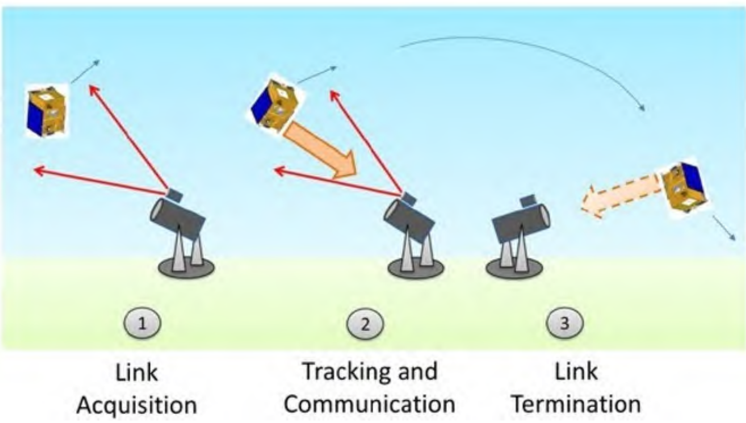


Figure 2.15: Schematic representation of the 3 phases in optical downlink. Figure taken from [61].

2.4.3 Low Earth Orbit-Direct to Earth Geometry

Link duration, range, and angular slew rate (rate at which a satellite changes its orientation, measured in degrees per second) for optical LEO-Direct to Earth (DTE) links are well-established from conventional LEO-satellite studies. A minimum elevation angle of 5° is assumed for the start of signal acquisition, with 10° or higher required for secure data transmission

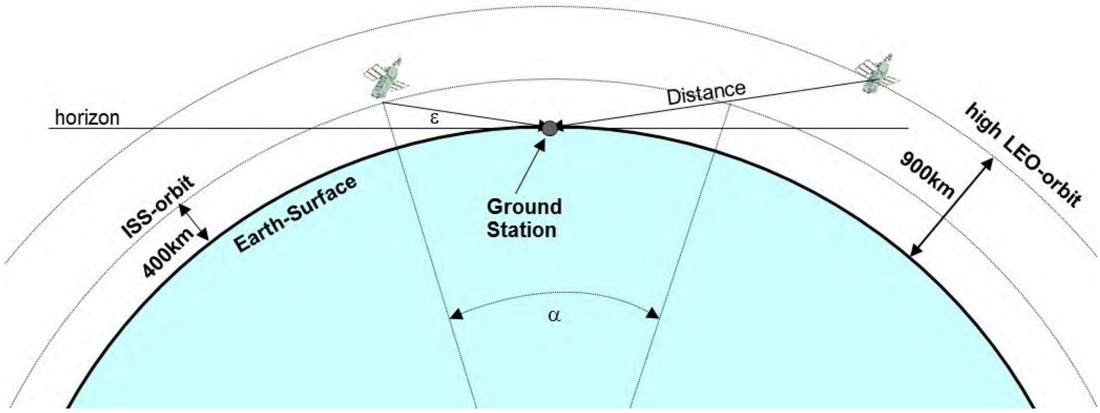


Figure 2.16: Link geometry of typical LEO satellite downlinks with circular orbits. Figure taken from [62].

Low elevations must be carefully considered, as the satellite spends nearly 80% of the time between 0° and 20°, as illustrated in Fig. 2.17.

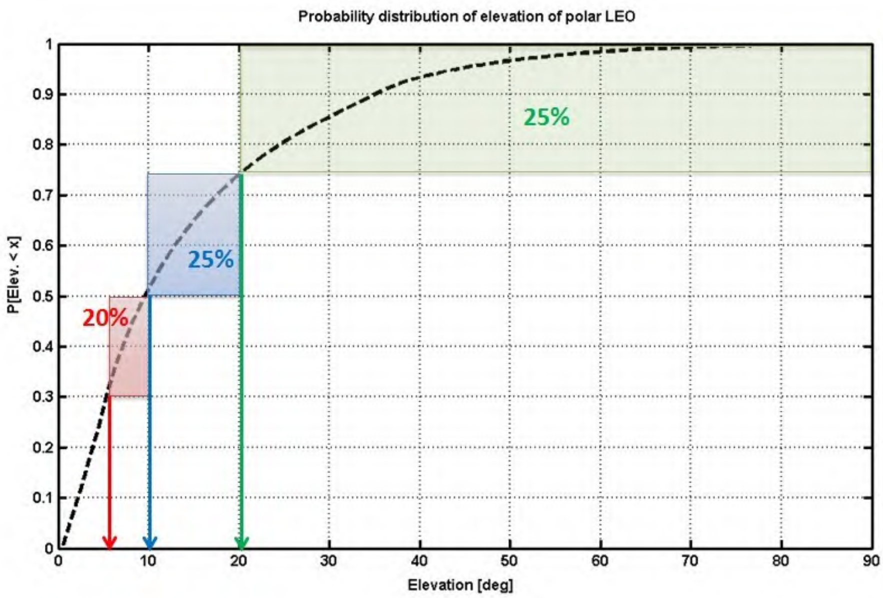


Figure 2.17: Typical distribution of the average viewing elevation for a polar LEO satellite (500 km orbit height). This relative distribution is qualitatively similar for any optical ground station (OGS) location on earth, although of course the absolute overall visibility changes depending on orbit and OGS latitude. Figure taken from [62] [63].

2.4.4 Loss-Effects in Optical Space-Ground Links

The losses in Optical Low Earth Orbit data DownLinks (OLEODL) are primarily due to three main factors: pointing-error losses caused by the satellite's pointing precision; Free-Space Losses (FSL) resulting from the orbit geometry; and atmospheric effects such as atmospheric attenuation, scintillation due to IRT, and obstruction. These factors are illustrated in Fig. 2.18.

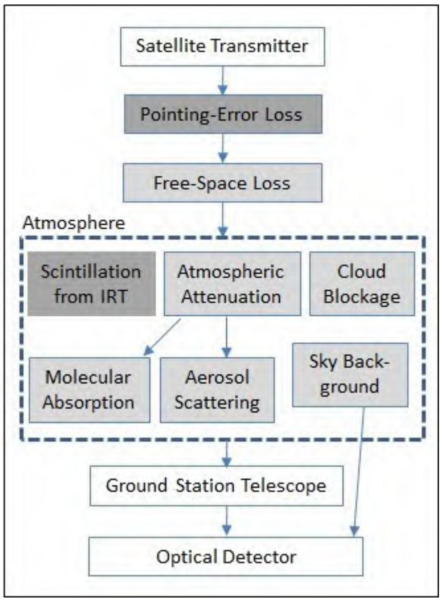


Figure 2.18: Link-parameters affecting the optical downlink quality. Effects in darker boxes change faster during downlink, values in blank boxes are static. Figure taken from [64].

Pointing-error losses are defined as the ratio between the actual received power at the OGS and the ideal received power [65]. It is crucial to address this issue as FSOC links have smaller divergence compared to RF links, risking missing the OGS entirely. This is one of the key aspects which we want to assess in this thesis, identifying how much did the satellite pointing deviated from its ideal value. For GEO satellites, the uplink uses the information of the downlink's angle-of-arrival (tilt component of the phase) to point towards the satellite and pre-correct for pointing fluctuations: This is achievable as both uplink and downlink paths are assumed the same for GEO links. For LEO satellites, the presence of the point-ahead angle causes the uplink and downlink paths to be uncorrelated, making this approach unsuitable.

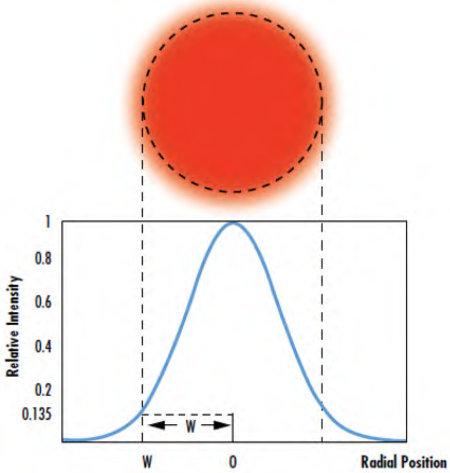


Figure 2.19: The waist of a Gaussian beam is defined as the location where the irradiance is $1/e^2$ (13.5%) of its maximum value. Figure taken from [66].

The impact of the residual pointing jitter can be estimated by normalizing the beam wander variance, $\sigma^2_{\text{pointing}}$ (presented on equation 2.2) by the beam divergence. The beam divergence is defined as the radius of the Gaussian wave where the intensity decays to $1/e^2$, also known as Half Angle Beam Divergence $\vartheta_{beam} = \lambda/\pi w_0$, where w_0 represents the beam waist radius, illustrated in Fig. 2.19. The impact factor in the pointing is defined by equation (2.5), and its relation with the beam waist radius can be appreciated in Fig. 2.20.

$$\beta_{\text{pointing}} = \frac{\sqrt{\sigma^2_{\text{pointing}}}}{\vartheta_{\text{beam}}} \tag{2.5}$$

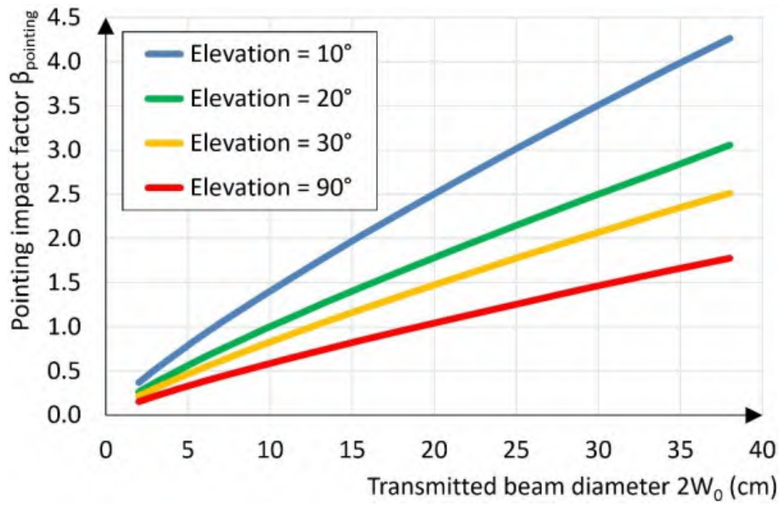


Figure 2.20: Pointing impact factor β_{pointing} vs transmitted beam diameter for several link elevation angles. Figure taken from [29].

Free-space losses decrease with both distance and wavelength, as shown in Fig. 2.21, where the satellite approaches zenith (closer to Earth). Shorter wavelengths should also minimize these losses, but this is not entirely the case, as larger scintillation is generated when passing through the atmosphere. This is one of the reasons—though not the primary one—why 1550 nm is typically used for LEO-DTE links, while 1064 nm is preferred for inter-satellite links (eye safety can be ignored in this scenario). The main reason of the dominance of the 1550 nm wavelenght is the superior availability of components, to the extent that organizations like NASA are beginning to adopt this wavelength even for inter-satellite links.

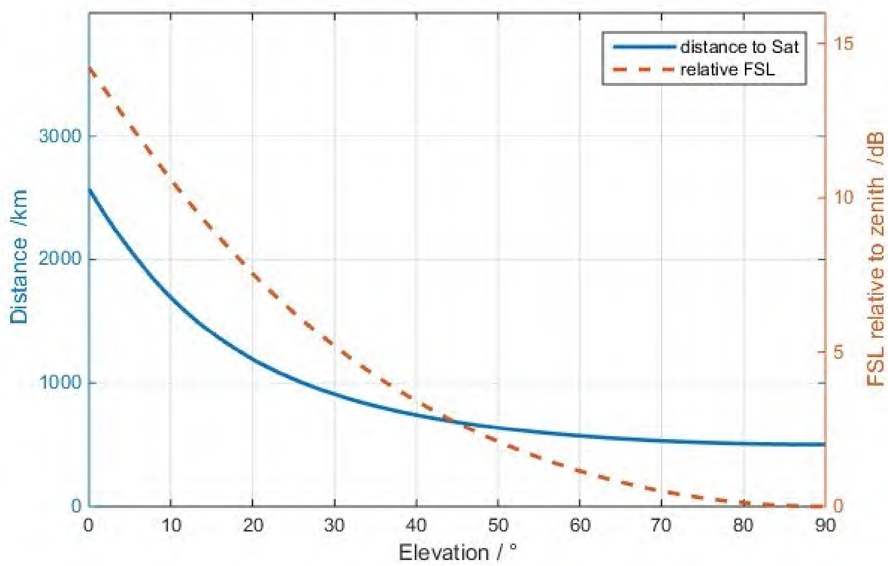


Figure 2.21: Distance and relative FSL for 500km orbit height. Figure taken from [64].

OLEODL systems deal with three major atmospheric effects: absorption, scattering, and scintillation. Absorption occurs when photons in the beam collide with particles suspended in the atmosphere—water vapor, volcanic ash, and aerosols—the latter being particularly problematic. This effect can be minimized by optimizing the location of the optical ground station. The three main communication wavelengths used in OLEODL occupy absorption-free windows, as shown in Fig. 2.22. In these cases, absorption can be considered negligible at zenith, however it still need to be regarded in lower elevation due to the larger presence of molecular and aerosol particles as appreciated in Fig. 2.23 and Fig. 2.24, respectively.

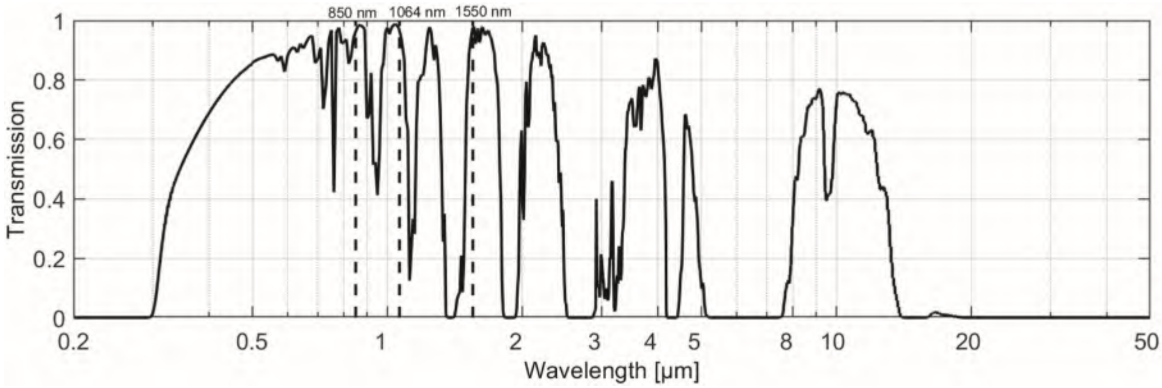


Figure 2.22: Atmospheric (clear-sky) transmission window for absorption only. 850nm, 1064nm and 1550nm windows are shown. Figure taken from [67].

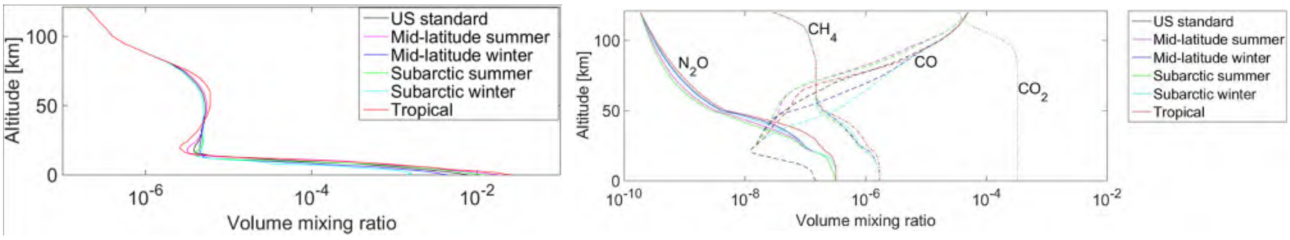


Figure 2.23: Volume mixing ratio of H_2O molecules with respect to altitude, for different atmospheric models (left). Volume mixing ratio of N_2O, CH_4, CO, and CO_2 with respect to altitude for different atmospheric models (right). Figure taken from [67].

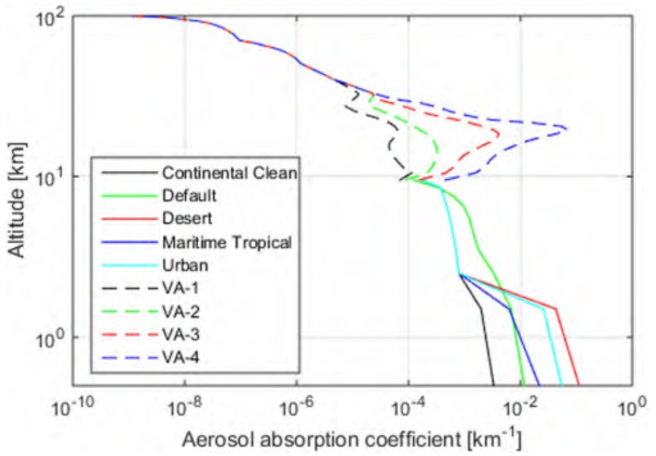


Figure 2.24: Aerosol absorption coefficients for different atmospheric models and volcanic activity (VA) levels (4 being the maximum) at 1550 nm. Figure taken from [67].

Scattering refers to the dispersal of the beam by suspended particles in the atmosphere. Aerosols particles are larger than the wavelength of the incident beam, and variate depending on the height: this known as Mie Scattering. Calculating its losses using simple equations is complex. For the OGS-OP of DLR, a flat-Earth approximation model already exists [64], which confirms that 1550 nm is the best choice. Scattering due to smaller particles than the wavelength, also known as Rayleigh scattering, is negligible for near-infrared or longer wavelengths [68].

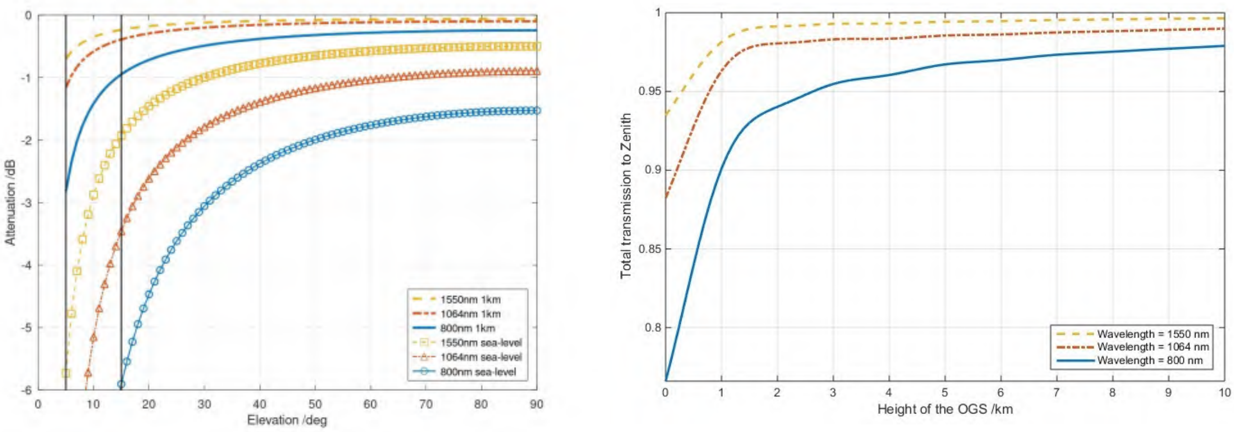


Figure 2.25: Atmos. attenuation over elevation at different wavelengths, air qualities, and OGS-altitudes, using flat-Earth modelling. The minimum (5°) and typical medium (15°) elevation for OLEODL are labeled as vertical lines (left). Zenith transmission vs OGS altitude, acc. to models (right). Figure taken from [64].

Scintillation, defined as the variance of the signal normalized to its squared mean, characterizes the temporal or spatial fluctuations of the received signal, this causes "speckle-patterns" in intensity through self-interference (the cause why we observe stars twinkle). The Power Scintillation Index (PSI) is used to describe this effect on the optical wave at a single point, like a receiver—if we quantify it with the normalized variance of the intensity, we obtain the Intensity-Scintillation Index (ISI), σ_I^2. When the aperture of the receiver increases beyond the correlation length of the intensity fluctuations, the scintillation decreases as a bigger receiver can collect multiple correlation lengths, averaging signal fluctuations. This is so-called aperture averaging.

As discussed in subsection 2.3.4, different apertures are often used for reception and transmission in satellite communications. For the downlink, the receiver aperture is larger than the intensity correlation length, applying aperture averaging and reducing scintillation. For the uplink, the turbulence is closer to the transmitter, causing the intensity correlation length to extend up to several hundred of meters, much larger that the transmitter aperture. The (ISI), for both paths can be estimated as follows (equations 2.6 and 2.7):

$$\sigma^2_{I,downlink} = \exp\left[\frac{0.49\sigma^2_{B_d}}{\left(1+1.11\sigma^{12/5}_{B_d}\right)^{7/6}} + \frac{0.51\sigma^2_{B_d}}{\left(1+0.69\sigma^{12/5}_{B_d}\right)^{5/6}}\right] - 1 \tag{2.6}$$

$$\sigma^2_{I,uplink} = \exp\left[\frac{0.49\sigma^2_{B_u}}{\left(1+0.56\sigma^{12/5}_{B_u}\right)^{7/6}} + \frac{0.51\sigma^2_{B_u}}{\left(1+0.69\sigma^{12/5}_{B_u}\right)^{5/6}}\right] - 1 \tag{2.7}$$

where:

$$\sigma^2_{B_d} = 2.25k^{7/6}\sec(\zeta)^{11/6}\int_{h_0}^{H} C_n^2(h)\,(h-h_0)^{5/6}\,dh \tag{2.8}$$

$$\sigma_{B_u}^2 = 2.25k^{7/6}L^{5/6}\int_{h_0}^{H} C_n^2(h)\left(1-\frac{h-h_0}{H-h_0}\right)^{5/6}\left(\frac{h-h_0}{H-h_0}\right)^{5/6}dh \tag{2.9}$$

Both equations give an estimation of the maximum expected scintillation. The wave number is represented by $k = 2\pi/\lambda$, with λ as the wavelenght; the link distance by L; the zenith angle of the link path is ζ; the height of the OGS is h_0; and the height of the satellite is indicated by H.

After aperture averaging, scintillation values can be estimated using equation (2.10). This approximation is valid only under weak turbulence conditions, which applies until 30°.

$$\sigma_{I,av}^2 = 8.70k^{7/6}(H-h_0)^{5/6}\sec(\zeta)^{11/6}\Re\left\{\int_{h_0}^{H} C_n^2(h)\left[\left(\frac{kD^2}{16L}+i\frac{h-h_0}{H-h_0}\right)^{5/6}-\left(\frac{kD^2}{16L}\right)^{5/6}\right]dh\right\} \tag{2.10}$$

The refractive index structure parameter or index-of-refraction, denoted as C_n^2 [69], describes the turbulence strength along the transmission path by a certain profile. One of the most used models is the Hufnagel-Valley profile [70], shown in equation (2.11).

$$C_n^2(h) = Ae^{-h/100} + 2.7\times10^{-16}e^{-h/1500} + 0.00594\left(\frac{\nu}{27}\right)^2(10^{-5}h)^{10}e^{-h/1000} \tag{2.11}$$

This parameter depends on the height on the altitude above ground h; the turbulence at ground level, defined by the structure parameter at zero height $A = C_n^2(0)$; and the mean cross-wind velocity v. Based on these factors, the structure parameter can be optimized or even ignored depending on the location of the OGS [64], as shown in Fig. 2.26.

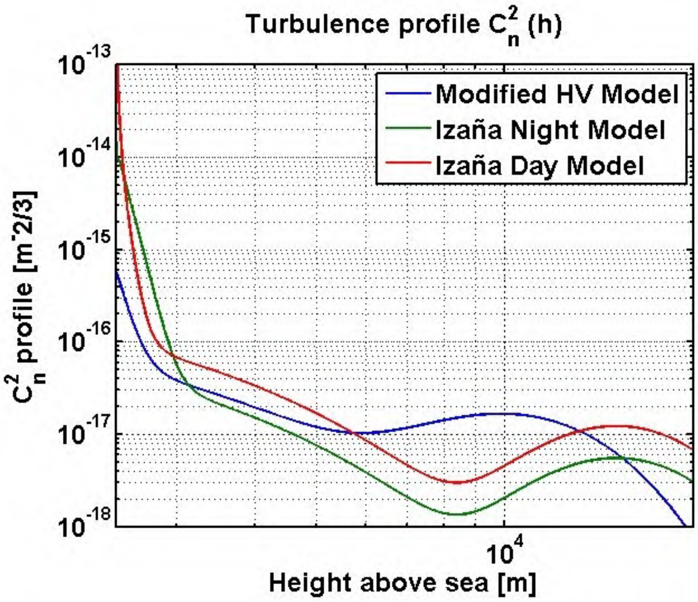


Figure 2.26: Sample turbulence profiles. High C_n^2 values mean strong atmospheric turbulence near to the surface. Figure taken from [33].

It is important for our application to discuss the relationship between scintillation and integration time. When using a monitoring device, such as an infrared camera, we can increase the exposure time to integrate the spatial fluctuations of the received signal. This is typically achieved from 100 ms onwards, as illustrated in Fig. 2.27.

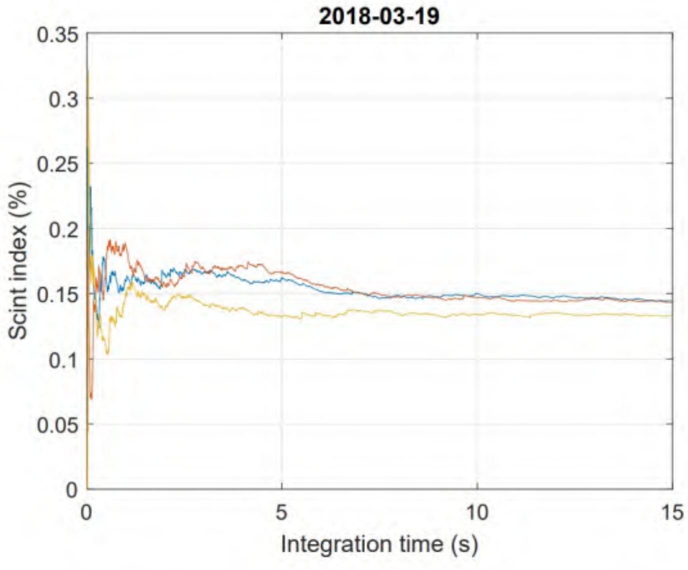


Figure 2.27: Scintillation index as a function of the integration time for 3 different data sets. The horizontal axis is limited to the range [0–15]s. Figure taken from [71].

The final atmospheric effect to consider is obstruction. Establishing an optical link requires a clear Line Of Sight (LOS) between the satellite and the OGS. The primary source of obstruction is cloud coverage. As demonstrated in [72], cloud attenuation cannot be significantly mitigated by selecting different wavelengths within the visible to near-infrared range. The most logical approach is to avoid these conditions when attempting to establish a link. One promising strategy, as suggested in [73], involves developing a network of OGSs to provide alternative communication pathways when cloud cover impacts the primary LOS [73].

2.4.5 Link Budget

The link budget is key for determining the performance of a lasercom system under various operating conditions. The total mean received power regarding all gains and losses can be calculated as the sum of all link budget components in dB [64]:

$$p_{Rx} = p_{Tx} + a_{Tx} + g_{Tx} + a_{BW} + a_{FSL} + a_{Atm} + a_{Sci} + g_{Rx} + a_{Rx} \tag{2.12}$$

p_{Tx}	average transmit optical source power	$[dBm]$
a_{Tx}	optical power loss inside the transmitter terminal	$[dB]$
g_{Tx}	transmitter antenna (telescope) gain	$[dB]$
a_{BW}	average loss by dynamic beam miss-pointing and beam wander	$[dB]$
a_{FSL}	free-space loss by link distance	$[dB]$
a_{Atm}	sum of atmospheric attenuation effects	$[dB]$
a_{Sci}	losses through atmospheric scintillation	$[dB]$
g_{Rx}	receiver antenna gain	$[dB]$
a_{Rx}	optical losses inside receiver terminal (attenuation and splitting)	$[dB]$
p_{Rx}	received power on detector	$[dBm]$

Gains (g) are positive values, while attenuations (a) are negative. The logarithmic calculation of the link budget simplifies the representation of complex propagation effects. The linear representation are G (values greater than 1 for gain) and A (with values ranging from 0 to 1 for attenuation).

- Tx-Antenna Gain $[dB]$:

$$g_{Tx} = 10\log_{10}\left(\frac{4\sqrt{\ln 2}}{\theta_{FWHM}}\right)^2 = 10\log_{10}\left(\frac{3.33}{\theta_{FWHM}}\right)^2 \tag{2.13}$$

Let θ_{FWHM} represents the Full Width at Half Maximum (FWHM) $= \sqrt{\frac{\ln 2}{2}}\theta_{e^{-2}}$. Where $\theta_{e^{-2}}$ denotes the full divergence angle $= 2\frac{\lambda}{\pi\cdot\omega_0(0)}$, with λ being the wavelength and $\omega_0(0)$ the waist radius at the beam's narrowest point.

- Pointing-error loss $[dB]$:

$$a_{BW} = 10\log_{10}\frac{\beta}{\beta+1} \tag{2.14}$$

Where $\beta = \frac{1}{2}\left(\frac{\theta_{e^{-2}}/2}{\sigma_{BW}}\right)^2 = \frac{\theta_{FWHM}{}^2}{4\cdot\ln 2\cdot\sigma_{BW}{}^2} = \left(\frac{0.85\cdot\theta_{FWHM}}{2\sigma_{BW}}\right)^2$ is a special case of the beta-distribution.

- Free-Space loss $[dB]$:

$$a_{FSL} = 10\log_{10}\left(\frac{\lambda}{4\pi L}\right)^2 \tag{2.15}$$

Where λ is the wavelength $[m]$, and the link distance L $[m]$ is computed as:

$$L = \sqrt{(R_E + H_{GS})^2[\sin\varepsilon]^2 + 2(H_O - H_{GS})(R_E + H_{GS}) + (H_O - H_{GS})^2} - (R_E + H_{GS})\sin\varepsilon \tag{2.16}$$

R_E	Earth radius	$[6370 \times 10^3 m]$
H_{GS}	height of the OGS over sea-level	$[DLR - OP = 650m]$
H_O	height of the circular orbit above Earth	$[m]$
ε	link elevation angle	$[°]$

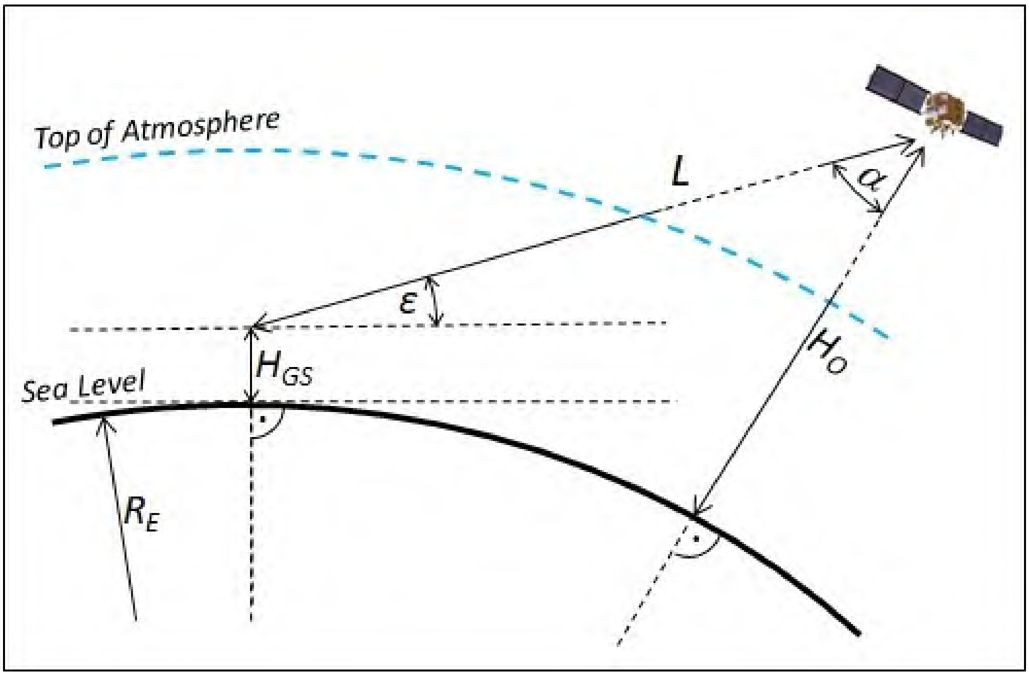


Figure 2.28: Angles and distances in the general triangle Sat-OGS-Earth. Figure taken from [64].

- Atmospheric effects loss $[dB]$:

$$a_{Atm} = 10\log_{10} T_Z^{1/\sin(\epsilon)} \tag{2.17}$$

As previously said in subsection 2.4.4, only aerosol absorption and scattering need to be considered. The zenith transmission value, T_Z, is computed assuming a flat-Earth model, as detailed [67]. Different values of T_Z will be used based on atmospheric conditions; for example $T_Z = 0.891$ for 1550nm under poor conditions [64].

- Scintillation loss $[dB]$ [74]:

$$a_{\mathrm{sci}} = 4.343\left\{\mathrm{erf}^{-1}(2p_{\mathrm{thr}}-1)\cdot\left[2\ln(\sigma_{\mathrm{p}}^2+1)\right]^{1/2} - \frac{1}{2}\ln(\sigma_{\mathrm{p}}^2+1)\right\}. \tag{2.18}$$

Where σ_{p}^2, the Power Scintillation Index (PSI), is defined as $\frac{\langle P_{Rx}^2\rangle - \langle P_{Rx}\rangle^2}{\langle P_{Rx}\rangle^2}$, representing the variance of the received optical power P_{Rx} $[W]$ [75]. The power threshold p_{thr} indicates the allowed fractional time during which the received power is above this threshold.

- Rx-Antenna Gain $[dB]$:

$$g_{Rx} = 10\log_{10}\left(\frac{4\pi A_{Rx}}{\lambda^2}\right) \tag{2.19}$$

Where A_{Rx} is the aperture area of the receiver, assumed to be smaller than the spot size. is assumed smaller than the spot size. When calculating the area of a Cassegrain-type telescope, we need to subtract the area of the inner obscuration, which is the area blocked by the secondary mirror.

The parameters p_{Tx} and a_{Tx} depend on the laser terminal, while a_{Rx} depends on the receiver telescope and is typically measured for each link.

2.5 Laser Transmitters for Optical Low Earth Orbit data DownLinks

Different types of laser terminals are employed for Optical Low Earth Orbit data DownLinks (OLEODL). These transmitters can be categorized into three distinct groups: No tracking systems, relaying on full body-pointing through satellite's attitude knowledge from star-camera sensors; Dynamic coarse body-pointing by the satellite during Optical Ground Station (OGS)-overflight, with fine-pointing achieved through beacon tracking by the optical terminal; and Terminals equipped with a coarse-pointing assembly for a hemispherical Field Of View (FOV), which actively track a ground-based beacon.

2.5.1 OSIRISv1 Onboard Flying Laptop

The small satellite "Flying Laptop" (FLP), launched in 2017, was developed and built by students at the University of Stuttgart. It has a mass of 110 kg and is equipped with the first version of the Optical Space InfraRed downlInk System (OSIRIS) [76], built by the Institute of Communications and Navigation (IKN) at the Deutsches Zentrum für Luft- und Raumfahrt (DLR). The satellite employs an open-loop body pointing mode, avoiding dedicated optomechanical pointing assemblies. Instead, the satellite relies on its entire rotation, controlled by onboard star cameras, to point towards the ground station without feedback from the instrument [77].

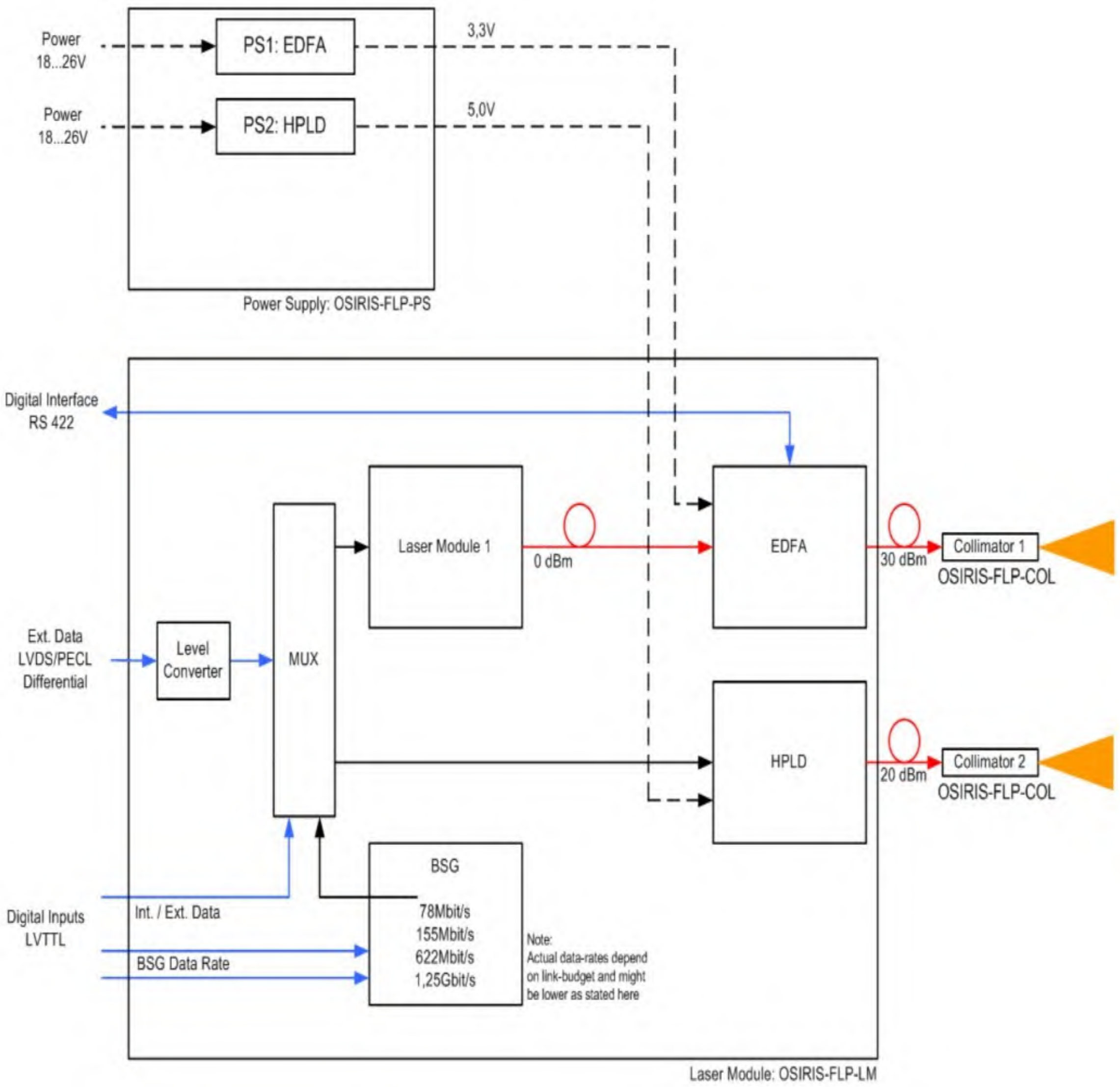


Figure 2.29: OSIRISv1 setup, as located onboard FLP. Figure taken from [78].

Figure 2.30: "Flying Laptop" satellite. Figure taken from [79].

2.5.2 OSIRIS4CubeSat Onboard Laser CubeSat

In contrast to the other OSIRIS payloads that focus on increasing data rates, OSIRIS4CubeSat aims to achieve a highly compact system design that enables the use of optical communication even on small satellites like CubeSats. Developed by the Deutsches Zentrum für Luft- und Raumfahrt (DLR) on behalf of Tesat Spacecom, the OSIRIS4CubeSat terminal, also known as CubeSat Laser Communication

Transmitter (CubeLCT), has established itself as the smallest laser communication terminal in the world.

The first demonstration of complete end-to-end transmission took place as part of the PIXL-1 mission [80]. On January 24, 2021, the CubeL satellite carrying the CubeLCT terminal was launched into space.

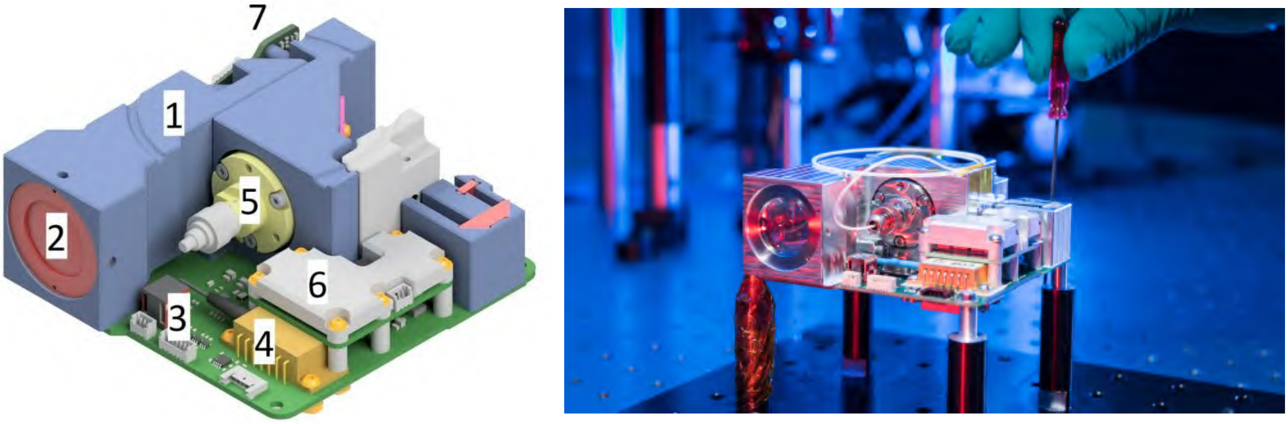


Figure 2.31: 3D model of OSIRIS4CubeSat Payload (left). Figure taken from [81]. OSIRIS4C - Flight model of the laser terminal OSIRIS4CubeSat (right). Figure taken from [82].

As shown in Fig. 2.31, the optomechanics, shown in blue (1), consist of the mechanical mounts and housings for the optical elements, highlighted in light red (2) while the electronics mainboard is depicted in green (3). The transmission system consists of the laser source in orange (4), and the transmit collimator, shown in yellow (5). The driver electronics are mounted on the green Printed Circuit Board (PCB) below a passive cooling element in gray (6). The receiver sensor is located on a separate PCB at the back of the terminal (7).

The OSIRIS4C system combines body pointing with closed-loop tracking. To establish a connection, the terminal uses the Pointing Acquisition and Tracking (PAT) system. A relatively broad beacon signal is sent from an OGS to illuminate the satellite. This beacon is acquired by a 4-Quadrant Diode (4QD), functioning as a tracking sensor within the satellite. The 4QD measures the angular error, and a Fine Steering Mirror (FSM) then corrects it. Throughout the transmission, the satellite must maintain a pointing accuracy of better than 1° towards the OGS. To ensure that the transmission beam accurately reaches the OGS, it is coupled into the same optical path as the incoming beacon.

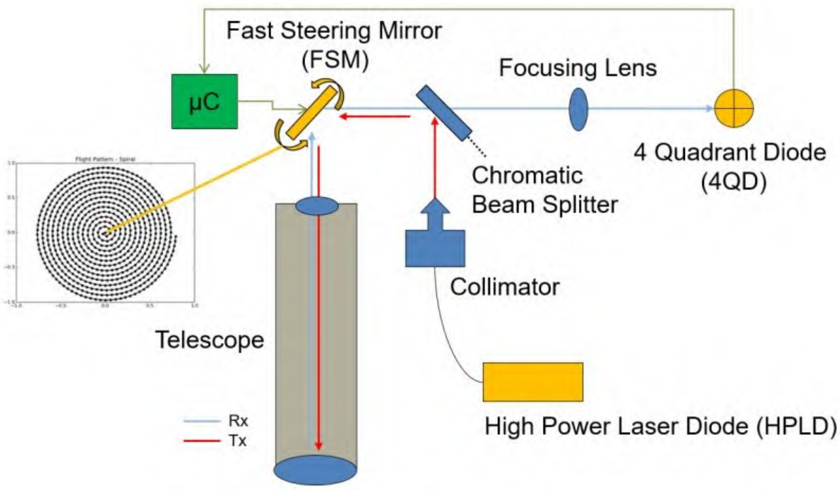


Figure 2.32: Block diagram of Fine Pointing Assembly (FPA) inside OSIRIS4C. Figure taken from [80].

The primary objective of the PIXL-1 mission was to integrate a high-power laser terminal into a 3U

CubeSat, such as the Laser CubeSat (CubeL), extending CubeSats' data transmission rates up to 100 Mbit/s.

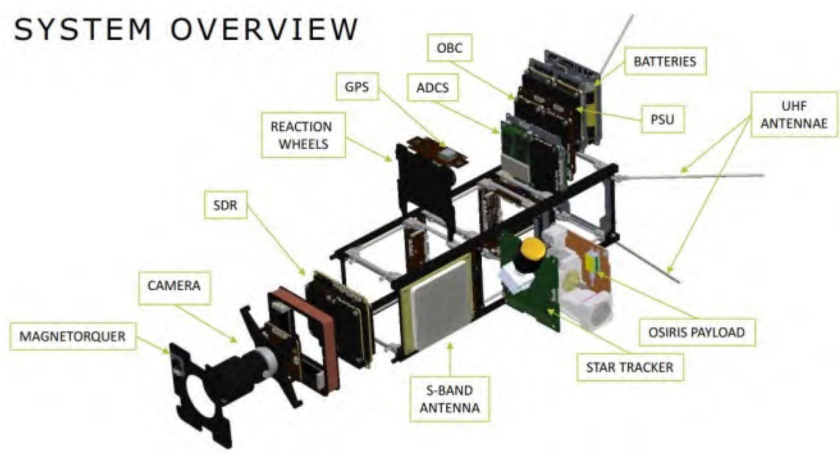


Figure 2.33: System overview of CubeL (left). Integrated CubeL (right). Figure taken from [81].

2.5.3 OSIRISv3 Onboard Titania

The next stage in the development of the OSIRIS program, OSIRISv3, aims to achieve higher data rates, up to 10 Gbit/s. With a smaller divergence, OSIRISv3 relies on a new dedicated alignment unit. This so-called Coarse Pointing Assembly (CPA) enables hemispheric movement, decoupling the laser beam from the satellite's position. The transmitter is also equipped with a FPA and a point ahead assembly for improved accuracy and performance.

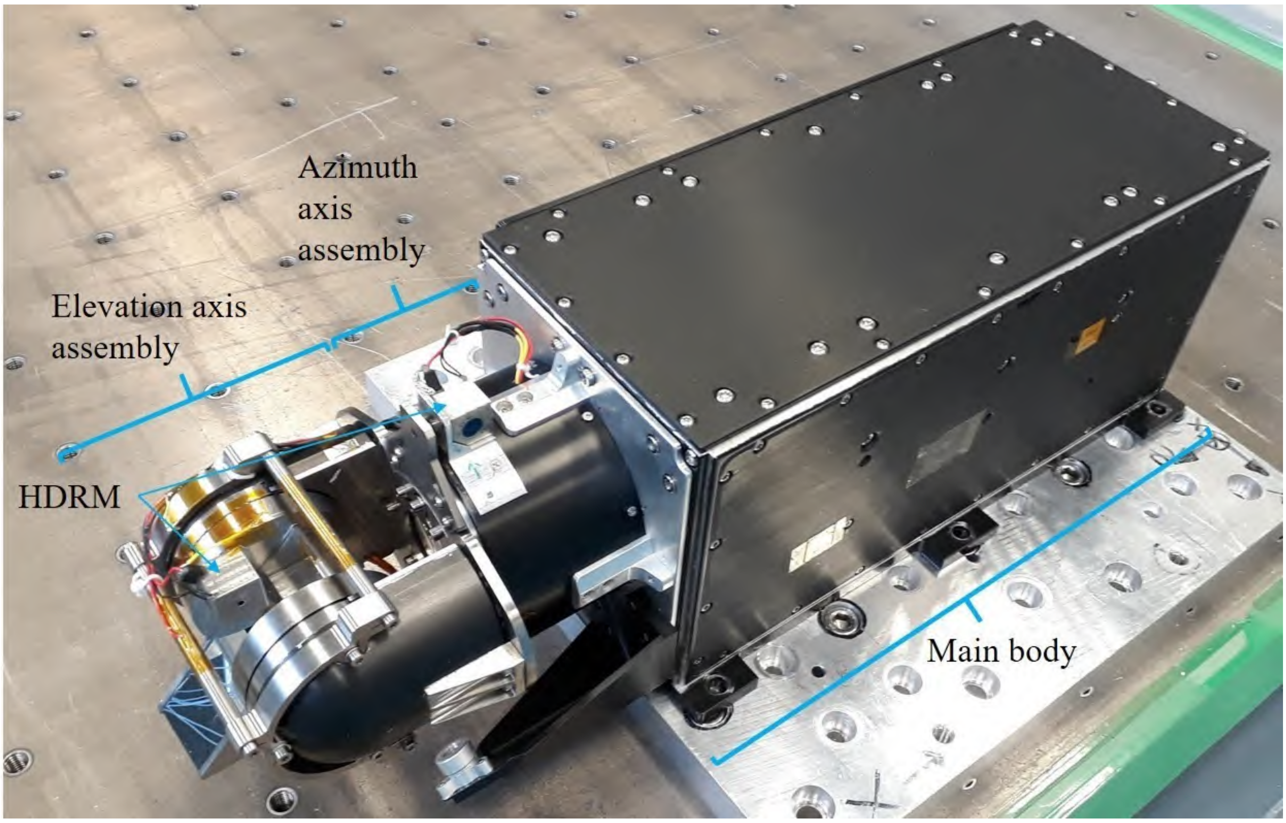


Figure 2.34: OSIRISv3 laser communication Terminal including CPA. Figure taken from [83].

The OSIRISv3 system provides a hemispherical Field Of Regard (FOR), which is the total area that can be observed by the movable sensor. This FOR is achieved through two axes of movement—elevation and azimuth—while maintaining an optical aperture of 30 mm. One key aspect of this system is the Absolute Pointing Knowledge (APK) in both axes, which measures the satellite's ability to accurately determine its orientation in space. The position data is transmitted to the motor controller unit, which manages the motor movements along the elevation and azimuth axes. For a Free-Space Optical (FSO) link to an optical ground station, the OSIRISv3 CPA steers the beam during the full overflight.

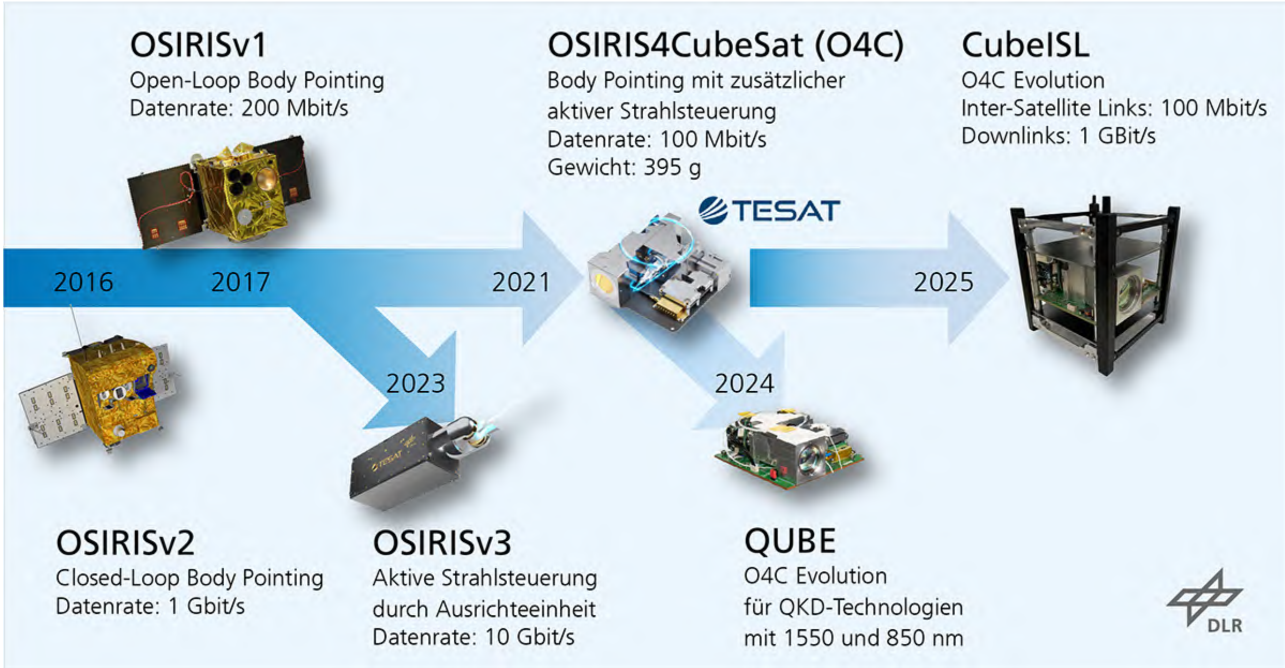


Figure 2.35: OSIRIS-Program Road map. Figure taken from [84].

3 Materials and Methods

In this chapter we will discuss the design, implementation, and testing of the AllSky-camera system, including its constrains and reasons behind the multiple decisions made throughout the process.

3.1 System Requirements

The proposed system was designed as a compact, portable, and self-sufficient device for use during Low Earth Orbit (LEO) satellite downlinks. Based on an Infrared (IR) camera equipped with a wide Field of View (FOV) lens, it covers the entire hemisphere without the need for satellite orbit information or pointing. This approach allows continuous observation of satellite passes, providing real-time assessment of elevation, azimuth, and intensity as detected by the camera with minimal user intervention. The system could be remotely operated or even fixed as a permanent installation in the future, serving dual purposes: evaluating the satellite's pointing accuracy, and acting as a validation tool for the Optical Ground Station (OGS) by verifying the correlation of results.

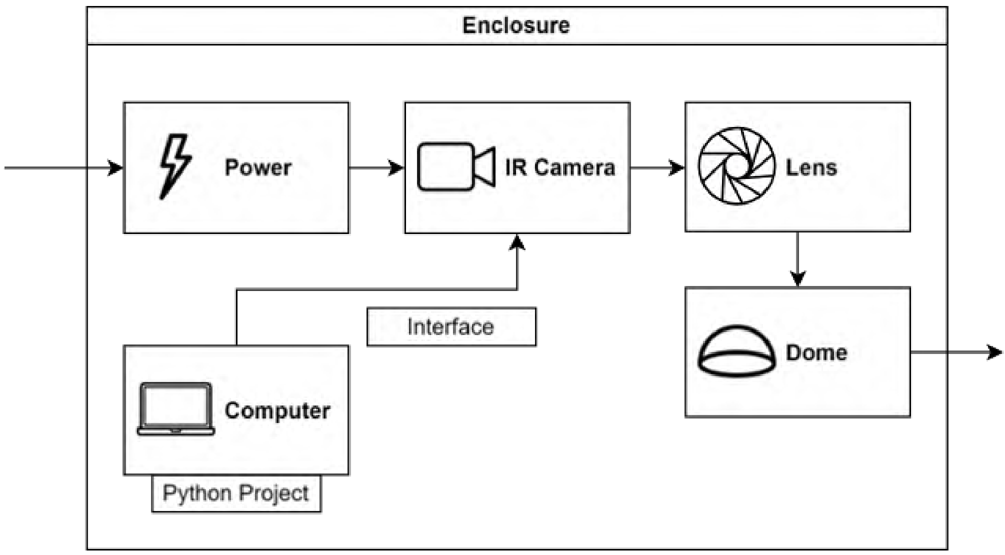


Figure 3.1: Initial diagram of the AllSky-camera system.

Given the system's operational conditions, waterproofing, ventilation, and internet connectivity must be regarded. Meeting those requirements was challenging, as we will discuss later.

3.2 Component Selection

The first step after defining the goals and requirements of the project, was to decide which components were the most suitable for our application. The key elements are the camera, the lens, the dome and the enclosure.

3.2.1 Indium Gallium Arsenide Camera

The camera serves as the primary component of our system. The satellites under observation are equipped with a 1550nm laser terminal. Indium Gallium Arsenide (InGaAs) is the most appropriate material for working at this wavelength.

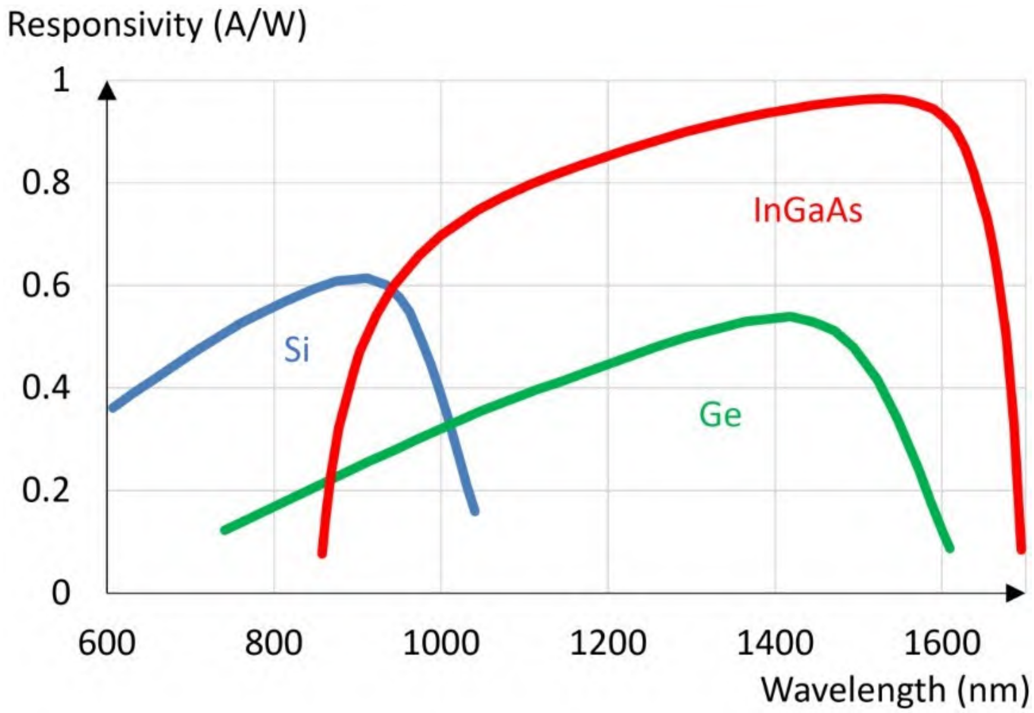


Figure 3.2: Typical responsivity vs wavelength for silicon, InGaAs, and germanium. Figure taken from [29]

Initially, the plan was to utilize the newly developed SONY Short-Wavelength InfraRed (SWIR) Image Sensor Technology, known as SenSWIR. This technology integrates InGaAs photodiodes with silicon readout circuits via Copper (Cu)-Copper bonding, resulting in a wide-band and highly sensitive SWIR image sensor. SenSWIR allows imaging across a broad spectrum of wavelengths, ranging from 0.4 μm to 1.7 μm. A single camera can now capture both the visible (VIS) light and the SWIR spectrum, which previously required separate cameras. These cameras are called Visible to Short-Wavelength InfraRed (VSWIR) cameras.

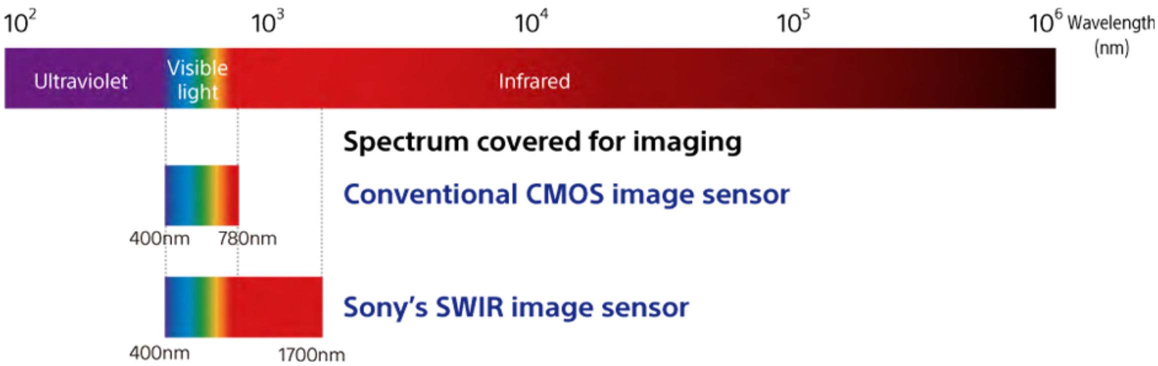


Figure 3.3: Operational Range SONY'S SWIR Sensor. Figure taken from [85]

There are two versions of the chip: The General-purpose SWIR Image Sensor (IMX990/IMX991) and the High-resolution, High-performance SWIR Image Sensor (IMX992/IMX993). The general-purpose chips feature 5 μm pixels, whereas the high-performance variants feature 3.45 μm pixels. The main problem of these sensors is that the IMX992 and IMX993 are not yet available for purchase until Q4 2024, and using the IMX990 or IMX991 would compromise the Field Of View (FOV) of our final system. The FOV of an imaging system depends on the focal length of the lens and the dimensions of the sensor of the camera, as we will later discuss in subsection 3.2.2.

The dimensions of a camera's sensor are calculated as follows, where the result can then be approximated to match one of the standard sensor sizes, as shown in Fig. 3.4:

$$V_{sensor\ size}\ [mm] = V_{resolution}\ [pixels] \cdot P_{size}\ [\mu m] \tag{3.1}$$

$$H_{sensor\ size}\ [mm] = H_{resolution}\ [pixels] \cdot P_{size}\ [\mu m] \tag{3.2}$$

$$D_{sensor\ size}\ [mm] = \sqrt{(H_{sensor\ size}\ [mm])^2 + (V_{sensor\ size}\ [mm])^2} \tag{3.3}$$

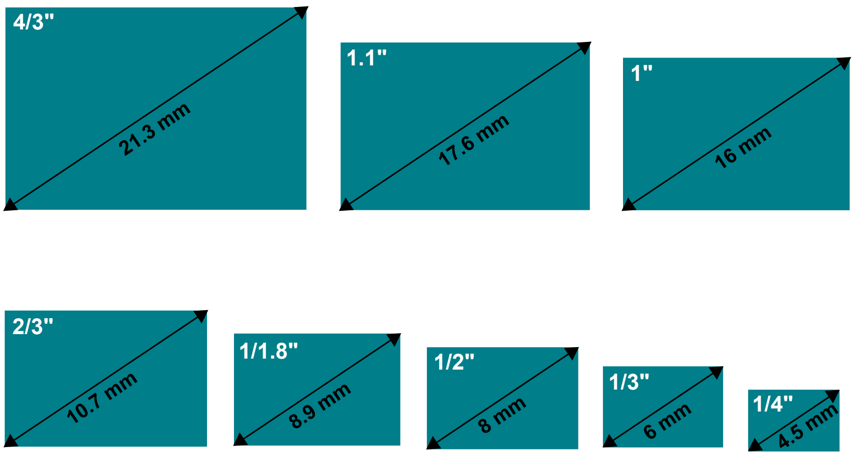


Figure 3.4: Common sensor formats and their diagonal dimensions. Figure taken from [86].

Device structure

Thermoelectric cooling element included model	IMX990-AABA	IMX991-AABA	IMX992-AABA	IMX993-AABA
Thermoelectric cooling element not included model	IMX990-AABJ	IMX991-AABJ	IMX992-AABJ	IMX993-AABJ
Image size	8.2 mm diagonal (Type 1/2)	4.1 mm diagonal (Type 1/4)	11.4 mm diagonal (Type 1/1.4)	8.9 mm diagonal (Type 1/1.8)
Effective pixels	1296 (H) × 1032 (V) approx. 1.34 megapixels	656 (H) × 520 (V) approx. 0.34 megapixels	2592 (H) × 2056 (V) approx. 5.32 megapixels	2080 (H) × 1544 (V) approx. 3.21 megapixels
Unit cell size	5 µm (H) × 5 µm (V)		3.45 µm (H) × 3.45 µm (V)	
Optical black — Horizontal direction	Front 0 pixels, rear 96 pixels		Front 96 pixels, rear 0 pixels	
Optical black — Vertical direction	front 20 pixels, rear 0 pixels		front 24 pixels, rear 0 pixels	
Input drive frequency	37.125 MHz/74.25 MHz/54 MHz		37.125 MHz/74.25 MHz/54 MHz	
Power supply	1.2 V, 1.8 V, 2.2 V, 3.3 V, 1.2 V (Pixel), 2.2 V (Pixel)		1.2 V, 1.8 V, 2.2 V, 3.3 V, 2.2 V (Pixel)	
Shutter mode	Global shutter		Global shutter (rolling shutter when DRRS on)	
Output interface	SLVS (2 ch/4 ch)		SLVS (2 ch/4 ch/8 ch) / MIPI (2 lane/4 lane)	
Package	Thermoelectric cooling element included:30.0 mm (H) × 30.0 mm (V) Thermoelectric cooling element not included:20.0 mm (H) × 16.8 mm (V)		Thermoelectric cooling element included:30.0 mm (H) × 30.0 mm (V) Thermoelectric cooling element not included:21.0 mm (H) × 20.0 mm (V)	

Figure 3.5: Specifications of the SONY SWIR image sensors. Figure taken from [85].

After careful evaluation, we decided to disregard the SenSWIR sensors and instead prioritized selecting the camera with the largest possible sensor, we chose the Goldeye G-008 TEC1 [87]. The selection of this specific camera was mainly backed by its software and Application Programming Interface (API) as specifications are consistent across all considered cameras.

The C-mount camera records in 320 × 256 pixels at 344 frames per second. Users can select between 8-bit, 12-bit, 12-bit packed, or 14-bit monochrome pixel format. The camera also features advanced image processing capabilities, such as background correction, Defect Pixel Correction (DPC), and Look-Up Tables (LUTs). Designed for durability, the camera operates within a temperature range of -20°C to +55°C, making it suitable for extended use in harsh environments. Additionally, the camera exhibits a quantum efficiency of approximately 78% at 1550 nm, as illustrated in Fig. 3.6.

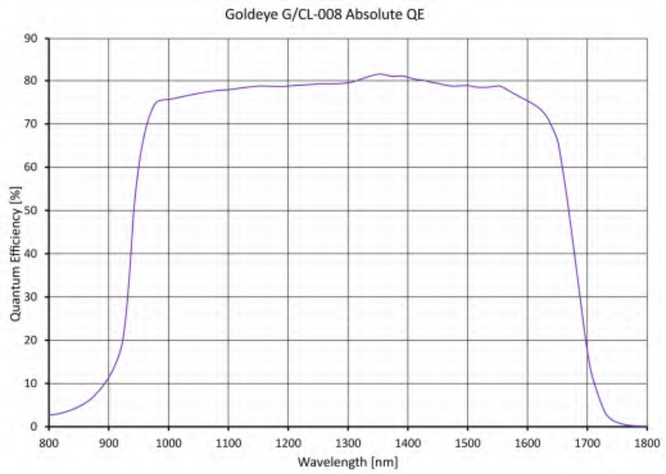


Figure 3.6: Goldeye G-008 TEC1 camera. (left). Quantum efficiency. (right). Figure taken from [87].

Table 3.1: Comparison of different cameras.

Specification	**Goldeye G-130 TEC1** [88]	**Goldeye G-008 TEC1** [87]	**Xenics Bobcat 320 CL 400** [89]
Interface	Power over Ethernet	Power over Ethernet	CameraLink
Spectral range [nm]	400 - 1700	900 - 1700	900 - 1700
Resolution [H×V $px.$]	1280 × 1024	320 × 256	320 × 256
Pixel size [H×V μm]	5 × 5	30 × 30	20 × 20
Sensor	Sony IMX990 | Type 1/2" VSWIR	Type 2/3" SWIR	Type 1/2" SWIR
Lens mount	C-Mount	C-Mount	C-Mount
Max. frames per second	94	344	400
Bit depth	8-bit to 12-bit	8-bit to 14-bit	8-bit to 16-bit
Power consumption [W]	<12.95	<12.95	2.8
Operating temperature [°C]	-20 to +55	-20 to +55	-40 to +70
Body dimensions*			
[L×W×H mm]	78 × 55 × 55	78 × 55 × 55	72 × 55 × 55
Mass* [g]	340	340	285

*Lens not included.

3.2.2 Wide Angle Lens

Once an appropriate InGaAs camera is selected, the next challenge is finding a suitable C-Mount lens that meets our requirements. The primary concern is achieving the largest possible field of view, ideally close to 180 degrees. Fulfilling this requirement is challenging due to the limitations of the C-Mount standard, which being relatively small, makes it difficult to accommodate the larger lenses providing a wide FOV. Additionally, the requirement for InfraRed (IR) operation adds further complexity to the selection process.

We considered two different approaches, the first used a specialized SWIR lens. The main problem of these kind of lenses, besides being expensive (around €2000), is their limited FOV—there are no lenses with a wide enough FOV for our application, as the materials required to construct them also make the lenses tedious to miniaturize, creating optics that are too large for a C-Mount (lenses like this can be custom-made, they are just not available in the market as they go in a different direction to the intended).

The second option featured a Visual and InfraRed (VIR) lens. Visible light lenses are more affordable and offer wider FOVs. While these lenses can operate in the infrared spectrum, their transmittance is reduced, leading to potential chromatic aberrations. This occurs when the lens fails to focus all wavelengths to the same convergence point or image plane, resulting in color distortion and reduced image quality.

Table 3.2: Comparison of different lenses.

Specification	**NAVITAR SWIR 8** [90]	**KOWA LM100JC1MS** [91]	**THOR-LABS MVL4WA** [92]	**FUJINON FE185C057HA-1** [93]
Spectral range [nm]	700 - 1900 (SWIR)	380 - 1400 (VIR)	380 - 1400 (VIR)	380 - 780 (VIS)
Transitivity at 1550 nm [%]	81.6*	65**	56.8***	42****
Focal length [nm]	8	100	3.5	1.8
F-Number [$f/\#$]	1.4 - 16	2.8 - 32	1.4 - 16	1.4 - 16
Focus control	Manual	Manual	Manual	Fixed
Max. Aperture diameter [mm]	5.7	35.7	2.5	1.3
Diagonal Field of View [°]				
2/3” sensor size	92.4	6.3	140*****	185
1/2” sensor size	70.7	4.6	132.1	185
Operating temperature [°C]	-10 to +45	-10 to +50	-10 to +45	-10 to +50
Mass* [g]	205	145	70	135

*Transitivity at 1500 nm - Confidential Document.
**Transitivity at 1400 nm [94].
***Transitivity at 1400 nm [95].
****Transitivity at 1000 nm - Confidential Document.
*****In theory not compatible with 2/3” sensor size.

As we can see in Tab. 3.2, the narrow fields of view render the Navitar and Kowa lenses unusable. When comparing the Thorlabs and Fujinon lenses, a compromise must be made: the MVL4WA offers a narrower field of view with higher transmittance, while the FE185C057HA-1 provides a wider field of view at the expense of lower transmittance. The Thorlabs lens is also incompatible with 2/3” sensors, which could lead to potential issues such as distortion, vignetting, or cropping of the field of view.

Given the difficulty of deciding based just on this data, we tested both lenses to determine which one would be more suitable for our specific application. The results of these tests will be discussed in subsection 3.5.2.

Figure 3.7: Fujinon FE185C057HA-1. (left). Figure taken from [93]. Thorlabs MVL4WA. (right). Figure taken from [96].

3.2.3 Dome

The lens will be protected from the exterior by a dome. We explored multiple options to ensure high transmittance at 1550 *nm*. We inquired multiple companies, such as TTV and RÖHM, about the transitivity of their Plexiglas/acrylic sheets that might meet our requirements. While the performance of the materials was adequate, shaping them into a dome proved to be complex and impractical. As an alternative, we considered crystal domes made from 1550 *nm* compatible materials.

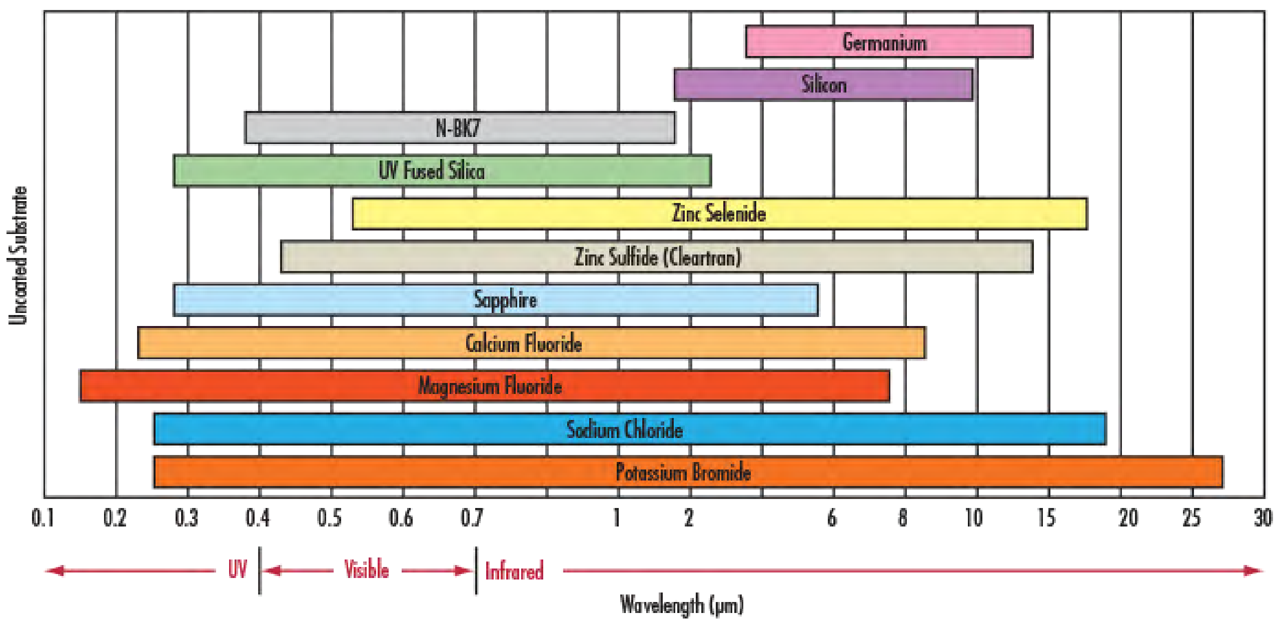


Figure 3.8: Infrared Substrate Comparison (Wavelength Range for N-BK7 is Representative for the Majority of Substrates Used for Visible Wavelengths Such as B270, N-SF11, BOROFLOAT®, etc.). Figure taken from [97]

N-BK7, fused silica, and sapphire exhibited excellent transmittance, but their cost was too much for what they provided. Ultimately, we opted for a plexiglas dome that we already had in our possession, as its performance in our tests was acceptable.

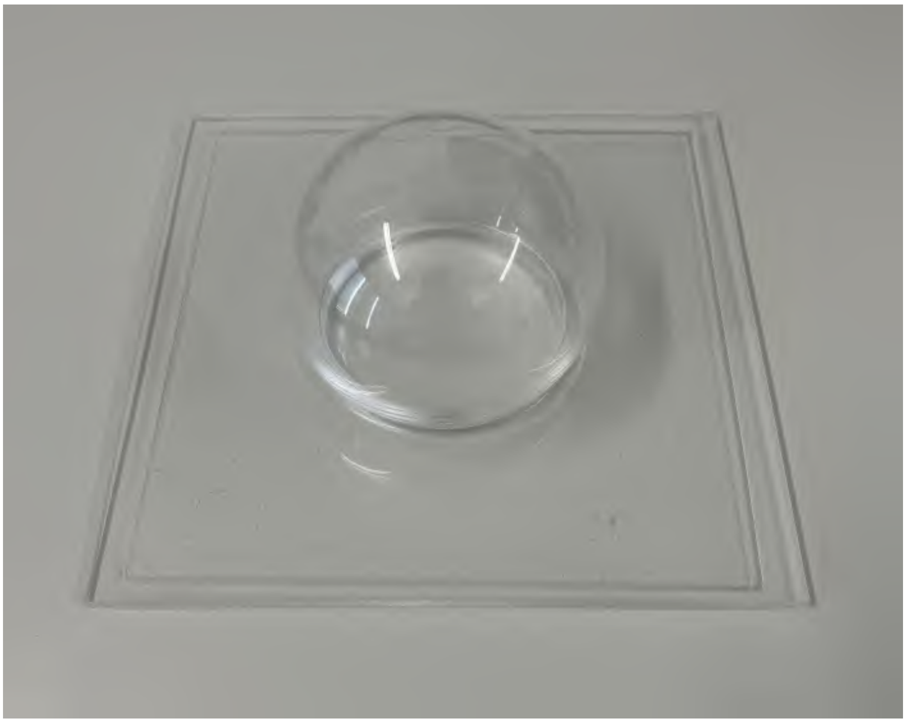


Figure 3.9: Final plexiglass dome.

3.2.4 Enclosure

All components must be enclosed in a box for transport and operation. This enclosure must meet specific requirements to withstand harsh conditions: waterproofing; dustproofing; and sturdiness, to ensure the components are fixed and can be moved without risk. The inclusion of a laptop for operating the camera forces us to increase the dimensions of the box.

Figure 3.10: Max Koffer MAX540H245 box closed. (left). Max Koffer MAX540H245 box opened. (right). Figures taken from [98].

The enclosure displayed in Fig. 3.10 has an IP67 rating, meaning it is dustproof, airtight, and waterproof, even if temporarily submerged. With internal dimensions of 53.8 × 40.5 × 19.5 [$L \times W \times H$ cm], weight of 7.7 kg, and operating temperature range from -30°C to +90°C, the box is more than sufficient to house all the required components and protect them from external conditions.

3.3 Final Overview of the AllSky-camera System

The final device will be controlled by a laptop housed inside the enclosure. This laptop will be connected to a switch, linking the laptop and the camera and providing an option for internet connectivity. The camera needs to go through a Power Over Ethernet (POE) injector before the switch to be granted power, this could be ignored if POE switch was available, but is not the case.

A four-outlet power strip will supply power to the system: one outlet for the laptop charger, one for the switch, one for the POE injector, and an additional outlet for a potential future dust filter. The power cable of the power strip will exit the left side of the enclosure through a cable gland, maintaining the system's waterproof integrity.

The camera will be mounted on the left side of the lid, opposite the laptop. A hole will be made in the lid for the camera lens, which will be protected from the exterior by the dome. The assembly of the dome is key to prevent condensation and ensure the box remains waterproof.

One challenge with this device is that when the laptop needs to be pulled out for operation, the Ethernet cable connected to the switch gets pinched between the lid and the box—not being able to close properly—the solution was to install a double Ethernet socket on the right of the box. Both internal connectors of the socket will be connected to the switch, this way if the laptop is inside of the box, the internet cable will be connected to one of the external socket connectors, leaving the second one free. If the laptop is outside the box, the internet cable will remain connected to one socket connector while the laptop will be connected to the second connector, keeping connection to the switch. The final iteration of the device can be shown in Fig. 3.11.

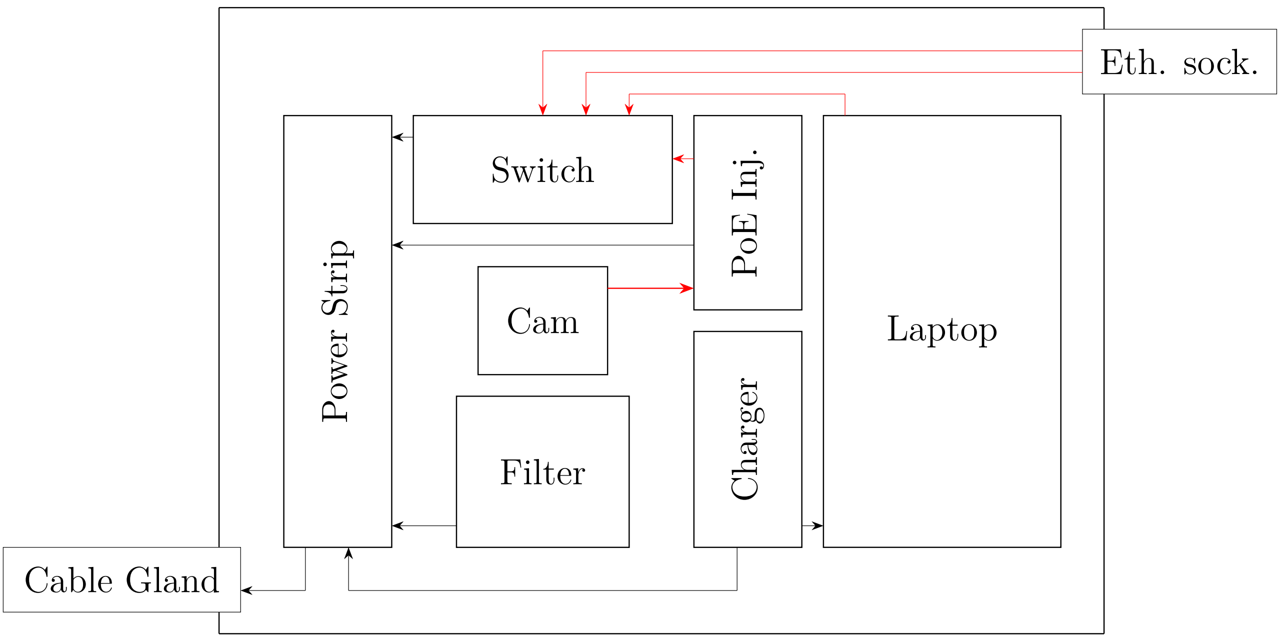


Figure 3.11: Final diagram of the proposed AllSkyCam4OLEODL system. Ethernet cables in red, power cables in black.

3.4 Controlling Software

We developed software capable of managing the camera operations and processing the images to obtain the required measurements. We will briefly explain the functionality of the code, rather than its construction, as detailed documentation has been written for that purpose [99]. The source code is included both in the documentation and in Appendix A.

Python was chosen as the programming language because the Application Programming Interface (API) is available only in C or Python [100], with Python being faster for creating an initial iteration of the software. In the future, the software could be ported to C if performance becomes a critical factor, but the image processing would be challenging.

The program installation files and process can be found in the GitHub repository [101]. Both Git and Python (version 3.7.4 or higher) need to be previously installed.

1. Install the PyPI package:

```
pip install allskycam4oleodl
```

2. Clone the AllSkyCam4OLEODL Git repository:

```
git clone https://github.com/Ikerald/AllSkyCam4OLEODL.git
```

3. Navigate to the AllSkyCam4OLEODL directory:

```
cd AllSkyCam4OLEODL/
```

4. Manually install the VmbPy API:

```
pip install './data/vmbpy-1.0.4-py3-none-any.whl[numpy,opencv]'
```

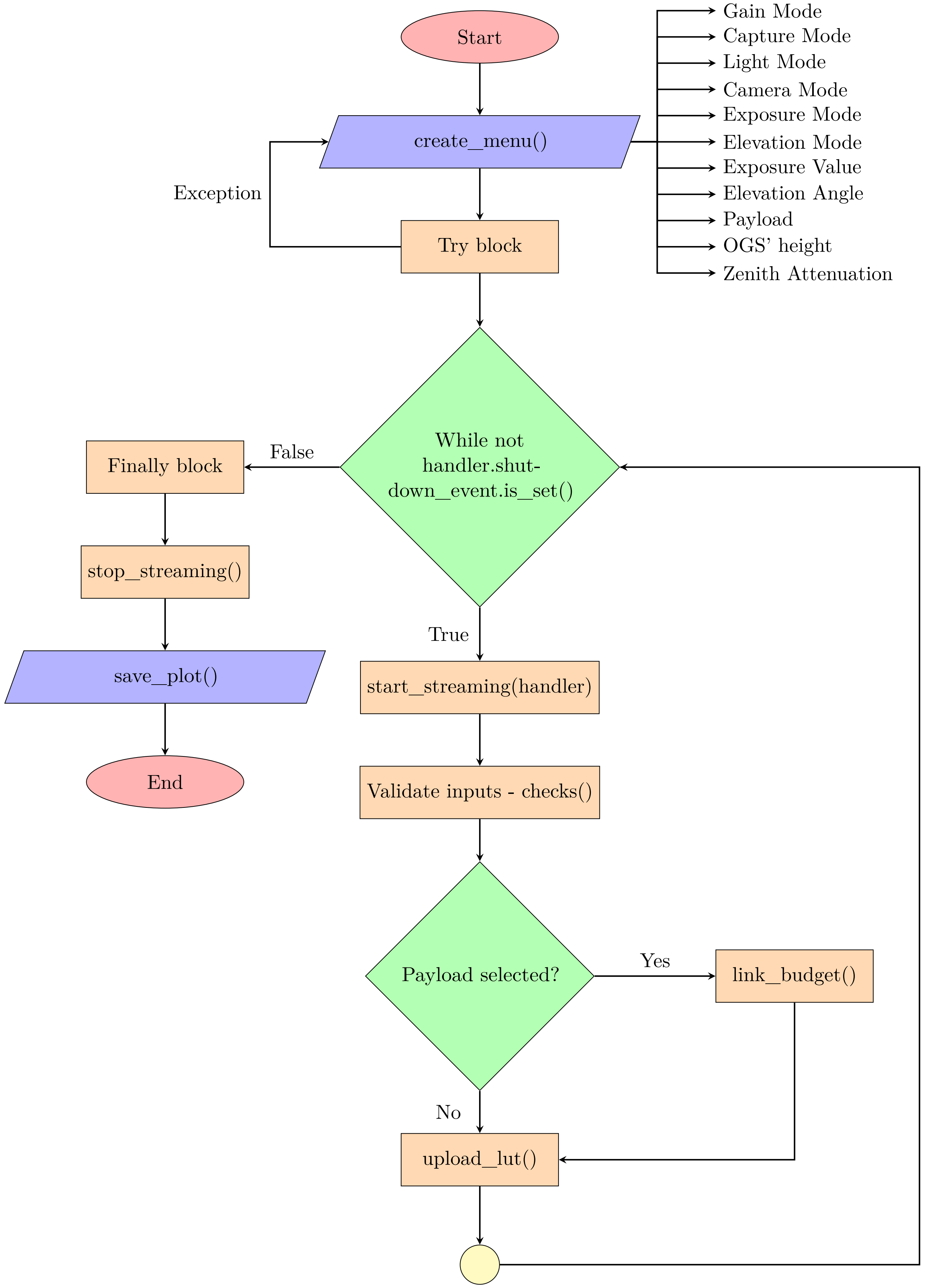
5. Execute the program:

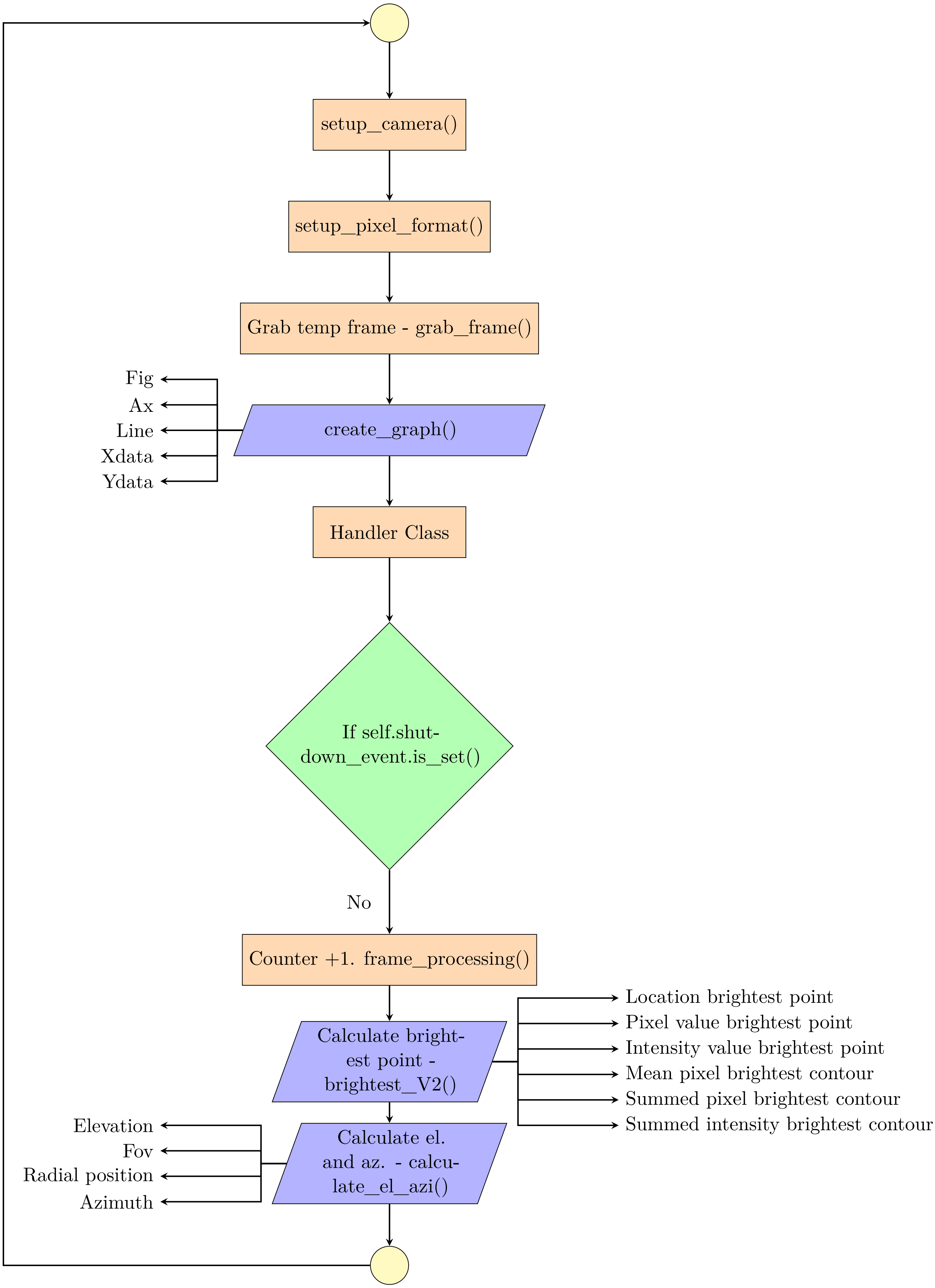
```
python main.py
```

The program is executed from the *main.py* file, which imports all necessary functions from the AllSkyCam4OLEODL package. This package consists of seven different scripts:

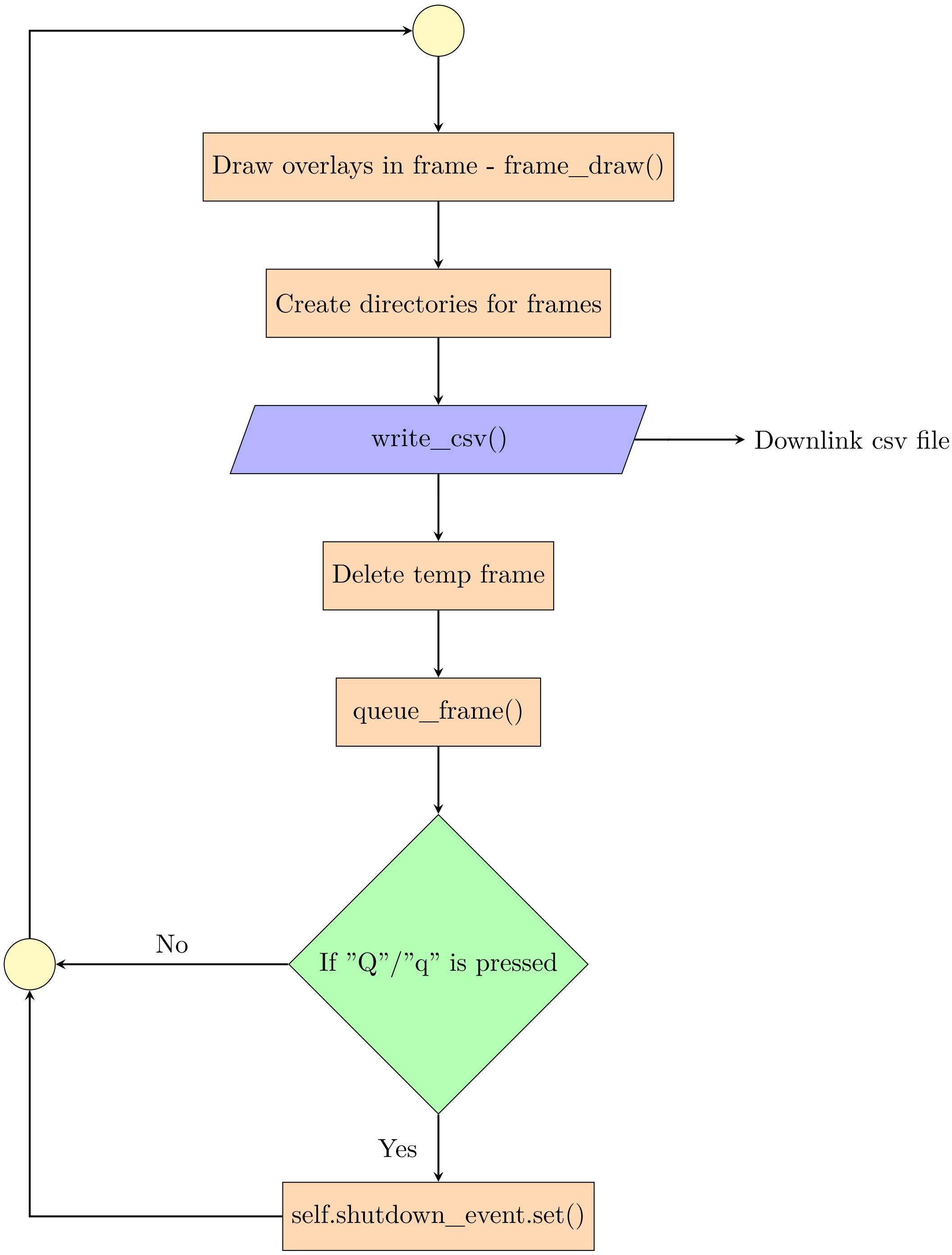
- *api.py*: Manages the API for camera control.
- *constants.py*: Stores all necessary constant values.
- *gui.py*: Handles the Graphical User Interface (GUI).
- *image_processing.py*: Processes the frames.
- *input_checks.py*: Grabs and validates input values.
- *link_budget.py*: Calculates and prints the link budget.
- *printer.py*: Prints the preambles.

The program operates as indicated in the following flow chart. This is a simplified diagram, as to accurately represent every aspect of the code, we would need an individual diagram for each function.


Start
create_menu()
Gain Mode
Capture Mode
Light Mode
Camera Mode
Exposure Mode
Elevation Mode
Exposure Value
Elevation Angle
Payload
OGS' height
Zenith Attenuation
Exception
Try block
While not handler.shut-down_event.is_set()
False
Finally block
stop_streaming()
save_plot()
End
True
start_streaming(handler)
Validate inputs - checks()
Payload selected?
Yes
link_budget()
No
upload_lut()



setup_camera()
setup_pixel_format()
Grab temp frame - grab_frame()
create_graph()
Fig
Ax
Line
Xdata
Ydata
Handler Class
If self.shut-down_event.is_set()
No
Counter +1. frame_processing()
Calculate brightest point - brightest_V2()
Location brightest point
Pixel value brightest point
Intensity value brightest point
Mean pixel brightest contour
Summed pixel brightest contour
Summed intensity brightest contour
Calculate el. and az. - calculate_el_azi()
Elevation
Fov
Radial position
Azimuth



3.4.1 Vmbpy Application Programming Interface

The Allied Vision's python API, Vmbpy [102], gives us control over most of the camera's settings directly from the code, allowing us to expand its functionalities.

One of the most important functions is *setup_camera*() in *api.py*. This function allows us to configure the exposure mode, which can be set either in manual or automatic, and the gain mode, which is chosen between gain 0 (0 dB), or gain 1 (18 dB)—the latter is preferred, as it enhances the visibility of the satellite. In *setup_pixel_format*() the pixel format is set to 8-bit. Although our camera is a 14-bit camera, this format is incompatible with the main processing tools like OpenCV or Pillow, forcing us to use the inferior mode.

The *upload_lut*() function is responsible for uploading the Look-Up Table (LUT) to the camera according to the selected mode. The goal is to correct the slight non-linearity in the camera output, as can be seen in Fig. 3.12, 3.13 and 3.14.

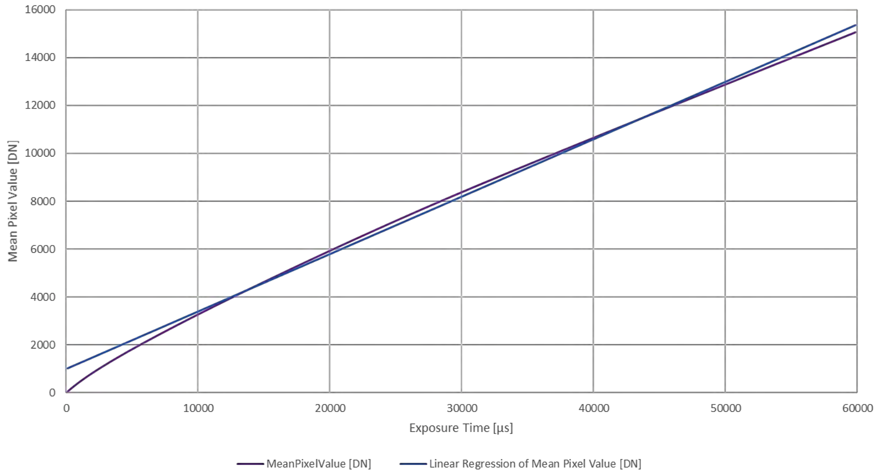


Figure 3.12: Mean pixel value of the camera with no LUT uploaded. Figure taken from [103]

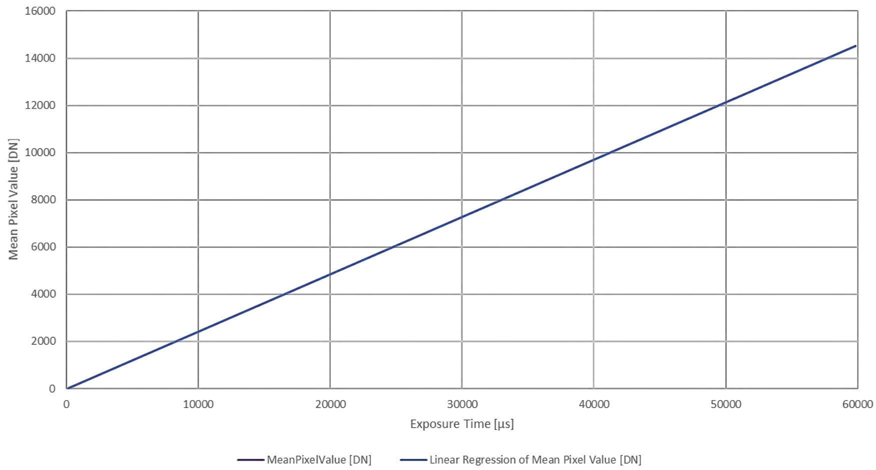


Figure 3.13: Mean pixel value of camera with LUT uploaded. Figure taken from [103]

Camera	Gain	Linearity Error LE [%]
G-008	Gain0	3.74
	Gain1	0.17

Figure 3.14: Percentage of linearity error of both modes. Figure taken from [103]

3.4.2 Image Processing

The main goal of the software is to process all captured images in real time and tracking the satellite while assessing its intensity, elevation and azimuth angle. This posed a challenge, as we had to balance what we wanted to do with the computational resources needed to do so.

The process goes as follows: first, the taken frame from the camera is duplicated. If we want to assess the intensity based on the pixel values of the image, the original must remain unaltered, or the values will change. Depending on the selected camera mode, different processing algorithms are applied.

- Hot-pixel removal:

  We captured a set of images with exposure times ranging from 20000 μs to 300000 μs with the lid on (lower exposure times were not considered, as hot pixels are not present at those levels). These images, which feature only the hot pixels of the camera, are stored in the following directory:

```
cd AllSkyCam4OLEODL\data\references\gain1
```

These frames will be thresholded with a value of 45 and scaled by 255, obtaining a black image with the hot pixels as maximum values. The normalized images will be subtracted from the temporal image, eliminating the hot pixels.

The drawback of this approach is that not only eliminates the hot pixels, but also sets the affected pixels to zero. If the satellite were to pass through one of these pixels, its value would be zero.

- Own background subtraction:

  In this mode, the temporal frame is directly subtracted from the current frame. One way to apply this subtraction is capturing the first frame with the lid on top, this way we can delete the background noise from the image.

  However, changing exposure times during operation makes the subtraction of the black values inaccurate, as we cannot put the lid back on to retake a frame with the updated exposure.

  The other way to use this mode is activating it with the lid already taken off, subtracting the entire image. If done before the satellite pass, detection will be easier, as the difference between the satellite's value and the rest of the image will be higher. We will still face the same problem when changing exposure times.

- Camera's own background subtraction technique:

  We discovered this option late in development, which is why we initially developed our own background subtraction technique. The camera's built-in subtraction outperforms ours, as it averages the first four frames before subtraction. When initiated with the lid on, this technique only reduces the mean pixel value of the image by a 0.04 %, whereas our subtraction reduces it by 10.13 %, as shown in Fig. 3.15.

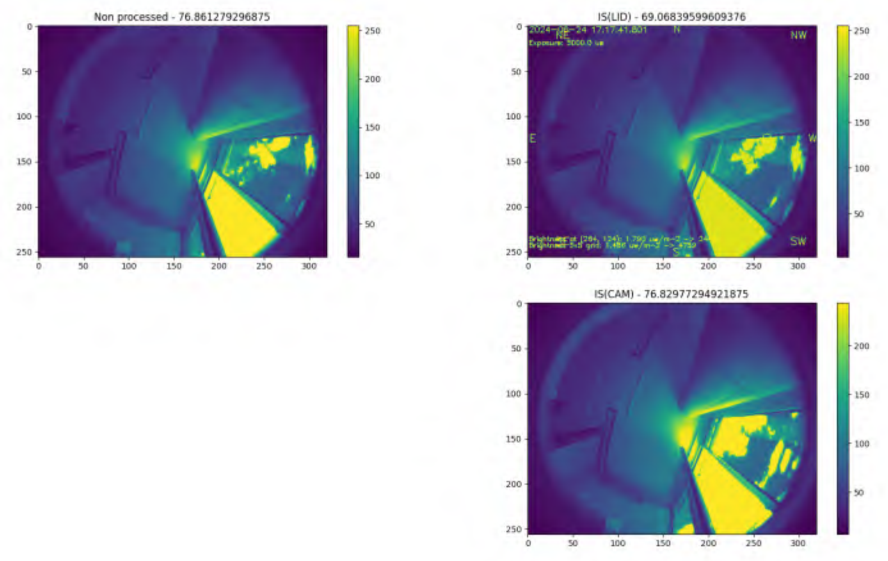


Figure 3.15: Taken images from the camera. Normal image (left). Own background subtraction (upper right). Camera's own background subtraction (lower right).

We do not need to process anything as the camera handles it all. However, the problem with the exposure times remains, besides the intensity values are now more difficult to obtain, as we do not know the subtracted value.

- Normal operation:

  Nothing is removed from the frame under normal operation. We recommend using both this method and the hot-pixel removal technique, as they are the most consistent. While background subtraction is useful in specific scenarios, the normal mode is expected to be used most of the time.

Now that the camera mode has been selected, the next step is tracking the satellite. We chose to avoid inputting the satellite orbit information because we wanted to be able to track any satellite without prior knowledge, discarding local thresholding. Therefore, the most effective approach was to track the brightest point in the image, filtering by minimum and maximum spot size.

We developed two different modes based on lightning conditions. The daytime mode involves applying a bilateral filter followed by an Otsu's thresholding. Bilateral filtering smooths images while preserving their edges as each pixel is replaced by a weighted average of its neighbours. The weighting is dictated by a spatial component that penalizes distant pixels and a range component that penalizes pixels with a different intensity. This combination ensures that only nearby similar pixels contribute to the result. This method is ideal for our application, as we will apply contour detection, where maintaining the edges around the satellite is crucial for distinguishing it from the rest of the image.

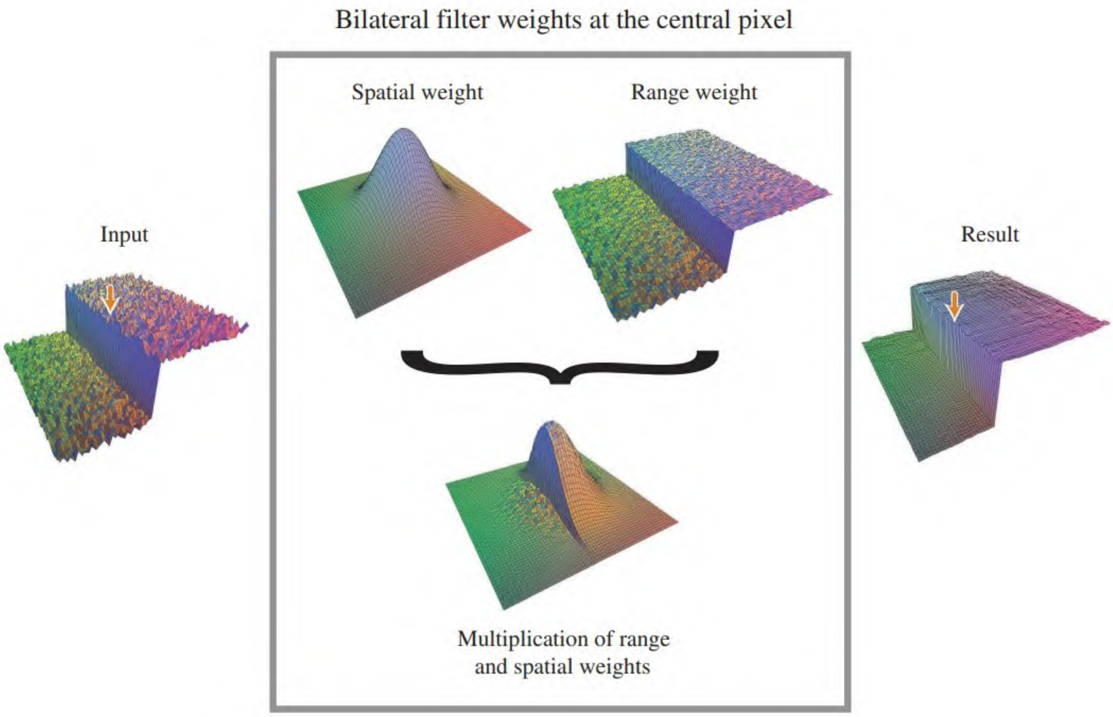


Figure 3.16: Bilateral filter weights. Figure taken from [104], reproduced from [105]

The key idea of the bilateral filter is that for a pixel to influence another pixel, it should not only occupy a nearby location but also have a similar value. Denoted by $BF[\cdot]$, the bilateral filter is defined by equation (3.4) [105]:

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)\, I_{\mathbf{q}} \tag{3.4}$$

Here, $G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)$ represents the spatial weight, a Gaussian function that decreases the influence of distant pixels. $G_{\sigma_r}(|I_\mathbf{p}-I_\mathbf{q}|)$ represents the range weight, a Gaussian function that reduces the influence of pixels $\mathbf{q}$ when their intensity values differ from $I_\mathbf{p}$). $W_\mathbf{p}$ represents the normalization factor, ensuring pixel weights sum to 1:

$$W_\mathbf{p} = \sum_{\mathbf{q}\in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_\mathbf{p}-I_\mathbf{q}|) \tag{3.5}$$

Fig. 3.16 shows how the weights are computed for a pixel near an edge. Fig. 3.17 compares different proposed blurring techniques, with the bilateral filter configured with a diameter of pixel neighborhood of 3 and a σ value of 200 for both the color and coordinate space, performing the best.

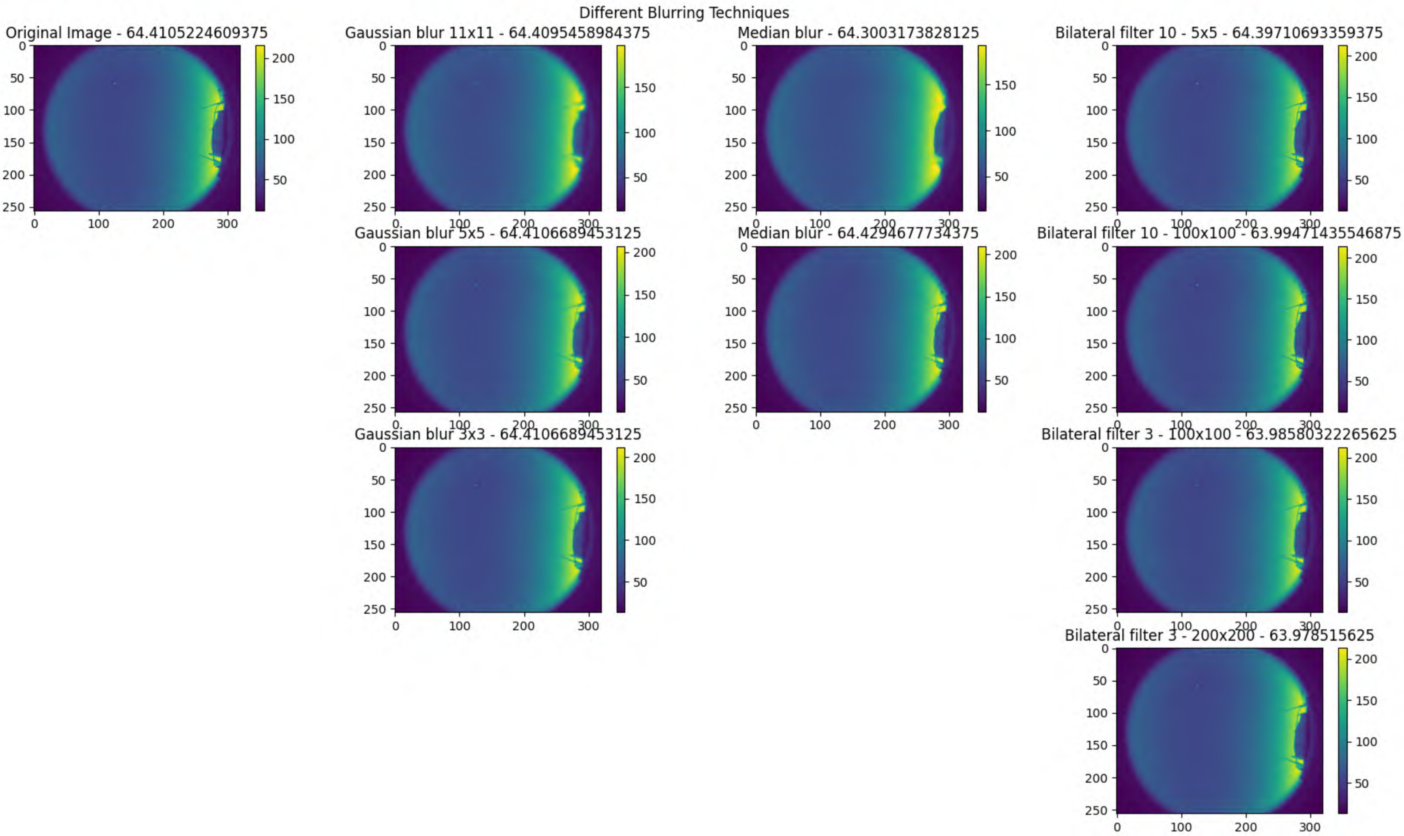


Figure 3.17: Different blurring techniques.

The Otsu's thresholding method, also known as the Otsu's method, was the solution for the changing exposure. When exposure times vary, the mean pixel values also change, which makes standard thresholding techniques unreliable, as they may segment the satellite. We first thought of an adaptive thresholding technique; however, as shown in Fig. 3.18, their performance was suboptimal. The Otsu's method automatically determines the threshold that separates pixels into two classes—foreground and background—as explained in [106].

Fig. 3.18 demonstrates that only the Otsu's thresholding method correctly localizes the satellite under daylight conditions. Fig. 3.19 illustrates the effectiveness of combining the Otsu's thresholding with the previously discussed filters. We apply a filter prior to thresholding because, as shown in the lower left image of Fig. 3.19, using Otsu's thresholding by itself results in the removal of pixels with lower values that are still illuminated by the satellite. By blurring, we average those pixels with the ones with higher values, capturing the full area of the satellite in the image. Among all the filters tested, the bilateral 3 200 × 200 performs best.

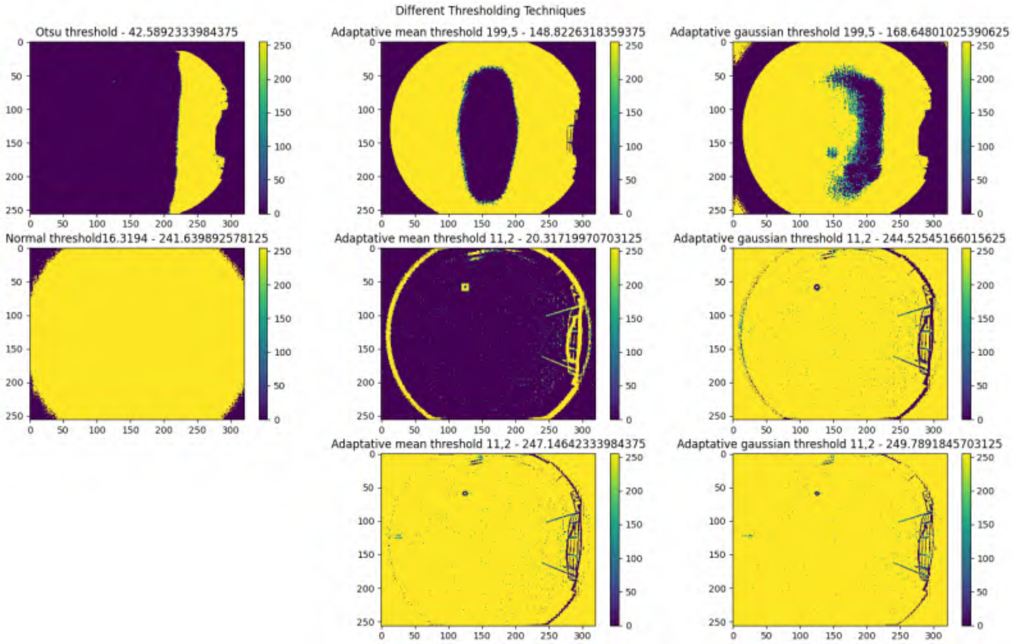


Figure 3.18: Different blurring techniques.

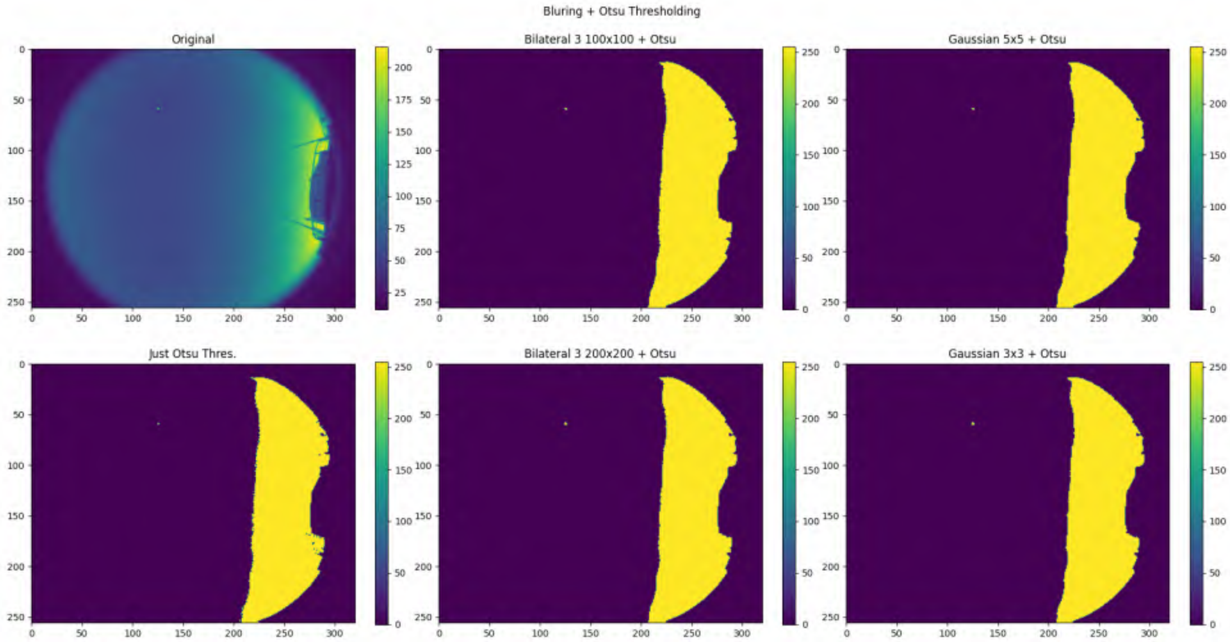


Figure 3.19: Otsu thresholding plus different blurring techniques.

After applying the appropriate threshold, we will obtain the contours of the mask, as shown in Fig. 3.20. These contours will then be filtered by the selected maximum and minimum spot size, based on equations (3.6) and (3.7): obtaining the contour with the maximum total pixel value, as illustrated in Fig. 3.21:

$$height\ of\ contour\ \times\ width\ of\ thecontour\ \geq minimum\ spot\ size\ value \tag{3.6}$$

$$height\ of\ contour\ \times\ width\ of\ the\ contour\ \leq maximum\ spot\ size\ value \tag{3.7}$$

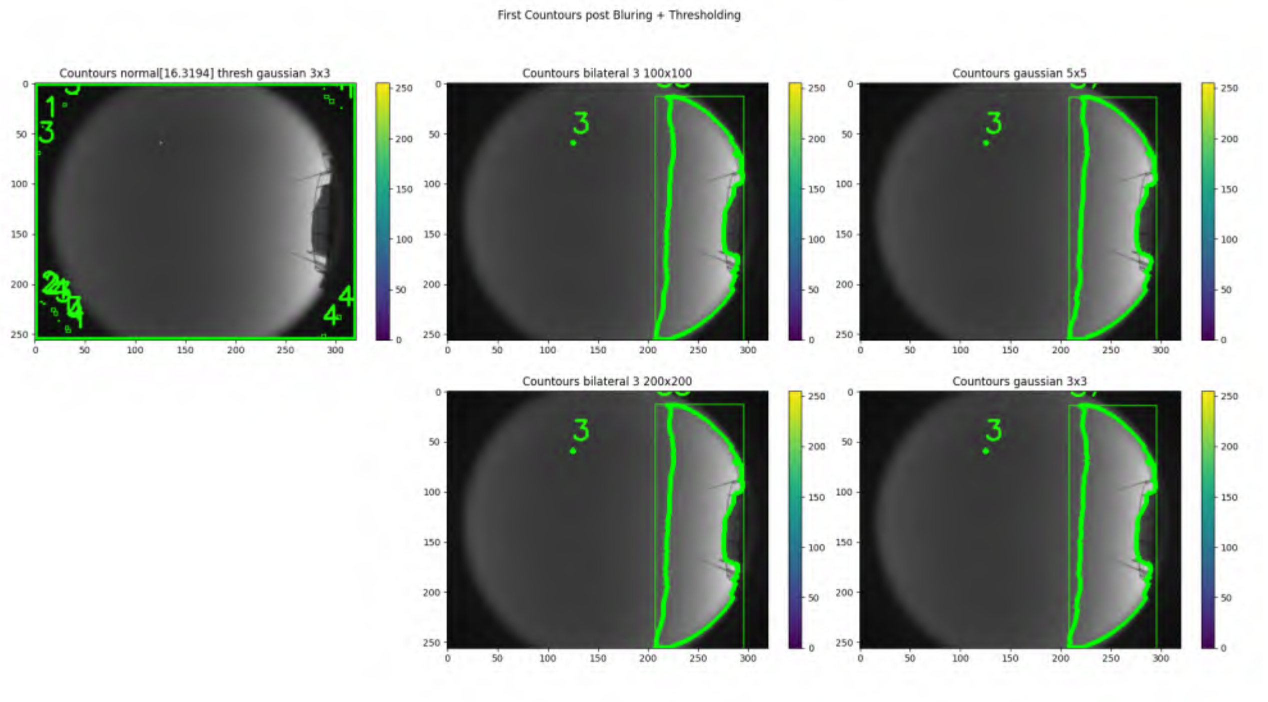


Figure 3.20: Contours of the image obtained by different techniques.

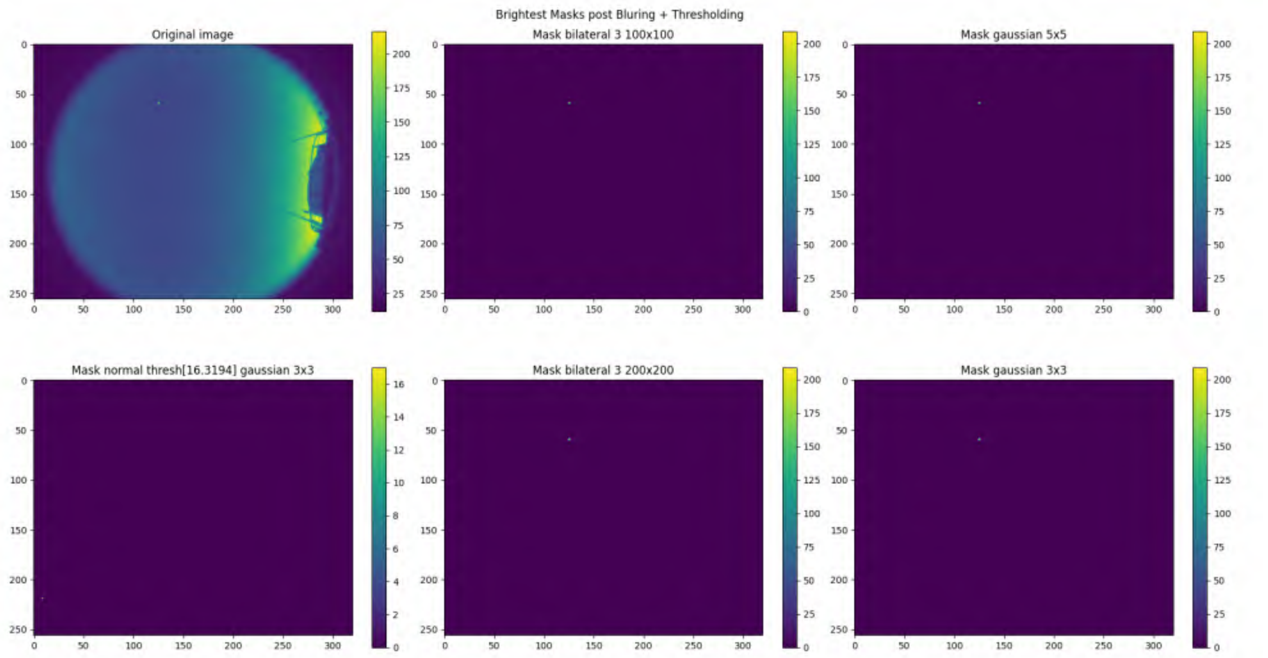


Figure 3.21: Brightest contour of the image by different techniques filtered by spot size value.

From the mask, we extract the location of the brightest contour, which is then overlaid onto the original image, as illustrated in Fig. 3.22. We extract the final values applying this location to the original image: the value of the brightest pixel and the mean and summed pixel value of the entire brightest contour. All these values along with the current date and time, exposure value, minimum and maximum spot size, elevation, azimuth, and the cardinal coordinates, will be overlaid onto the original image."

The process of obtaining the intensity values involves applying a correction factor, the selection of which will be explained in subsection 3.5.4.

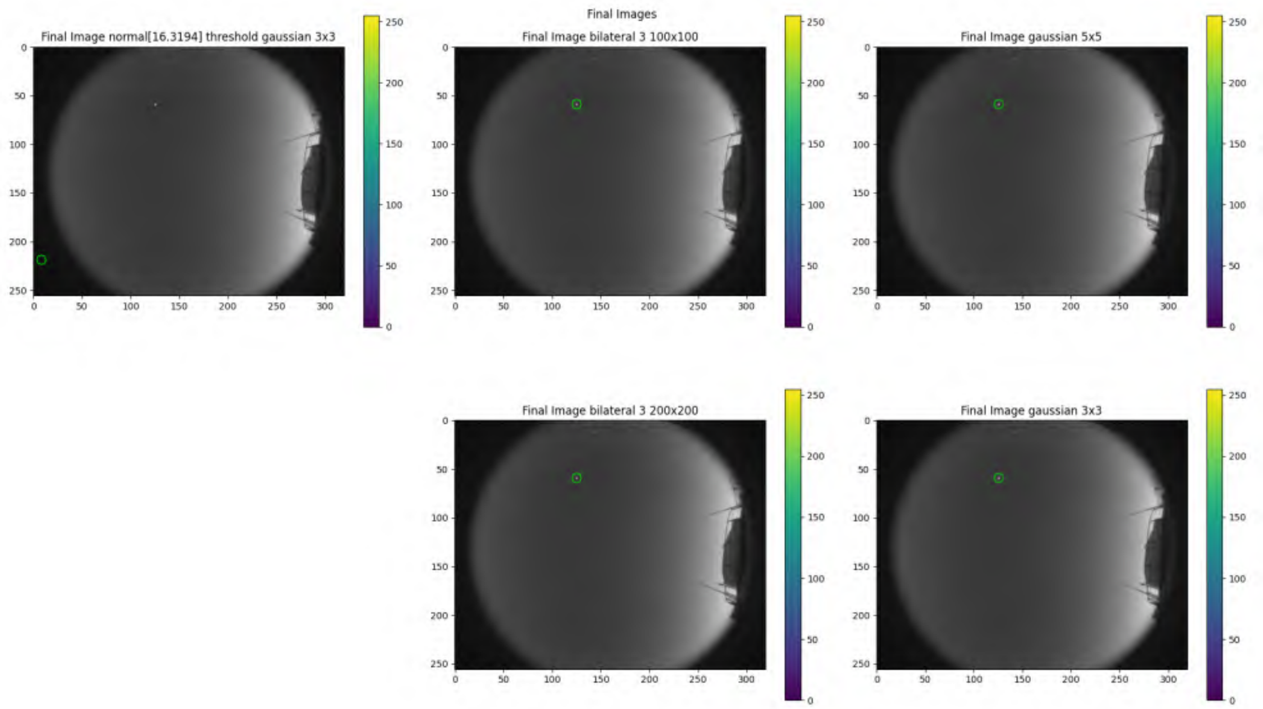


Figure 3.22: Final brightest point of the image by different techniques filtered by spot size value.

When using the night-time mode, the procedure remains same as in daytime, with two differences: the bilateral filter is replaced by a Gaussian 3×3 filter, and the Otsu's thresholding method is substituted with a binary threshold determined by the background noise curve of the camera. At night-time, the mean pixel value is lower, therefore the difference with the satellite is higher, which allows us to threshold the image based on the camera's background noise. To obtain the noise curve, we captured frames with exposure times ranging from 10 μs to 256000 μs, as illustrated in Fig. 3.23

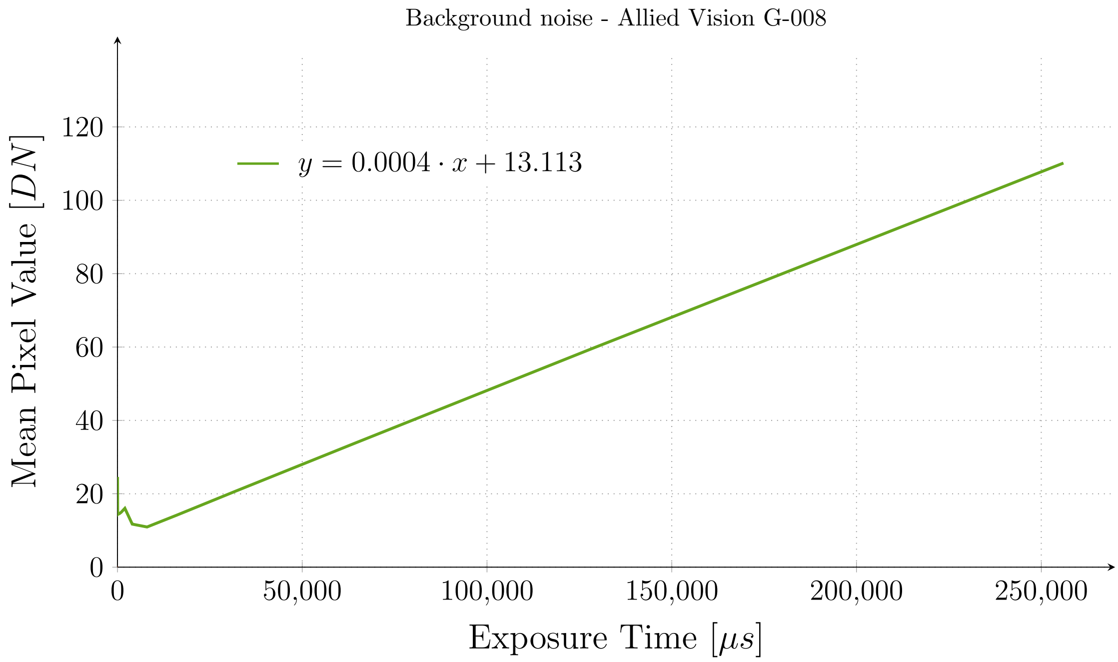


Figure 3.23: Camera's background noise curve of gain1 mode.

Both processed and unprocessed frames will be saved in the following directory with an specific name depending on the date, time, exposure mode and time, camera mode, gain mode and payload:

```
AllSkyCam4OLEODL/data/tracking_images
```

A CSV file containing all the relevant parameters will be saved in the same directory as the processed frames. This file is crucial, as it all the values necessary to assess the performance of the link. The Intensity received - expected $[dB]$ parameter will indicate us how far were we from the estimated value from the link budget:

- Frame Number
- Gain Mode
- Time $[CEST]$
- Exposure $[\mu s]$
- Location [x, y]
- Elevation [°]
- Azimuth [°]
- FOV
- R
- Brightest Pixel Value $[DN]$
- Intensity Brightest Pixel $[\mu W/m^2]$
- Mean Pixel Value of the Frame $[DN]$
- Summed Brightest Contour Pixel Value $[DN]$
- Summed Brightest Contour Intensity $[\mu W/m^2]$
- Payload
- Payload Intensity $[\mu W/m^2]$
- Intensity received - expected $[\mu W/m^2]$
- Intensity received - expected $[dB]$

A graph of the entire recording is also saved in the same directory as the recorded frames. An example of the graph is provided in Fig. 3.24.

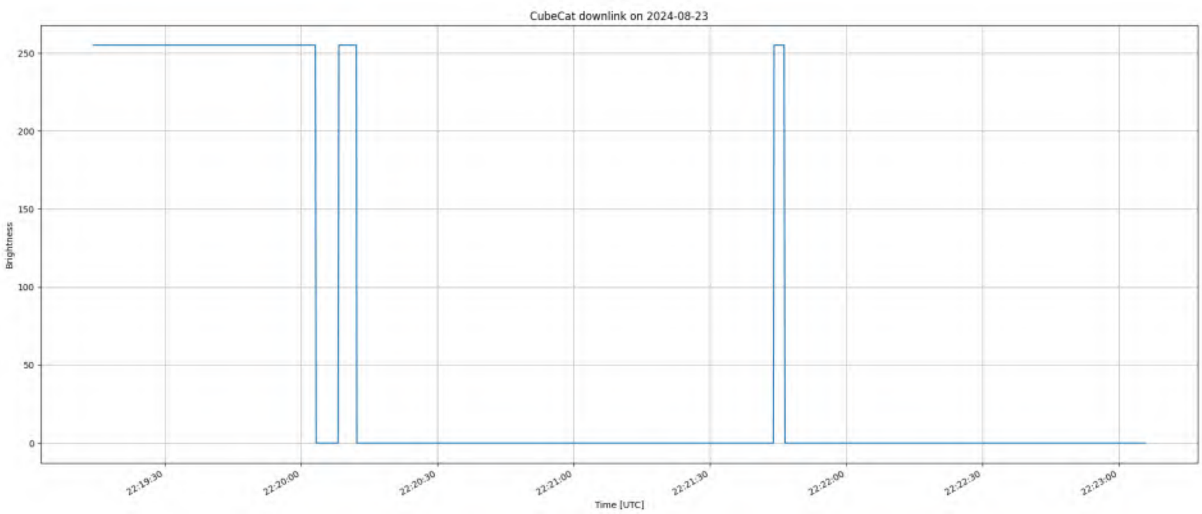


Figure 3.24: Graph of a failed CubeCat downlink.

The y-axis represents the pixel value from the camera, as the intensity is not yet properly calibrated; once calibrated, intensity will be the variable used. The title of the graph is automatically generated based on the payload and current date.

3.4.3 Elevation and Azimuth

Elevation and azimuth are angular measurements to identify the position of a satellite relative to an observer location. Measured both in degrees, they start from 0 degrees: azimuth starts from the north and covers 360 degrees clockwise, being 90° east, 180° south and 270° east; elevation goes from horizon (0°) until zenith at 90°. To accurately calculate these angles, it is necessary to know the focal length of the lens, its mapping projection and the center coordinates of the capture image.

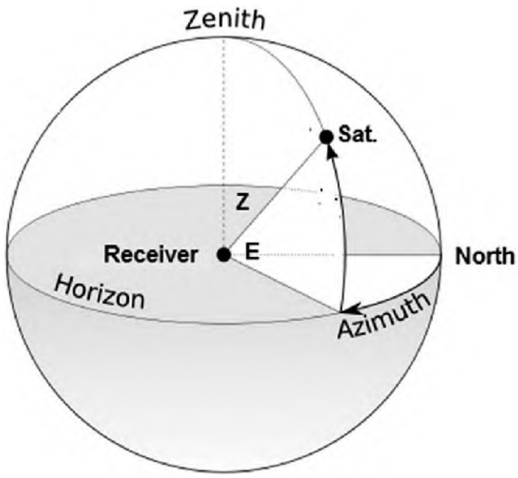


Figure 3.25: Diagram of the elevation and azimuth of an object. Figure taken from [107]

In this work, we use wide angle or fisheye lenses, this type of lenses provide a very wide FOV by distorting the image. This distortion is achieved through an specific mapping function, referred as a projection. There are several projection as illustrated in Fig. 3.26, but we will focus in the most used, the equisolid projection.

projection	math
equidistant fisheye	$R = f \cdot \theta$
stereographic	$R = 2f \cdot \tan\left(\frac{\theta}{2}\right)$
orthographic	$R = f \cdot \sin(\theta)$
equisolid (equal-area fisheye)	$R = 2f \cdot \sin\left(\frac{\theta}{2}\right)$
Thoby fisheye	$R = k_1 \cdot f \cdot \sin(k_2 \cdot \theta)$ with $k_1 = 1.47$ and $k_2 = 0.713$
PTGui 11 fisheye	$R = \begin{cases} \frac{f}{k} \cdot \tan(k \cdot \theta) & \text{for } 0 < k \leq 1 \\ f \cdot \theta & \text{for } k = 0 \\ \frac{f}{k} \cdot \sin(k \cdot \theta) & \text{for -1} \leq k < 0 \end{cases}$

Figure 3.26: Different types of fish-eye lens mapping functions. Figure taken from [108]

Let θ represent the angle in radians between a point in the real world and the real axis, f denotes the focal length of the lens, and R is the radial distance in the image (the distance from the center to an specific point). In a circular lens, θ can be interpreted as the field of view of the lens. For any point within a given frame, all points at the same radius with respect of the center will share the same

θ. Thus, θ can be considered as the FOV that allows observation of that specific circle, defined by equation (3.8).

$$\theta = 2 \arcsin \left(\frac{R \cdot P_{size}}{2f} \right) \tag{3.8}$$

Where P_{size} is the pixel size in $mm/pixel$ and the radial distance R is calculated as $R = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ $[pixels]$.

Our final field of view and elevation angle will be defined by equations (3.9) and (3.10), respectively.

$$\theta_{deg} = 2\theta \cdot \left(\frac{180}{\pi} \right) \tag{3.9}$$

$$elevation = \frac{(180 - \theta_{deg})}{2} \tag{3.10}$$

To calculate the azimuth angle, we will define θ as the angle measured in the clockwise direction from the origin of the north line A, to the object B, as illustrated in Fig. 3.27.

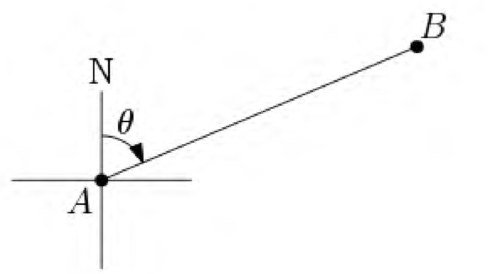


Figure 3.27: Azimuth diagram of an object. [109]

Then the coordinates of point B can be described by equation (3.11):

$$(b_1, b_2) = (a_1 + r \sin \theta, a_2 + r \cos \theta) \tag{3.11}$$

Where r is the length of the line segment AB. θ is therefore defined by equation (3.12):

$$\theta = \arctan \left(\frac{b_1 - a_1}{b_2 - a_2} \right) \tag{3.12}$$

The frame is divided into three quadrants: one for the bottom section of the image and two dividing the upper section in half. It is important to note that the origin is at the upper left corner of the image. Additionally, the images captured by the camera are mirrored, the left side of the image corresponds to the west side, and vice versa. This is clearly illustrated in Fig. 3.28.

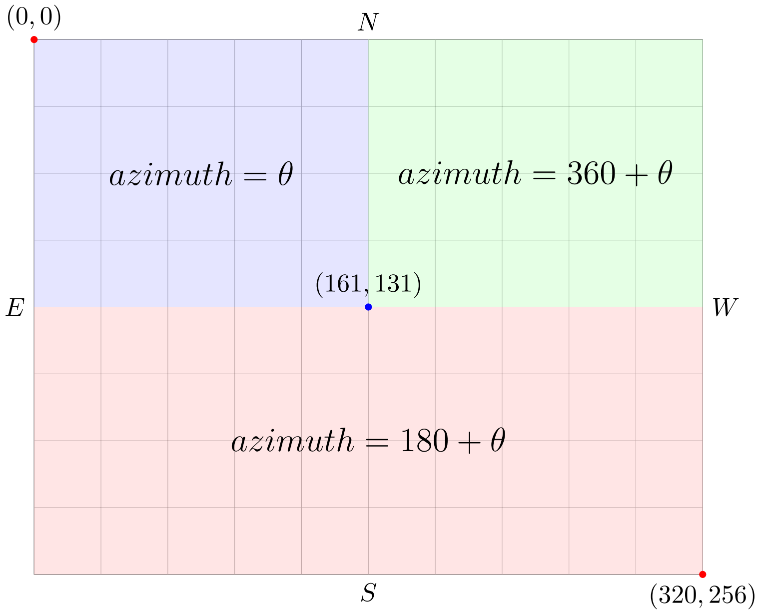


Figure 3.28: Diagram of the calculation of the azimuth for the different quadrants of a picture.

3.4.4 Graphical User Interface

When executed, the program displays a Graphical User Interface (GUI) menu, allowing the user to choose the settings for both the camera and link budget. The camera settings include the gain mode, capture mode, lightning mode, camera mode, exposure mode and exposure time. For the link budget, the user can adjust the payload, the height of the Optical Ground Station (OGS), the zenith attenuation, the elevation mode and the elevation angle.

When capturing is initiated, three new windows appear on screen. The one on the bottom right is tasked with controlling the exposure time, and the minimum and maximum spot size value—the sliders use a logarithmic scale, making it easier to select lower values. The window on the bottom left, displays a live graph, similar to the one shown in Fig. 3.24, which plots time against the mean pixel value in real-time. The final window on the top left, shows both the processed and unprocessed frames from the camera. An image of an actual operation of the system is displayed in Fig. 3.29.

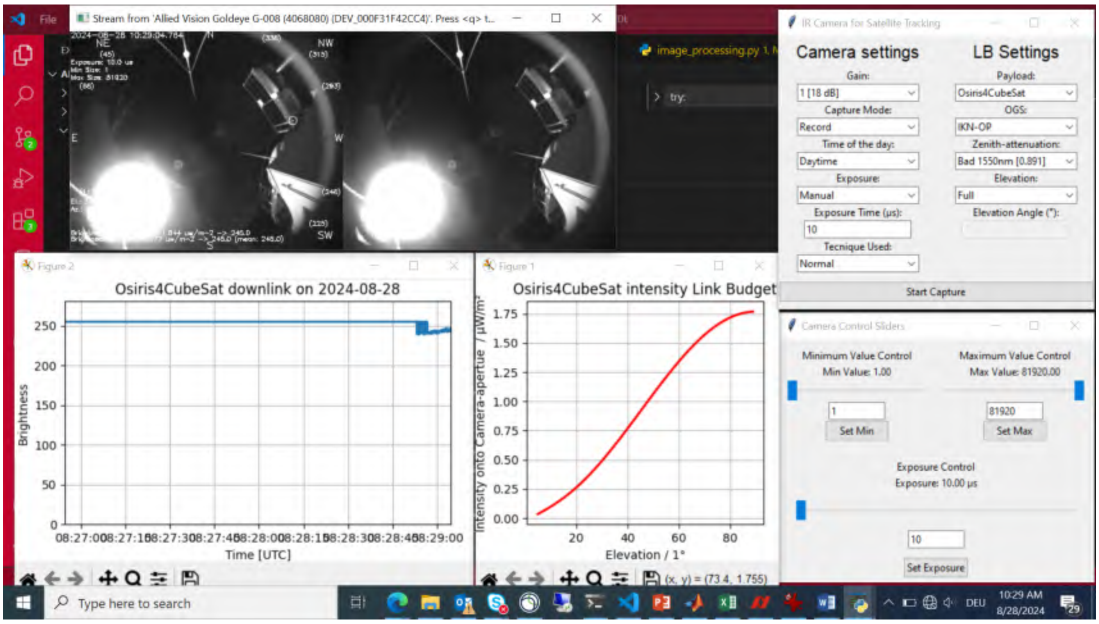


Figure 3.29: Real time system operation.

3.5 Testing

In this section we will go over all the executed tests to ensure that we meet all the necessary requirements of the final system.

3.5.1 Calculation of the Link Budget for the Observed Satellites

The first we challenge we faced was the complexity of the satellite's visibility. We required an intensity value to determine whether the camera could detect satellite. We chose the OSIRISv1 onboard Flying Laptop (FLP), described in subsection 2.5.1, because it is one of the dimmest satellites that we will test with the AllSky camera system.

Using the procedure outlined in subsection 2.4.5, we will perform the link budget analysis for the laser terminal. Since we are using a camera, few exceptions need to be considered: The optical losses inside the receiver terminal, a_{Rx}, are disregarded as we want to measure the intensity before it goes through the camera; beam wander losses, or the average loss due to dynamic beam miss-pointing, a_{BW} are also ignored, as our camera is able to see the whole hemisphere, ensuring that the satellite will be within the field of view of the receiver, regardless of pointing accuracy; finally, scintillation losses a_{Sci} will be negligible. As explained in subsection 2.4.4, using a long exposure time (around 100000 μs) will average the scintillation index over time.

For our calculations, we will use 15° as the minimum elevation angle visible:

Table 3.3: Link Budget from the observed satellites.

Parameter (formula)	**OSIRISv1 on Flying Laptop [64]:** *595km polar orbit, 1.0 mrad FWHM Tx-div., 1W Tx-power of $\lambda = 1545$ nm into the 2.5mm ∅ effective aperture of the InGaAs on the IKN*		**OSIRIS4CubeSat on CubeL [80]:** *560km polar orbit, 120 µrad FWHM Tx-div., 85mW Tx-power of $\lambda = 1550$ nm into the 2.5mm ∅ effective aperture of the InGaAs on the IKN*		**CubeCat on NORSAT-TD:** *455km polar orbit, 104 µrad FWHM Tx-div., 300mW Tx-power of $\lambda = 1545$ nm into the 2.5mm ∅ effective aperture of the InGaAs on the IKN*	
	15° elevation	**zenith**	**15° elevation**	**zenith**	**15° elevation**	**zenith**
Mean source power p_Tx	+30 dBm	+30 dBm	+19.29 dBm	+19.29 dBm	+24.77 dBm	+24.77 dBm
Tx internal losses a_Tx	-1 dB	-1 dB	NA	NA	NA	NA
Tx antenna gain g_Tx (2.13)	+70.4 dB	+70.4 dB	+88.9 dB	+88.9 dB	+90.1 dB	+90.1 dB
Pointing loss a_{BW} (2.14)	NA	NA	NA	NA	NA	NA
Distance L (2.16)	1613 km	594 km	1538 km	559 km	1303 km	454 km
Freespace loss a_{FSL} (2.15)	-262.4 dB	-253.7 dB	-261.9 dB	-253.1 dB	-260.5 dB	-251.4 dB
Atmos. attenuation a_{Atm} (2.17)	-1.94 dB	-0.50 dB	-1.94 dB	-0.50 dB	-1.94 dB	-0.50 dB
Scintillation loss a_{Sci}	NA	NA	NA	NA	NA	NA
Rx antenna gain g_{Rx} (2.19)	+74.1 dB	+74.1 dB	+74.1 dB	+74.1 dB	+74.1 dB	+74.1 dB
Power into camera's aperture p_{Rx}	-90.7 dBm	-80.6 dBm	-81.6 dBm	-71.4 dBm	-73.4 dBm	-62.9 dBm
Rx-internal losses and signal splitting for tracking a_{Rx}	NA	NA	NA	NA	NA	NA
Intensity into camera's aperture	**0.1723** $\mu W/m^2$	**1.7679** $\mu W/m^2$	**1.407** $\mu W/m^2$	**14.819** $\mu W/m^2$	**9.228** $\mu W/m^2$	**105.64** $\mu W/m^2$
Link margin for communication	-31.7 dB	-21.6 dB	-22.6 dB	-12.4 dB	-14.4 dB	-3.8 dB

To obtain the Intensity into the camera's aperture $[W/m^2]$, we transformed the Power into camera's aperture p_{Rx} (in dBm) and divided the resulting value by the effective area of the camera's aperture as shown in equation (3.13):

$$\frac{10^{\frac{p_{Rx}}{10}} \cdot 10^{-3}}{area_{Rx}} \tag{3.13}$$

Where the effective area of the camera $area_{rx}$ is given by the area of a circle:

$$area_{Rx} = \pi \cdot \left(\frac{\varnothing_{ap}}{2}\right)^2 \tag{3.14}$$

Being $\varnothing_{ap}$ the diameter of the effective aperture of the camera, $2.5{\cdot}10^{-3}$ m, in our case.

The intensity requirements are strict—the link margin does not affect us, as we do not want to establish any communication. Our goal is to visualize the satellite and assess weather observed intensity is comparable to the value calculated under "Intensity into camera's aperture" in the link budget. After recording, we will calculate the difference in decibels at each elevation angle of the satellite.

3.5.2 Evaluation of the Lenses

As explained in subsection 3.2.2, the two major factors to consider for the lenses are the Field Of View (FOV) and transmittance at the desired wavelength of 1550 nm. The transmittance is already suboptimal: it is below 45 % for the Fujinon lens at 1000 nm, and around 60 % for the Thorlabs lens at 1400 nm. The field of view values are better: the FE185C057HA-1 maintains its 185° FOV, as it is compatible with our 2/3" sensor; on the contrary, the MVL4WA is only compatible with sensors up to a 1/2" size (we should obtain around 140° FOV with a 2/3" sensor based on our calculations). The purchase of the Thorlabs MVL4WA was risky because using a lens not suitable for a certain sensor could lead to the presence of vignetting, unexpected distortions or even a reduction in the camera's field of view. We tested the lens to verify its correct functionality.

We took two pictures to estimate the field of view: one with the subject positioned at the center of the frame, and the other with the subject at the edge of the field of view, illustrated in Fig. 3.30. The used setup is shown in Fig. 3.31.

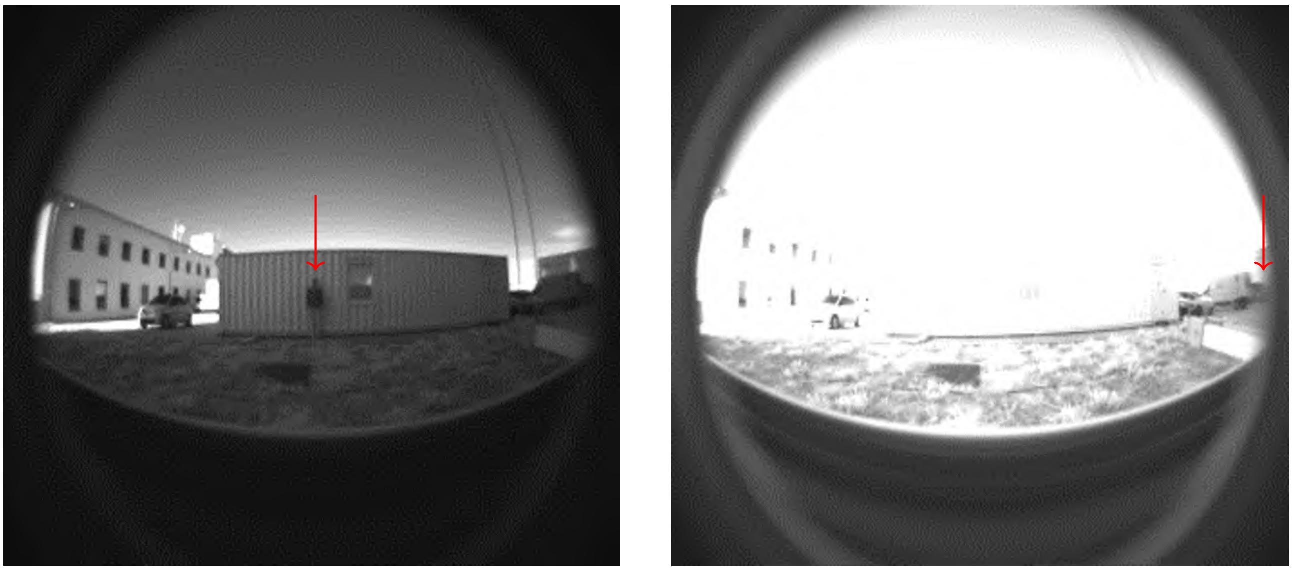


Figure 3.30: Subject in the middle of the frame. (left). Subject on the right edge of the frame. (right)

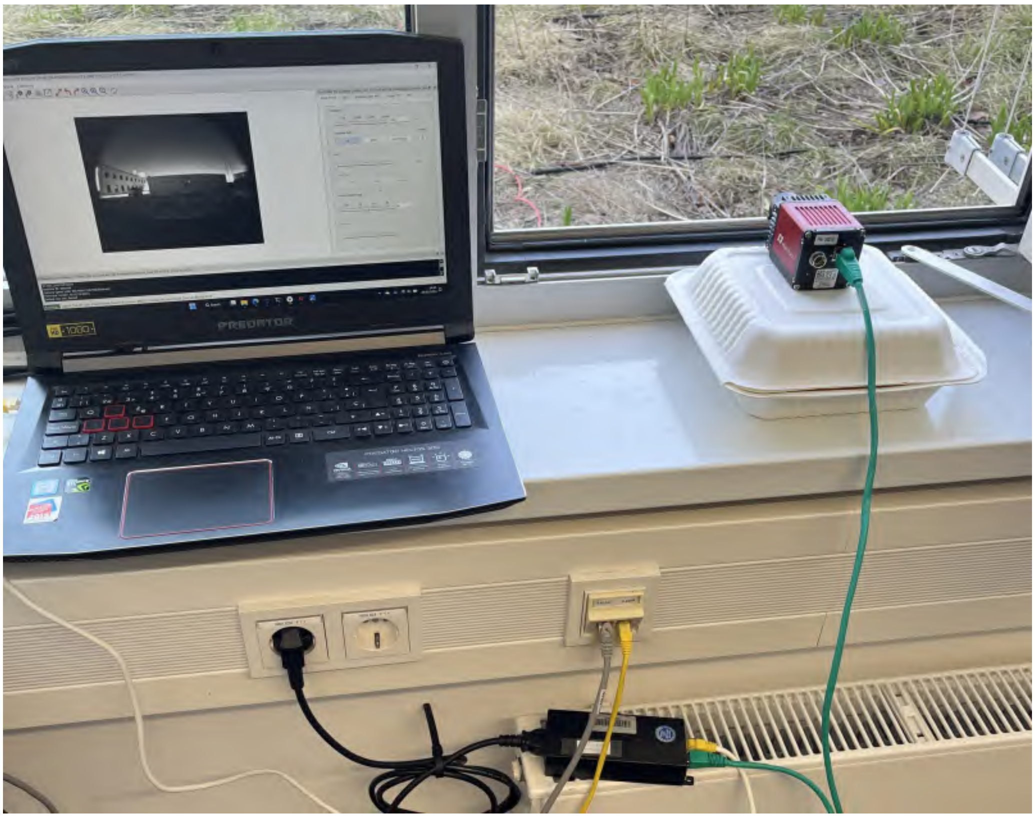


Figure 3.31: Setup of the first FOV experiment.

In the first image, the subject is positioned at 7.42 m from the camera, while in the second image, the subject is 19.32 m away from its initial position. Using basic trigonometry, the field of view of the lens can calculated, as shown in equation (3.15):

$$FOV[°] = \arctan \frac{19.32m}{7.42m} = 68.99° \Rightarrow 137.98° \tag{3.15}$$

To confirm these results, we used a Pan-Tilt-Unit (PTU). By selecting a reference point in the center of the frame and panning the PTU along the x-axis until the point reaches the edge of the frame, we precisely determine the camera's field of view, as illustrated in Fig. 3.32.

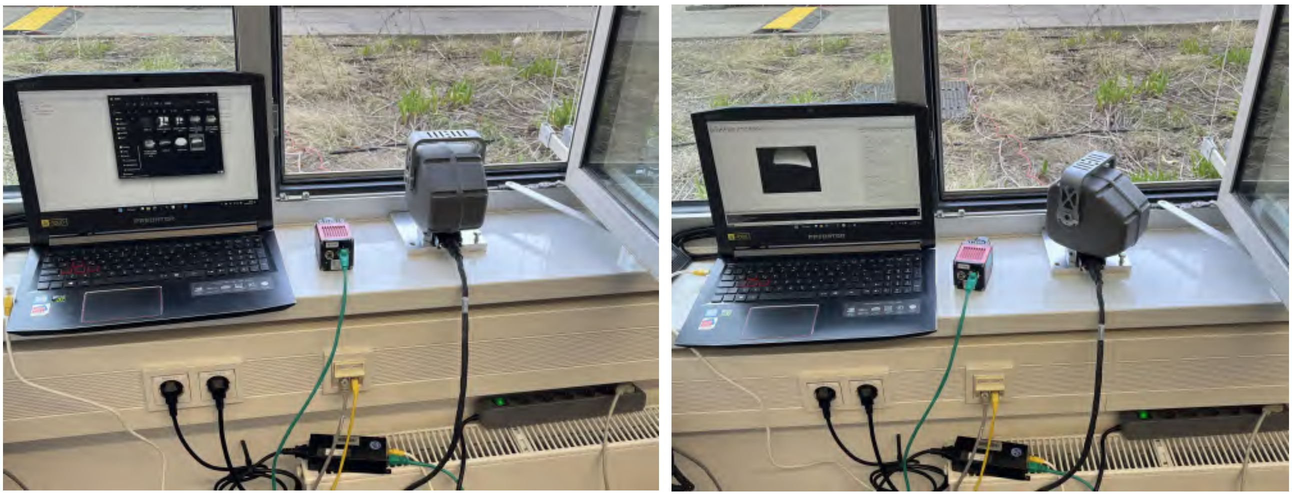


Figure 3.32: PTU pointing to 0° (left). PTU pointing to 70° (right).

Using the right edge of the sewer as a reference, as shown in Fig. 3.33, we confirm that the camera's field of views approximately 140 degrees, as estimated.

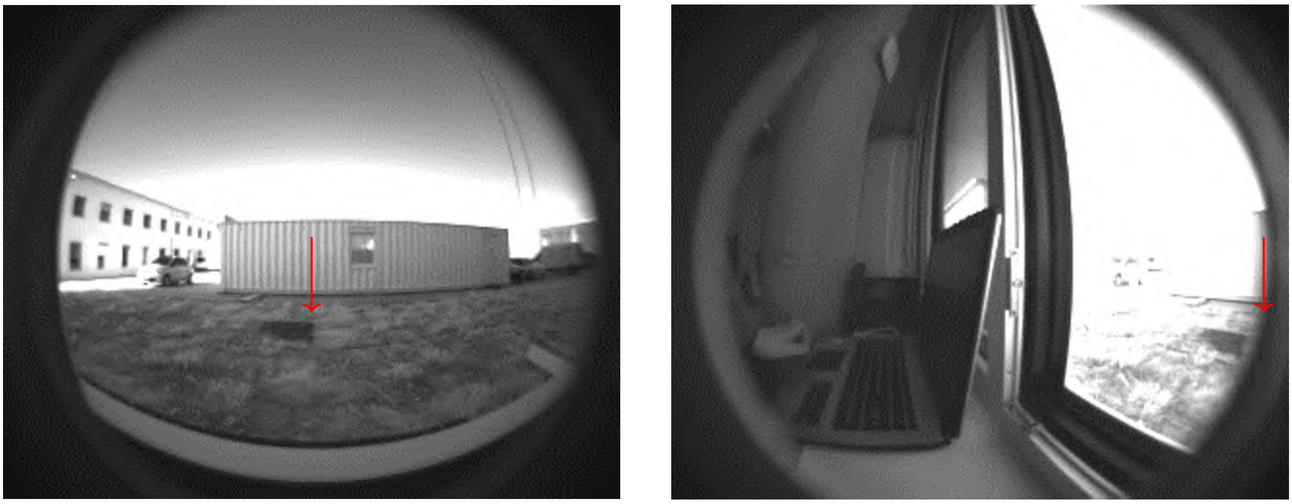


Figure 3.33: Reference point in the middle of the frame. (left). Reference point on the right edge of the frame. (right)

The primary issue with the Fujinon FE185C057HA-1, as noted in Table 3.2, is its fixed focus, which results in blurry images. This is a major drawback when attempting to evaluate laser intensity. We tried adjusting the focus using spacer rings, but it was unsuccessful as it was impossible to screw the lens in a consistent way. This lead to to instability and inconsistency in the focus, which rendered our results unreliable. Summed to the low transitivity of the lens made us disregard the Fujinon lens.

3.5.3 Intensity Measurement with a Coarse Wavelength Division Multiplexing Transceiver

After calculating the required minimum intensity and checking the correct functioning of the lens, we checked whether the system could detect Flying Laptop at 15-degrees of elevation, our worst-case scenario. We used a 1.6 mW Coarse Wavelength Division Multiplexing (CWDM) single-mode transceiver, centered at 1550 nm, connected to a Single Mode Fiber (SMF) patch, to simulate the laser signal. This approach provided an affordable and efficient method to validate the intensity received by the camera.

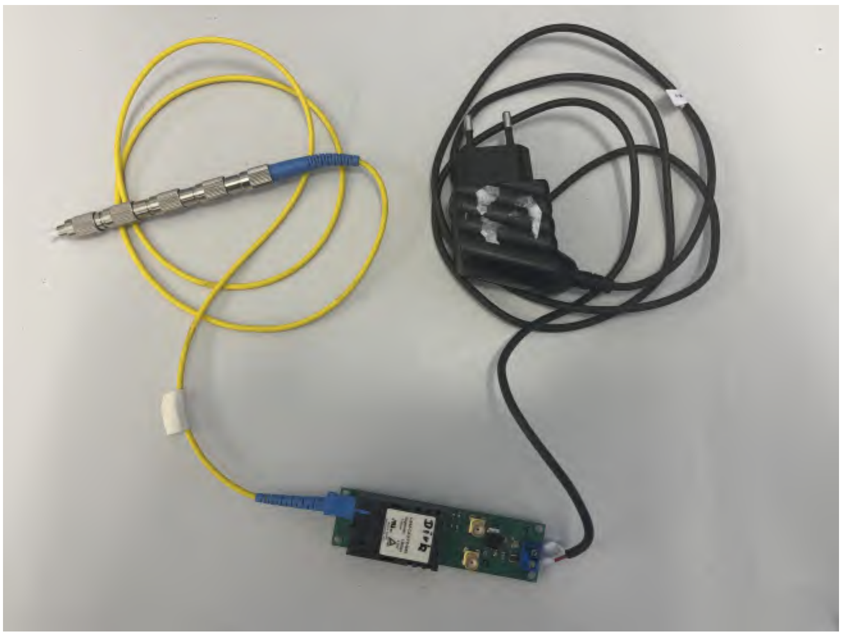


Figure 3.34: LS42-CAU-TC-N55 CWDM transceiver.

With the aforementioned transceiver, we only needed the Full Width Half Angle (FWHM) to assess the distance required to setup the laser away from the camera, so the axial intesity received replicates the one we would receive from flying laptop at 15° elevation angle.

$$I_0(L) = \frac{4\ln 2}{\pi} \cdot \frac{P_0}{(L \cdot 1.18 \cdot \theta_{\text{SM}})^2} \tag{3.16}$$

The axial intensity does not account for atmospheric or pointing losses and assumes a Gaussian far-field beam pattern [64]. Our main unknown, θ_{SM}, is defined by equation (3.17):

$$\theta_{\text{SM}} = M^2 \frac{\lambda}{\pi w_0} \tag{3.17}$$

The divergence or acceptance angle θ_{SM}, is defined as the point with $1/e^2$ times the maximum intensity of a Gaussian beam. However, in telecommunications, the full width half angle is more commonly used. This full beam divergence angle is 1.18 times the half-angle divergence [110], as shown in equation (3.17).

The full width half angle is highly dependent on the fiber used—we connected the transmitter side of the transceiver to a SMF-28 fiber patch [111]. The beam quality factor, or M^2 parameter, is defined as the ratio of the beam's divergence to that of an ideal fundamental Gaussian beam. Since the fundamental Gaussian beam has the least divergence, real beam always follow $M^2 > 1$ (1.123 for the fundamental mode LP_{01} [112]). The beam waist w_0, described in subsection 2.4.4, can be approximated to half of the Mode Field Diameter (MFD) for single-mode fibers, as the beam profile at the fiber output is identical to that within the fiber [113], [114].

$$w_0 = \frac{MFD}{2} \tag{3.18}$$

With all necessary parameters, we calculated the required distance using equation (3.19):

$$L = \sqrt{\frac{4\ln 2 \cdot P_0}{\pi \cdot (1.18 \cdot \theta_{\text{SM}})^2 \cdot I_0(L)}} \tag{3.19}$$

Where:

- $\theta_{\text{SM}} = 1.1^2 \frac{1550 \cdot 10^{-9}\ m}{\pi \frac{10.5 \cdot 10^{-6}\ m}{2}} = 0.1137\ rad$
- $I_0(L) = 0.17232\ \mu W/m^2$
- $P_0 = 1.6\ mW = 1600\ \mu W$

The required distance for the test is $L = 674.71\ m$, an unachievable distance, as the test was intended to be conducted in the basement of the Institute of Communications and Navigation IKN. To address this, we used different attenuators to lower the power, and therefore the distance, as illustrated in Fig. 3.35.

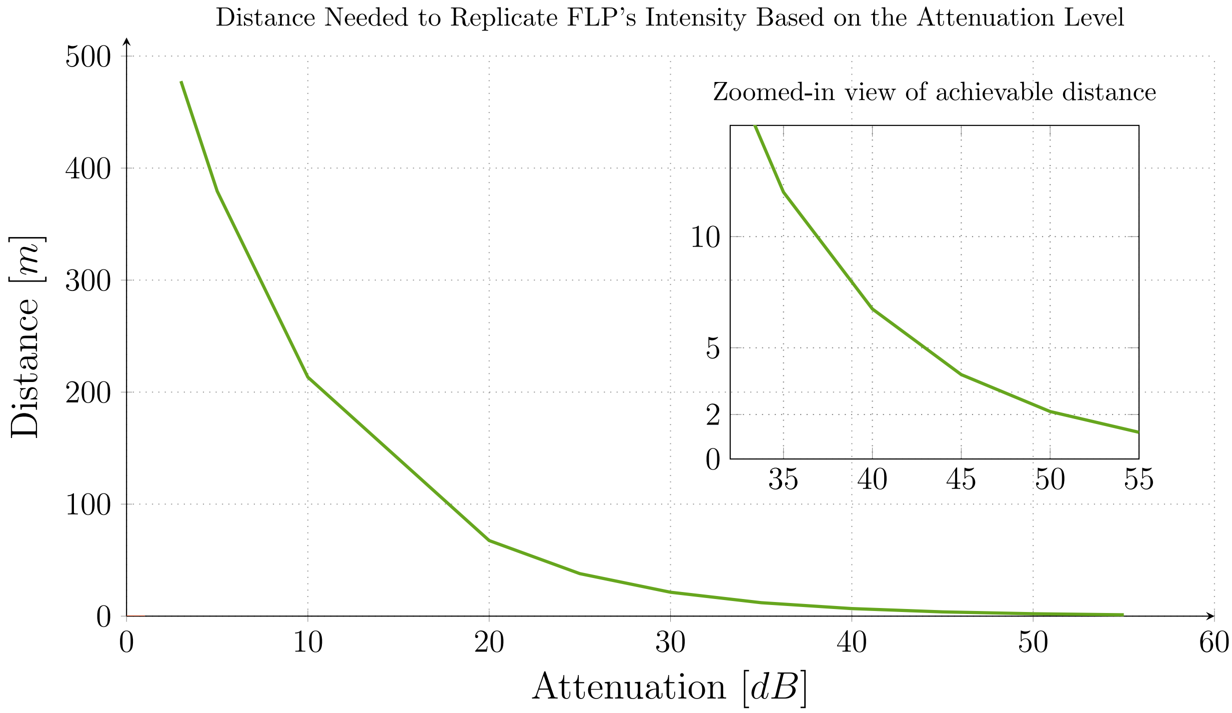


Figure 3.35: Attenuation for achievable distance.

The room used for the experiment has a length of 5 m, so we decided to use 50 dB of attenuation. The laser was positioned 2.65 meters away from the camera, farther than the 2.13 m required to achieve the target intensity. As seen in Fig 3.36, the transceiver was visible with the Thorlabs lens, meaning that the satellite would also be detectable. This was not the case with the Fujinon lens, providing another reason to discard it.

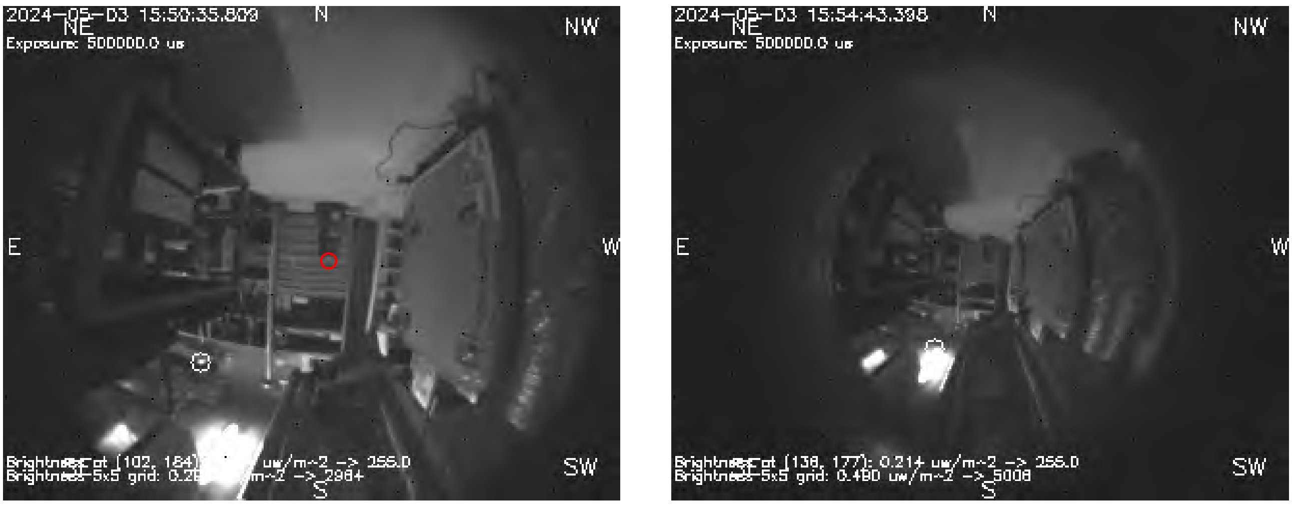


Figure 3.36: Thorlabs lens taken frame, laser not tracked as software was not ready yet. (left). Fujinon lens taken frame. (right)

We measured the power on the fiber tip using a power-meter, registering a value of 0.02 μW. According to equation (3.16), this equaled to an axial intensity of 0.139 W/m^2 at the camera. We used this value to calibrate the system—the system captures frames, which are measured by the digital number of their pixel values, to convert them to intensity we need to apply a conversion—multiplying the brightest pixel value by the obtained axial intensity, and then divided it by the curve generated from taking pictures of the spot when it was undetectable by the camera until saturation (255 pixel value), as illustrated in Fig. 3.37.

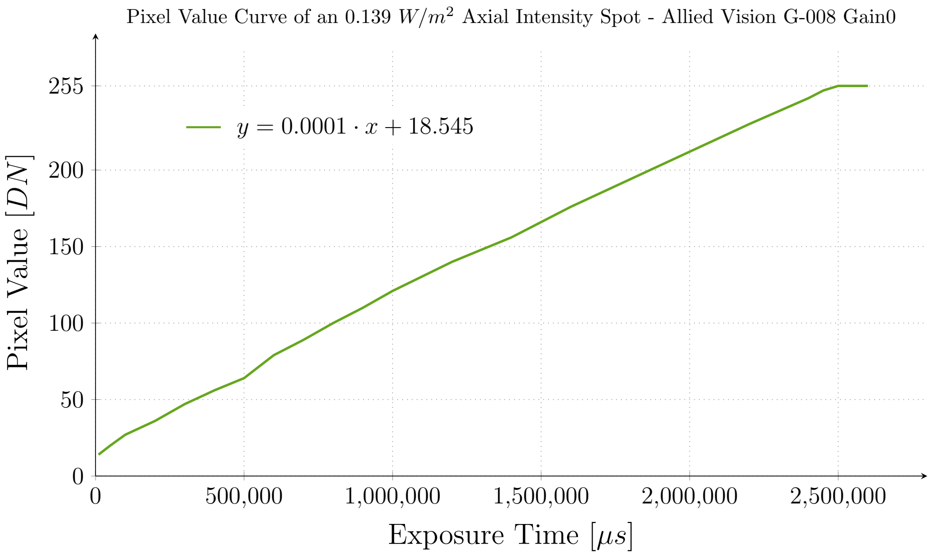


Figure 3.37: Camera´s curve for an 0.139 W/m^2 axial intensity in gain 0 mode.

While this method allowed us to estimate the correct intensity regardless of the exposure time, it raised two main issues. First, we had no way to precisely point the laser to the camera, leading to loses in the detected intensity with no way to account for them. Second, the calibration factor was specific for an intensity value similar to the one used, meaning that only similar intensity values would be estimated accurately. The next experiment aimed to fix these issues.

3.5.4 Camera Calibration and Intensity Assessing with the Radio Tower

At the Deutsches Zentrum für Luft- und Raumfahrt (DLR), we have equipped a radio tower with a 2 mW, 1550 nm laser (with a minimum power output of 0.79 mW), an Erbium-Doped Fiber Amplifier (EDFA); and a Pan-Tilt Unit (PTU), all controlled via DLR servers. This setup enabled us to replicate the previous experiment, this time conducted on the IKN rooftop, aiming the laser with an angular precision of one degree. The setup is identical as the one used by a previous student at DLR [115], replacing the optical ground station focal assembly (SOFA with our camera positioned besides the OGS-OP. The presence of the Pan-Tilt unit in the setup solved the pointing problem, allowing us to precisely direct the laser. The calibration problem was harder to solve. A diagram of the experiment is shown in Fig. 3.38.

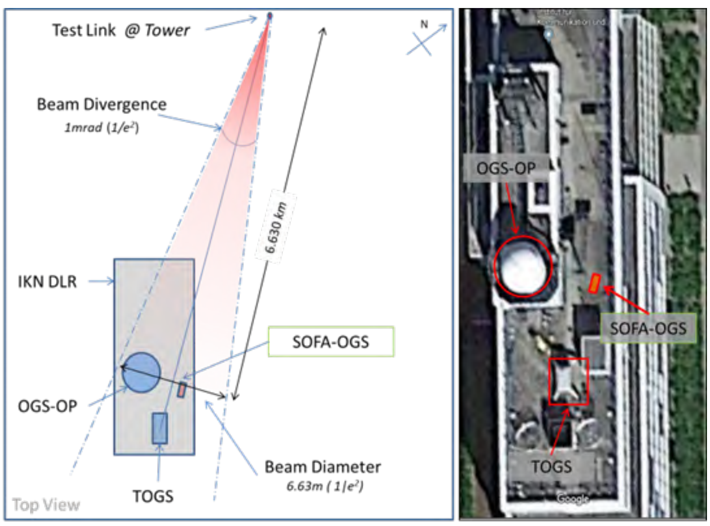


Figure 3.38: IKN rooftop experiment viewed from above

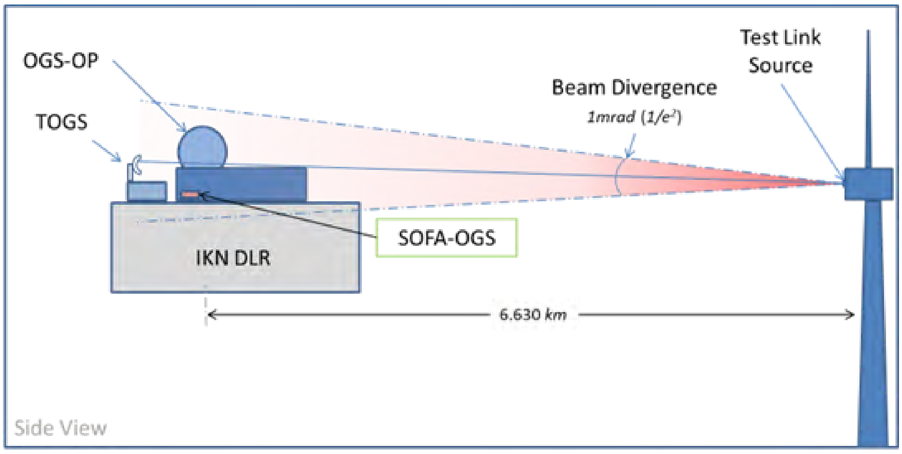


Figure 3.39: IKN rooftop experiment viewed from the side.

The tower is located 6.6 km away from the OGS-OP with a full width half angle of 1 $mrad$. Based on these parameters and equation (3.16), the lowest intensity generated was 15.95 $\mu W/m^2$, much higher than the laser we were using in the basement. This always saturated the camera, regardless of the exposure time. While adjusting the laser's pointing was a solution, it reintroduced the original pointing problem.

We were unable to calibrate the camera using this method. The solution is either to install a filter on the radio tower or use a laser with adjustable power, fixed on a movable mount. The correct procedure for calibrating the camera is detailed in [116]. This method is similar to what we applied in subsection 3.5.3, but in this case, the exposure remains fixed, while the power is changed from undetectable levels until saturation, allowing us to obtain an accurate calibration curve for the entire exposure range, as shown in Fig. 3.40.

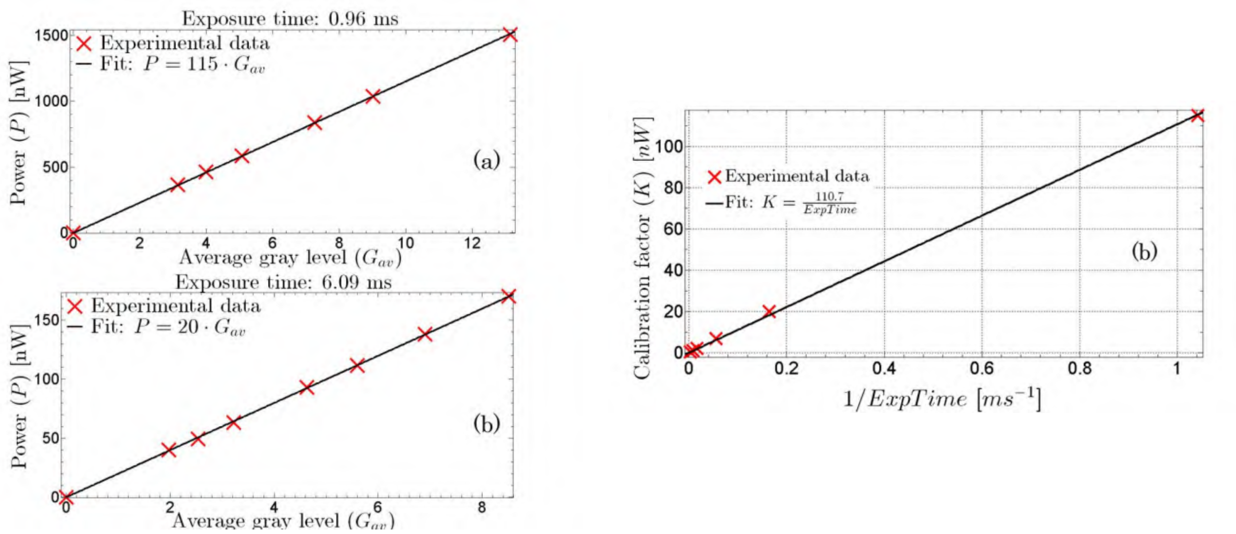


Figure 3.40: Exposure curves with moving power (left). Final exposure curve (right). Figure taken from [116].

A picture of the the saturated result are found next in Fig. 3.41. Pictures of the PTU menu and the setup of the experiment are shown in Fig. 3.42 and Fig. 3.43 and 3.44, respectively. The pathing from both the IKN and GSOC is shown in Fig. 3.45 and 3.46.

Figure 3.41: Picture of the radio tower taken by the camera using gain 1 mode at 0 dBm

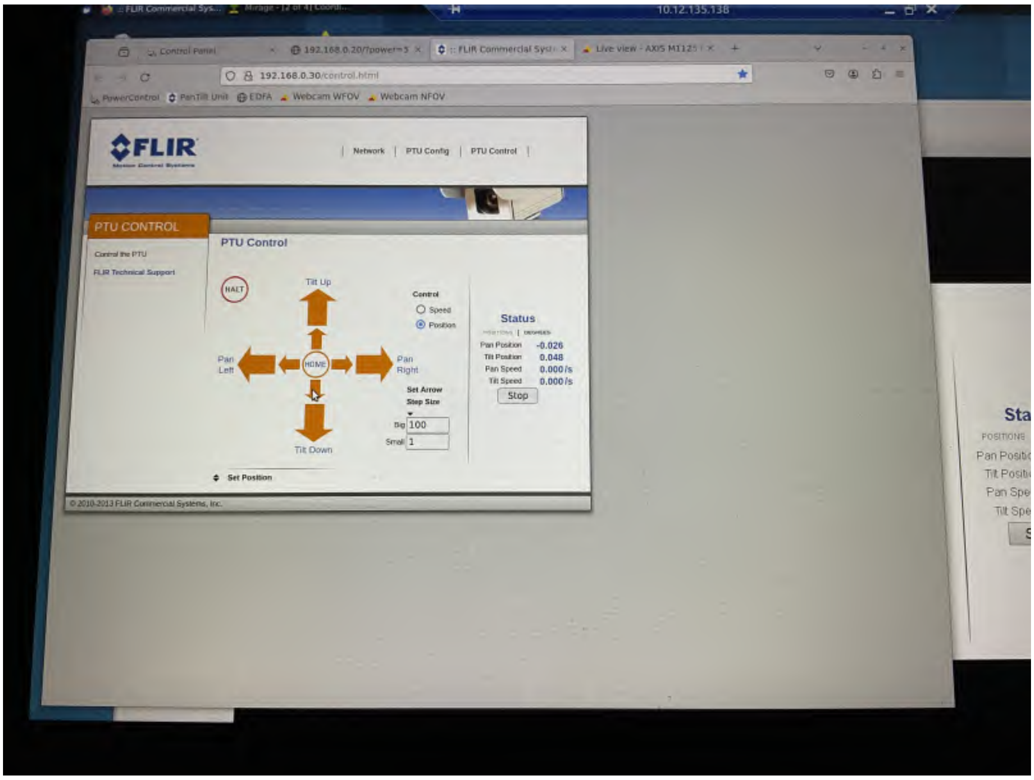


Figure 3.42: Picture of the pan-tilt unit menu

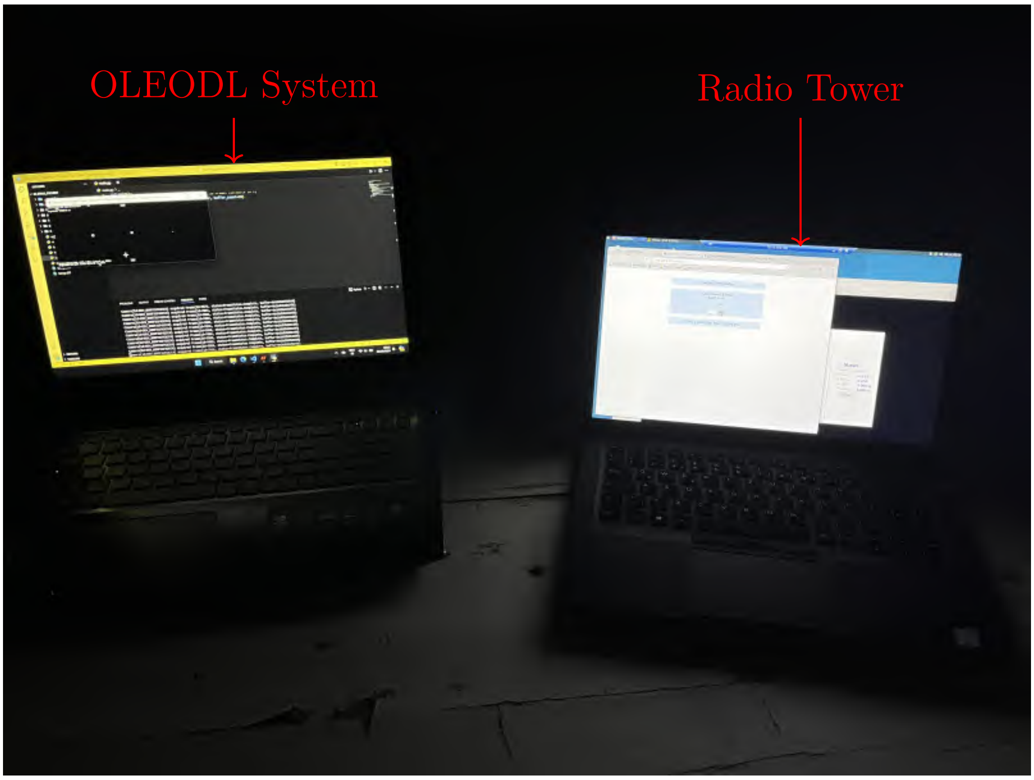


Figure 3.43: Picture of the two laptops controlling the OLEODL system (left) and the radio tower (right)

Figure 3.44: Picture of the Allied Vision G-008 camera besides the OGS-OP

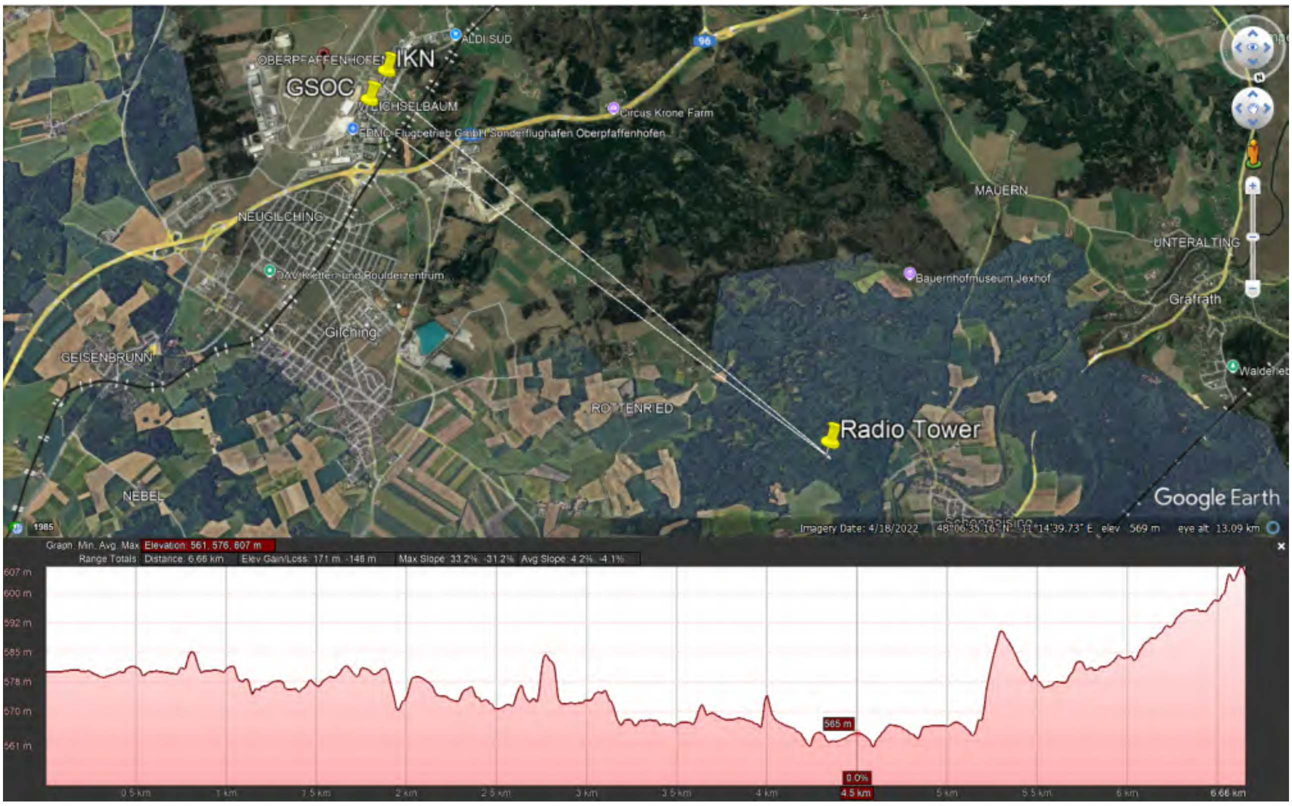


Figure 3.45: Image of the distance from the radio tower to both the IKN and GSOC. Figure taken from Google Earth Pro.

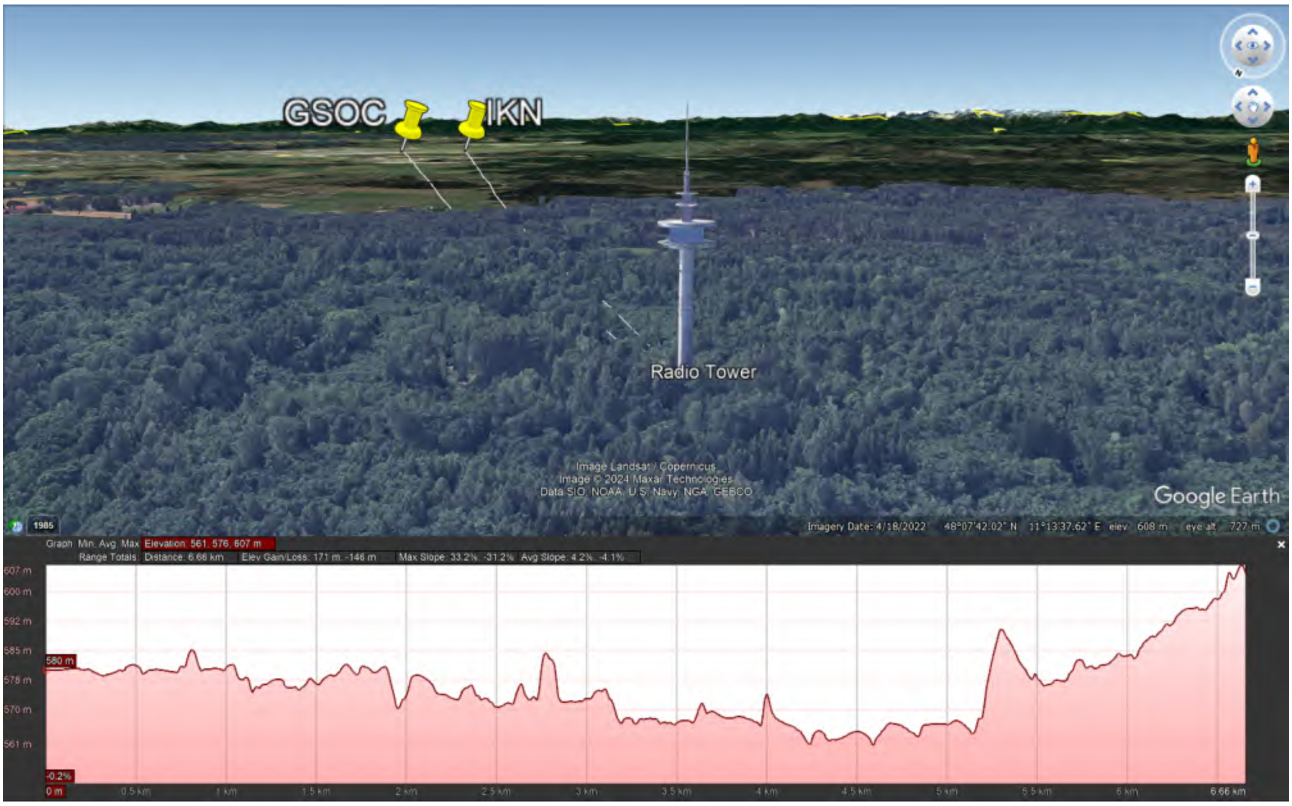


Figure 3.46: Image of the radio tower. Figure taken from Google Earth Pro.

4 Results & Discussion

We found that the device was useful, capable of detecting the satellite downlink and helping with the identification of the failure point of the system. We tested the device while this work was still in development, which means that the state of the device in the point of each measurement was usually different. This proved to be a challenging process as satellite downlinks are organized in campaigns—periods of approximately two weeks where links are conducted every two days or so. If the device was not ready for a specific campaign, we would not get any measurements until the next campaign, which are really limited, taking several months between each other.

The first campaign we participated in was the Flying Laptop (FLP) campaign in April. This campaign was unsuccessful as the software was incomplete at that time. The exposure time could not be changed during operation, and the satellite was tracked by just looking for the brightest point of the image without any filtering, thresholding or spot size filtering. Also, the visibility conditions were inadequate during the campaign, which means that we probably would have not seen the laser signal from OSIRIS on the satellite either way.

4.1 OSIRIS4CubeSat Onboard Laser CubeSat Campaign

The Laser CubeSat (CubeL) campaign took place in early July, while our software was still under development. During this campaign, the outdated image processing technique was still used, however we had already implemented the higher gain mode and the Look-Up Table (LUT) into the camera, improving our detection threshold.

We detected the OSIRIS4CubeSat laser terminal onboard CubeL during this campaign, whereas the OGS-OP did not. Initially, we considered the detection to be due to sun reflecting onto the satellite, as some links were attempted still in the visible time. However, we believe we saw the actual signal from the terminal, as sunlight reflection would have produced sudden sparks of light, rather than a continuous signal during the satellite pass.

CubeL, as explained in subsection 2.5.2, is the type of satellite that requires to receive a beacon to start the downlink. We tried to detect the satellite again when the beacon had not been sent, and did not observe it, which supports our hypothesis of seeing the signal. Although this result may seem minor, just for our camera to be able to see the laser terminal from the satellite, suggest that the Optical Ground Station (OGS) should also be able to see it. The OGS not detecting the signal indicates a potential issue with the telescope, probably related to the pointing accuracy. This was exactly one of the main goals of this work, and we managed to prove its value.

Although highly unlikely, the satellite's orbit could be slightly off, enough to miss it. For this reason, the camera system is also designed to assess the elevation and azimuth of the satellite, and the dB lost with respect to the expected intensity received from the laser terminal at that precise elevation angle: assessing how off the satellite was from the expected values. This feature was not implemented for this campaign and still requires further testing.

A frame of the pass can be seen in Fig. 4.1. We could also present the intensity values during the pass, but it is not really useful as the calibration of the intensity is inaccurate.

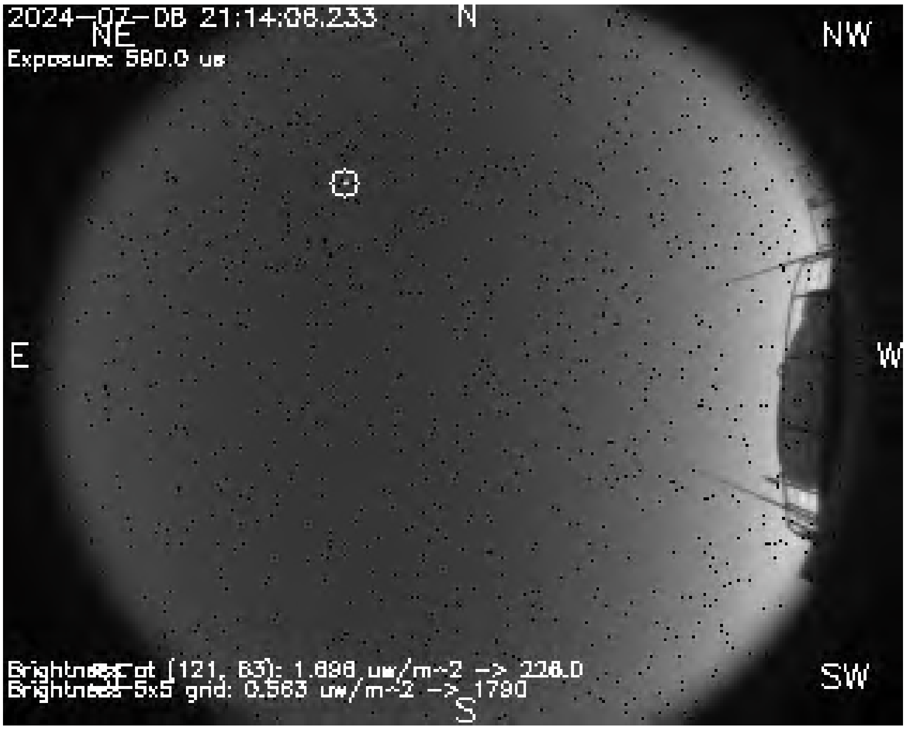


Figure 4.1: Pictures taken with the OLEODL system of the CubeL satellite

4.2 CubeCat Onboard NORSAT-TD Campaign

We contacted the Dutch Organization for Applied Scientific Research (TNO) to establish a link with their satellite, NORSAT-TD, fitted with the CubeCat laser terminal. This satellite operates similarly to CubeL, with the principal difference being that its downlink remains active from the horizon, using the beacon only to locate the targeted OGS, and refine its pointing accuracy. This satellite stays within visible range for approximately the first 12 degrees, before shifting to the infrared spectrum.

Conducted at the German Space Operations Center (GSOC) during the last week of August, this campaign was also unsuccessful. While the satellite downlink process worked normally for TNO in The Hague, we were unable to consistently detect the CubeCat laser terminal onboard NORSAT-TD: nor did the GSOC-OGS, nor did I with the camera. We detected the laser terminal during the visual range on the first day, but we were unable to replicate the results the following day due to clouds.

We arranged a meeting to discuss potential causes of the issues we encountered. Two main points were identified: first, the Two-Line Elements (TLE) files we used might not have been precise enough, as TNO uses Consolidated Prediction Format (CPF) files, which they find to be more accurate. The second was the differences in our beacons; while ours is a 4 W beacon, TNO's is 6 W. Although this difference in power should not have been a problem, the divergence angle difference was significant. TNO's beacon had a divergence angle of 220 µrad, whereas our beacon had a divergence angle of 2.1 mrad. This 10x increase could have prevented the satellite to detect our beacon, resulting in the satellite spiraling instead of sending the downlink directly to us.

The following day, we found out that the link was impeded due to problems with the orbital elements and the too low intensity of the beacon pointing to the satellite. This made sense, because even if the TLE was imprecise and the satellite was not pointing directly at us, the OLEODL system developed in this work should have detected the satellite, however it did not. After fitting a new collimator into the telescope, we tried again the same night, but nothing was seen except three or four sparks in the OLEODL system. These sparks seem to be the reflection of the downlink interacting with the beacon of the telescope. This occurred due to humid conditions that night, which caused some fog, leading to atmospheric interactions and reflections.

The presence of these reflections indicated that the satellite was likely sending the downlink as intended. However, we were unable to assess the exact cause of our failure.

Figure 4.2: Pictures taken with the OLEODL system of the reflections from the NORSAT-TD satellite.

Figure 4.3: Picture taken of the setup for the NORSAT-TD link

5 Conclusion

5.1 Summary

Optical Low Earth Orbit DownLinks (OLEODL) are establishing themselves as an alternative to Radio Frequency (RF) links in specific scenarios due to their reduced equipment volume, lower power consumption, and avoidance of the regulatory restrictions and tariffs associated with RF.

When attempting Optical Low Earth Orbit Downlinks (OLEODL), various factors can cause a failure, such as weather conditions, visibility, performance of the Optical Ground Station (OGS), or issues with the satellite itself. Identifying the precise cause of a failure is challenging.

This work focuses on building a compact, portable, and waterproof validation tool for OLEODL links. This tool is capable of assessing the signal intensity of a laser terminal onboard a satellite, including deviations from the expected intensity as per the link budget. The azimuth and elevation angle of the satellite are also computed, evaluating the overall performance of the link. Our device is based on an Indium Gallium Arsenide (InGaAs) camera, which operates in the $900 - 1700$ nm spectral range. The captured images are processed using a filtering-thresholding-contouring technique, for the intensity-dependent value on that precise elevation to be compared to the expected intensity on that same elevation angle.

This thesis demonstrates that the proposed proof-of-concept is feasible, and can be a valuable tool during the downlinks, capable of detecting the satellite even when the OGS did not, providing us with another tool to assess the failure point of the system. The goal is to establish this device as a commonly used tool during the OLEODL campaigns. Although the system is operational, it is still under development, with potential identified improvements and further testing needed.

The lack of consistent downlink campaigns, combined with the difficulties in establishing the link, made testing challenging. Additionally, the imprecise intensity calibration caused the recorded values to deviate from the expected values; if the values are far from our calibration factor.

5.2 Future Work

The AllSky-Camera system for Monitoring of Optical Satellite Downlinks aims to become a standard tool for optical satellite downlinks. The main improvement needed lies in the intensity calibration process, explained in subsection 3.5.4, which could not be fully accomplished due to the lack of appropriate tools (a challenge impossible to solve within our time frame).

Having proven the device's usefulness, it will be upgraded with more suitable components, replacing off-the-shelf components with custom-made ones. A custom-designed Field of View closer to 180 degrees (allowing us to see the whole hemisphere) with appropriate transitivity will be commissioned, as well as a crystal dome made of a more fitting material, such as BK7. These upgrades were initially omitted to keep the proof-of-concept within a limited budget.

A 1550 nm band pass filter could be implemented. The main problem with these filters is that their performance varies, highly, depending on the incident angle, rendering them useless for our case. If we really want to implement the band pass filter, it should be placed between the sensor and the lens, where the incident angles are much lower than before the lens. The spacing between the lens and the sensor is approximately 4 mm, which would allow us to find a suitable filter if desired.

As a minor improvement, remote operation of the system could also be implemented in the future. The project was developed with this in mind, but the DLR firewalls prevented its deployment. An exception to the firewall is being considered, but this will take time.

In conclusion, this work successfully developed a device that will serve as an automated verification tool for any satellite downlink in the near future. The closest application of the AllSky-Camera system for Monitoring of Optical Satellite Downlinks is its deployment in Almeria, where it will operate alongside the newly developed robotic Optical Ground Station (OGS) to monitor the planned satellite downlinks.

Bibliography

[1] N. Takato, N. Okada, G. Kosugi, M. Suganuma, A. Miyashita, and F. Uraguchi, "All-sky 10-um cloud monitor on mauna kea," in *Large Ground-based Telescopes*, vol. 4837. SPIE, 2003, pp. 872–877.

[2] Y. Wang, D. Liu, W. Xie, M. Yang, Z. Gao, X. Ling, Y. Huang, C. Li, Y. Liu, and Y. Xia, "Day and night clouds detection using a thermal-infrared all-sky-view camera," *Remote Sensing*, vol. 13, no. 9, p. 1852, 2021.

[3] P. Crispel and G. Roberts, "All-sky photogrammetry techniques to georeference a cloud field," *Atmospheric measurement techniques*, vol. 11, no. 1, pp. 593–609, 2018.

[4] E. Kerr, B. D. C. López, N. Maric, J. N. Torres, G. Falco, N. Sánchez Ortiz, C. Dorn, and S. Eves, "Design and prototyping of a low-cost leo optical surveillance sensor," in *Proceedings of the 8th European Conference on Space Debris, Darmstadt, Germany*, 2021, pp. 20–23.

[5] M. N. Sarvi, D. Abbasi-Moghadam, M. Abolghasemi, and H. Hoseini, "Design and implementation of a star-tracker for leo satellite," *Optik*, vol. 208, p. 164343, 2020.

[6] K. Riesing, H. Yoon, and K. Cahoy, "A portable optical ground station for low-earth orbit satellite communications," in *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*. IEEE, 2017, pp. 108–114.

[7] N. Estell, D. Ma, and P. Seitzer, "Daylight imaging of leo satellites using cots hardware," in *Proc. AMOS*, 2019.

[8] H. Kaushal, V. Jain, and S. Kar, *Free space optical communication*. Springer, 2017, vol. 60.

[9] A. G. Bell, "Apparatus for signaling and communi-cating, called photophone," U.S. Patent No. 235,199, Dec. 7, 1880.

[10] G. Marconi, "Improvements in transmitting electrical impulses and signals and in apparatus there-for," British Patent No. 12,039, Jul. 7, 1897.

[11] A. G. Bell, "The photophone," *Science*, no. 11, pp. 130–134, 1880.

[12] D. Giggenbach, "Optimierung der optischen freiraumkommunikation durch die turbulente atmosphäre-focal array receiver," Ph.D. dissertation, Shaker-Verlag, 2005.

[13] M. M. Hassan and G. Rather, "Free space optics (fso): a promising solution to first and last mile connectivity (flmc) in the communication networks," *IJ Wireless and Microwave Technologies*, vol. 4, no. 1, p. 1, 2020.

[14] M. N. Sadiku, S. M. Musa, and S. R. Nelatury, "Free space optical communications: an overview," *European scientific journal*, vol. 12, no. 9, pp. 55–68, 2016.

[15] K. A. Sebastian. "UCS Satellite Database". Union of Concerned Scientist. Accessed: Aug. 9, 2024. [Online]. Available: https://www.ucsusa.org/resources/satellite-database

[16] P.-H. Chen and R. L.-T. Cho, "The technological trajectory of leo satellites: Perspectives from main path analysis," *IEEE Transactions on Engineering Management*, 2023.

[17] V. Pesce, A. Colagrossi, and S. Silvestrini, “Chapter four - orbital dynamics,” in *Modern spacecraft guidance, navigation, and control: from system modeling to AI and innovative applications.* Elsevier, 2022, pp. 131–206.

[18] H. Riebeek. ”Catalog of Earth Satellite Orbits”. NASA Earth Observatory. Accessed: Aug. 9, 2024. [Online]. Available: https://earthobservatory.nasa.gov/features/OrbitsCatalog

[19] “Syncom II Summary Report,” NASA - Hughes Aircraft Co. SSD 3128R, Culver City, CA, USA, Sum. Rep. N66-23589, 1963, Accessed: Aug. 9, 2024. [Online]. Available: https://ntrs.nasa.gov/api/citations/19660014300/downloads/19660014300.pdf.

[20] P. Dickson, *Sputnik: The shock of the century.* Bloomsbury Publishing USA, 2001.

[21] K. Maine, C. Devieux, and P. Swan, “Overview of iridium satellite network,” in *Proceedings of WESCON’95.* IEEE, 1995, p. 483.

[22] J. Puig-Suari, J. Schoos, C. Turner, T. Wagner, R. Connolly, and R. P. Block, “Cubesat developments at cal poly: the standard deployer and polysat,” in *Small Payloads in Space*, vol. 4136. SPIE, 2000, pp. 72–78.

[23] C. Kitts, J. Hines, E. Agasid, A. Ricco, B. Yost, K. Ronzano, and J. Puig-Suari, “The genesat-1 microsatellite missiona challenge in small satellite design,” 2006.

[24] F. Michel, M. Trevisan, D. Giordano, and O. Bonaventure, “A first look at starlink performance,” in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 130–136.

[25] C. Pardini and L. Anselmo, “Environmental sustainability of large satellite constellations in low earth orbit,” *Acta Astronautica*, vol. 170, pp. 27–36, 2020.

[26] J. N. Pelton, “The space debris threat and the kessler syndrome,” in *Space Debris and Other Threats from Outer Space.* Springer, 2013, pp. 17–23.

[27] “ESA’S Annual Space Environment Report,” ESA Space Debris Office, Darmstadt, Germany, Ann. Rep., 2024. Accessed: Aug. 10, 2024. [Online]. Available: https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf.

[28] “International Space Station Deorbit Analysis Summary,” NASA, Washington, DC, USA, White Paper, 2024. Accessed: Aug. 10, 2024. [Online]. Available: https://www.nasa.gov/wp-content/uploads/2024/06/iss-deorbit-analysis-summary.pdf.

[29] A. Carrasco-Casado and R. Mata-Calvo, “Space optical links for communication networks,” in *Springer Handbook of Optical Networks.* Springer, 2020, pp. 1057–1103.

[30] Z. Sodnik, J. P. Armengol, R. H. Czichy, and R. Meyer, “Adaptive optics and esa’s optical ground station,” in *Free-Space Laser Communications IX*, vol. 7464. SPIE, 2009, pp. 47–55.

[31] J. Crass, “The adaptive optics lucky imager: combining adaptive optics and lucky imaging,” Ph.D. dissertation, 2014.

[32] C. Fuchs, D. Giggenbach, R. M. Calvo, and W. Rosenkranz, “Optical transmitter diversity with phase-division in bit-time,” in *2022 IEEE International Conference on Space Optical Systems and Applications (ICSOS).* IEEE, 2022, pp. 1–4.

[33] R. M. Calvo, P. Becker, D. Giggenbach, F. Moll, M. Schwarzer, M. Hinz, and Z. Sodnik, “Transmitter diversity verification on artemis geostationary satellite,” in *Free-space laser communication and atmospheric propagation xxvi*, vol. 8971. SPIE, 2014, pp. 24–37.

[34] D. Giggenbach, F. Moll, C. Schmidt, C. Fuchs, and A. Shrestha, “Optical on-off keying data links for low earth orbit downlink applications,” *Satellite Communications in the 5G Era*, vol. 79, pp. 307–339, 2018.

[35] A. Mustafa, D. Giggenbach, J. Poliak, A. Shrestha, R. Mata-Calvo, and C. Fuchs, “Lab implementation of 10 gbps/channel optical transmitter diversity scheme for geostationary satellite feeder links,” in *Photonic Networks; 16. ITG Symposium.* VDE, 2015, pp. 1–3.

[36] R. Kaushik, V. Khandelwal, and R. C. Jain, “Effect of aperture averaging and spatial diversity on capacity of optical wireless communication systems over lognormal channels,” *Radioelectronics and Communications Systems*, vol. 59, pp. 527–535, 2016.

[37] A. Mustafa, “Spectrally efficient transmitter diversity scheme for optical satellite feeder links,” Ph.D. dissertation, Universität Stuttgart, 2024.

[38] M. R. Garcia-Talavera, J. A. Rodriguez, T. Viera, H. Moreno-Arce, J. L. Rasilla, F. Gago, L. F. Rodriguez, P. Gomez, and E. B. Ramirez, “Design and performance of the esa optical ground station,” in *Free-Space Laser Communication Technologies XIV*, vol. 4635. SPIE, 2002, pp. 248–261.

[39] A. Alonso, M. Reyes, and Z. Sodnik, “Performance of satellite-to-ground communications link between artemis and the optical ground station,” in *Optics in Atmospheric Propagation and Adaptive Systems VII*, vol. 5572. SPIE, 2004, pp. 372–383.

[40] K. Saucke, C. Seiter, F. Heine, M. Gregory, D. Tröndle, E. Fischer, T. Berkefeld, M. Feriencik, M. Feriencik, I. Richter *et al.*, “The tesat transportable adaptive optical ground station,” in *Free-Space Laser Communication and Atmospheric Propagation XXVIII*, vol. 9739. SPIE, 2016, pp. 37–47.

[41] IQOQI Vienna, Austrian Academy of Sciences. ”Laser from Optical Ground Station on Tenerife”. ESA. Accessed: Aug. 12, 2024. [Online]. Available: https://www.esa.int/ESA_Multimedia/Images/2012/09/Laser_from_Optical_Ground_Station_on_Tenerife

[42] E. Fischer, T. Berkefeld, M. Feriencik, M. Feriencik, V. Kaltenbach, D. Soltau, B. Wandernoth, R. Czichy, J. Kunde, K. Saucke *et al.*, “Development, integration and test of a transportable adaptive optical ground station,” in *2015 IEEE International Conference on Space Optical Systems and Applications (ICSOS).* IEEE, 2015, pp. 1–6.

[43] N. Perlot, M. Knapek, D. Giggenbach, J. Horwath, M. Brechtelsbauer, Y. Takayama, and T. Jono, “Results of the optical downlink experiment kiodo from oicets satellite to optical ground station oberpfaffenhofen (ogs-op),” in *Free-Space Laser Communication Technologies XIX and Atmospheric Propagation of Electromagnetic Waves*, vol. 6457. SPIE, 2007, pp. 28–35.

[44] J. Prell, A. Duliu, R. Andrew, I. Hristovski, S. Amita, F. Moll, and C. Fuchs, “Optical ground station oberpfaffenhofen next generation: first satellite link tests with 80 cm telescope and ao system,” in *2023 IEEE International Conference on Space Optical Systems and Applications (ICSOS).* IEEE, 2023, pp. 42–48.

[45] I. R. Hristovski, A. R. Campelo, B. Femenía-Castella, E. Doensdorf-Sternal, A. O. Duliu, S. Haeusler, J. F. Holzman, K. Klemich, D. J. Laidlaw, T. Marynowski *et al.*, “Pre-distortion adaptive optics: experimental results from bi-directional tracking links between dlr’s optical ground station and alphasat’s tdp-1 terminal,” in *Free-Space Laser Communications XXXVI*, vol. 12877. SPIE, 2024, pp. 328–336.

[46] G. E. Stillman, “Optoelectronics,” in *Reference data for engineers: radio, electronics, computers and communications.* Elsevier, 2002, pp. 21–1–21–31.

[47] B. Rödiger, D. Ginthör, J. P. Labrador, J. Ramirez, C. Schmidt, and C. Fuchs, “Demonstration of an fso/rf hybrid-communication system on aeronautical and space applications,” in *Laser Communication and Propagation through the Atmosphere and Oceans IX*, vol. 11506. SPIE, 2020, p. 1150603.

[48] K. E. Wilson, J. R. Lesh, and T.-Y. Yan, "Gopex: a laser uplink to the galileo spacecraft on its way to jupiter," in *Free-Space Laser Communication Technologies V*, vol. 1866. SPIE, 1993, pp. 138–146.

[49] K. Araki, Y. Arimoto, M. Shikatani, M. Toyoda, M. Toyoshima, T. Takahashi, S. Kanda, and K. Shiratama, "Performance evaluation of laser communication equipment onboard the ets-vi satellite," in *Free-Space Laser Communication Technologies VIII*, vol. 2699. SPIE, 1996, pp. 52–59.

[50] G. Fletcher, T. Hicks, and B. Laurent, "The silex optical interorbit link experiment," *Electronics & communication engineering journal*, vol. 3, no. 6, pp. 273–279, 1991.

[51] J. Horwath, N. Perlot, M. Knapek, and F. Moll, "Experimental verification of optical backhaul links for high-altitude platform networks: Atmospheric turbulence and downlink availability," *International Journal of Satellite Communications and Networking*, vol. 25, no. 5, pp. 501–528, 2007.

[52] M. Toyoshima, H. Takenaka, Y. Shoji, Y. Takayama, Y. Koyama, and H. Kunimori, "Results of kirari optical communication demonstration experiments with nict optical ground station (koden) aiming for future classical and quantum communications in space," *Acta Astronautica*, vol. 74, pp. 40–49, 2012.

[53] R. Fields, C. Lunde, R. Wong, J. Wicker, D. Kozlowski, J. Jordan, B. Hansen, G. Muehlnikel, W. Scheel, U. Sterr *et al.*, "Nfire-to-terrasar-x laser communication results: satellite pointing, disturbances, and other attributes consistent with successful performance," in *Sensors and Systems for Space Applications III*, vol. 7330. SPIE, 2009, pp. 211–225.

[54] D. M. Boroson, B. S. Robinson, D. V. Murphy, D. A. Burianek, F. Khatri, J. M. Kovalik, Z. Sodnik, and D. M. Cornwell, "Overview and results of the lunar laser communication demonstration," in *Free-Space Laser Communication and Atmospheric Propagation XXVI*, vol. 8971. SPIE, 2014, pp. 213–223.

[55] A. Carrasco-Casado, H. Takenaka, D. Kolev, Y. Munemasa, H. Kunimori, K. Suzuki, T. Fuse, T. Kubo-Oka, M. Akioka, Y. Koyama *et al.*, "Leo-to-ground optical communications using sota (small optical transponder)–payload verification results and experiments on space quantum communications," *Acta Astronautica*, vol. 139, pp. 377–384, 2017.

[56] D. Giggenbach, J. Poliak, R. Mata-Calvo, C. Fuchs, N. Perlot, R. Freund, and T. Richter, "Preliminary results of terabit-per-second long-range free-space optical transmission experiment thrust," in *Unmanned/Unattended Sensors and Sensor Networks XI; and Advanced Free-Space Optical Communication Techniques and Applications*, vol. 9647. SPIE, 2015, pp. 61–73.

[57] J. Yin, Y. Cao, Y.-H. Li, S.-K. Liao, L. Zhang, J.-G. Ren, W.-Q. Cai, W.-Y. Liu, B. Li, H. Dai *et al.*, "Satellite-based entanglement distribution over 1200 kilometers," *Science*, vol. 356, no. 6343, pp. 1140–1144, 2017.

[58] D. Calzolaio, F. Curreli, J. Duncan, A. Moorhouse, G. Perez, and S. Voegt, "Edrs-c–the second node of the european data relay system is in orbit," *Acta Astronautica*, vol. 177, pp. 537–544, 2020.

[59] D. J. Israel, B. L. Edwards, R. L. Butler, J. D. Moores, S. Piazzolla, N. Du Toit, and L. Braatz, "Early results from nasa's laser communications relay demonstration (lcrd) experiment program," in *Free-Space Laser Communications XXXV*, vol. 12413. SPIE, 2023, pp. 10–24.

[60] D. Rieländer, A. Di Mira, D. Alaluf, R. Daddato, S. Mejri, J. Piris, J. Alves, D. Antsos, A. Biswas, N. Karafolas *et al.*, "Esa ground infrastructure for the nasa/jpl psyche deep-space optical communication demonstration," in *International Conference on Space Optics—ICSO 2022*, vol. 12777. SPIE, 2023, pp. 159–169.

[61] D. Giggenbach, A. Shrestha, C. Fuchs, C. Schmidt, and F. Moll, “System aspects of optical leo-to-ground links,” in *International Conference on Space Optics—ICSO 2016*, vol. 10562. SPIE, 2017, pp. 1635–1643.

[62] D. Giggenbach and F. Moll, “Scintillation loss in optical low earth orbit data downlinks with avalanche photodiode receivers,” in *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*. IEEE, 2017, pp. 115–122.

[63] D. Giggenbach, F. Moll, C. Fuchs, T. de Cola, and R. Mata-Calvo, “Space communications protocols for future optical satellite-downlinks,” 2011.

[64] D. Giggenbach, M. T. Knopp, and C. Fuchs, “Link budget calculation in optical leo satellite downlinks with on/off-keying and large signal divergence: A simplified methodology,” *International Journal of Satellite Communications and Networking*, vol. 41, no. 5, pp. 460–476, 2023.

[65] A. Carrillo-Flores, D. Giggenbach, M. Knopp, D. Orsucci, and A. Shrestha, “Effects of pointing errors on intensity losses in the optical leo uplink,” in *International Conference on Space Optics—ICSO 2022*, vol. 12777. SPIE, 2023, pp. 2476–2491.

[66] EDMUND OPTICS. ”Gaussian Beam Propagation”. EDMUND OPTICS. Accessed: Feb. 27, 2024. [Online]. Available: https://www.edmundoptics.com/knowledge-center/application-notes/lasers/gaussian-beam-propagation/

[67] D. Giggenbach and A. Shrestha, “Atmospheric absorption and scattering impact on optical satellite-ground links,” *International Journal of Satellite Communications and Networking*, vol. 40, no. 2, pp. 157–176, 2022.

[68] H. Hemmati, “Near-earth laser communications,” in *Near-Earth Laser Communications, Second Edition*. CRC press, 2020, pp. 1–40.

[69] A. Tunick, “The refractive index structure parameter/atmospheric optical turbulence model: Cn2 (no. arl-tr-1615),” *ARMY RESEARCH LABORATORY ADELPHI MD*, 1998.

[70] Y. K. Chahine, S. A. Tedder, B. E. Vyhnalek, and A. C. Wroblewski, “Beam propagation through atmospheric turbulence using an altitude-dependent structure profile with non-uniformly distributed phase screens,” in *Free-Space Laser Communications XXXII*, vol. 11272. SPIE, 2020, pp. 263–277.

[71] E. Aristidi, A. Ziad, J. Chabé, Y. Fantéi-Caujolle, C. Renaud, and C. Giordano, “A generalized differential image motion monitor,” *Monthly Notices of the Royal Astronomical Society*, vol. 486, no. 1, pp. 915–925, 2019.

[72] F. Moll and M. Knapek, “Wavelength selection criteria and link availability due to cloud coverage statistics and attenuation affecting satellite, aerial, and downlink scenarios,” in *Free-Space Laser Communications VII*, vol. 6709. SPIE, 2007, pp. 347–358.

[73] J. P. Jakobs, A. Köhler, M. T. Knopp, and A. Ohndorf, “Development of an optical ground station (ogs) network in germany and europe,” 2023.

[74] D. Giggenbach and H. Henniger, “Fading-loss assessment in atmospheric free-space optical communication links with on-off keying,” *Optical Engineering*, vol. 47, no. 4, pp. 046 001–046 001, 2008.

[75] F. Moll, D. Giggenbach, C. Schmidt, and C. Fuchs, “Analysis of power scintillation and fading margin in the leo-ground downlink with the osirisv1 laser terminal on flying laptop and the dlr optical ground station oberpfaffenhofen,” in *Environmental Effects on Light Propagation and Adaptive Systems V*, vol. 12266. SPIE, 2022, pp. 75–82.

[76] J. Keim, S. Gaißer, P. Hagel, M. Böttcher, M. Lengowski, M. Graß, D. Giggenbach, C. Fuchs, C. Schmidt, S. Klinkner *et al.*, "Commissioning of the optical communication downlink system osirisv1 on the university small satellite "flying laptop"," in *70th International Astronautical Congress (IAC), Washington*, 2019.

[77] D. Giggenbach, C. Fuchs, C. Schmidt, B. Rödiger, S. Gaißer, S. Klinkner, D.-H. Phung, J. Chabé, C. Courde, N. Maurice *et al.*, "Downlink communication experiments with osirisv1 laser terminal onboard flying laptop satellite," *Applied optics*, vol. 61, no. 8, pp. 1938–1946, 2022.

[78] D. Giggenbach, P. Karafillis, J. Rittershofer, A. Immerz, A. Spoerl, S. Gaisser, S. Klinkner, and M. Knopp, "Transmitter beam bias verification for optical satellite data downlinks with open-loop pointing–the 3-ogs-experiment," in *International Conference on Space Optics—ICSO 2022*, vol. 12777. SPIE, 2023, pp. 398–412.

[79] C. Fuchs, F. Moll, D. Giggenbach, C. Schmidt, J. Keim, and S. Gaisser, "Osirisv1 on flying laptop: Measurement results and outlook," in *2019 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*. IEEE, 2019, pp. 1–5.

[80] B. Rödiger and C. Schmidt, "In-orbit demonstration of the world's smallest laser communication terminal-osiris4cubesat/cubelct," in *Small Satellites Systems and Services-The 4S Symposium 2024*, 2024.

[81] B. Rödiger, C. Menninger, C. Fuchs, L. Grillmayer, S. Arnold, C. Rochow, P. Wertz, and C. Schmidt, "High data-rate optical communication payload for cubesats," in *Laser Communication and Propagation through the Atmosphere and Oceans IX*, vol. 11506. SPIE, 2020, pp. 12–24.

[82] DLR Institute of Communications and Navigation. "OSIRIS4CubeSat / CubeLCT". DLR. Accessed: Aug. 14, 2024. [Online]. Available: https://www.dlr.de/en/kn/research-transfer/projects/osiris-optical-communication-in-space/cube4cubesat-cubelct

[83] B. Rödiger, L. R. Rodeck, M.-T. Hahn, and C. Schmidt, "Transformation of dlr's laser communication terminals for cubesats towards new application scenarios," in *Small Satellites Systems and Services-The 4S Symposium 2024*, 2024.

[84] DLR Institute of Communications and Navigation. "OSIRIS - Optical Communication in Space". DLR. Accessed: Aug. 14, 2024. [Online]. Available: https://www.dlr.de/en/kn/research-transfer/projects/osiris-optical-communication-in-space

[85] SONY. "Short-Wavelength InfraRed Image Sensor Technology SenSWIR™". SONY. Accessed: Jan. 18, 2024. [Online]. Available: https://www.sony-semicon.com/en/technology/industry/senswir.html

[86] IDS Imaging Development Systems GmbH. "Sensor sizes". IDS. Accessed: Jan. 21, 2024. [Online]. Available: https://www.1stvision.com/cameras/IDS/IDS-manuals/en/basics-sensor-size.html

[87] Allied Vision. "Goldeye G-008 TEC1". Allied Vision. Accessed: Feb. 13, 2024. [Online]. Available: https://www.alliedvision.com/en/camera-selector/detail/goldeye/g-008-tec1/

[88] ——. "Goldeye G-130 TEC1". Allied Vision. Accessed: Feb. 13, 2024. [Online]. Available: https://www.alliedvision.com/en/camera-selector/detail/goldeye/g-130-tec1/

[89] Exosens. "Bobcat 320 Series". Exosens. Accessed: Feb. 22, 2024. [Online]. Available: https://www.exosens.com/products/bobcat-320-series

[90] NAVITAR. "SWIR/Hyperspectral Lenses - SWIR-8". NAVITAR. Accessed: Aug. 16, 2024. [Online]. Available: https://www.navitar.com/-/media/project/oneweb/oneweb/navitar/navitar-product/swir/swir-catalog-page-navitar.pdf?la=en&revision=6d4f2a19-1a0b-49c9-8719-d26a6d921765

[91] KOWA. ”LM100JC1MS | 2/3” 100mm 2MP C-Mount Lens”. KOWA. Accessed: Mar. 12, 2024. [Online]. Available: https://www.kowa-lenses.com/en/lm100jc1ms-2mp-industrial-lens-c-mount

[92] THORLABS. ”MVL4WA - 3.5 mm EFL, f/1.4, for 1/2” C-Mount Format Cameras, with Lock”. THORLABS. Accessed: Mar. 12, 2024. [Online]. Available: https://www.thorlabs.de/thorproduct.cfm?partnumber=MVL4WA

[93] FUJIFILM. ”FE185 Series”. FUJIFILM. Accessed: Mar. 12, 2024. [Online]. Available: https://www.fujifilm.com/de/en/business/optical-devices/mvlens/fe185

[94] KOWA. ”LM100JC1MS(100mmF2.8) Transmission”. KOWA. Accessed: Mar. 12, 2024. [Online]. Available: https://www.kowa-lenses.com/media/pdf/34/d4/83/LM100JC1MS_Transmission_03_2020.pdf

[95] THORLABS. ”MVL4WA - Machine Vision Lens Transmission”. THORLABS. Accessed: Mar. 12, 2024. [Online]. Available: https://view.officeapps.live.com/op/view.aspx?src=https%3A%2F%2Fwww.thorlabs.de%2Fimages%2Ftabimages%2FMVL4WA.xlsx&wdOrigin=BROWSELINK

[96] ——. ”Camera Lenses for Machine Vision”. THORLABS. Accessed: Mar. 12, 2024. [Online]. Available: https://www.thorlabs.de/newgrouppage9.cfm?objectgroup_id=1822

[97] EDMUND OPTICS. ”The Correct Material for Infrared (IR) Applications”. EDMUND OPTICS. Accessed: Mar. 27, 2024. [Online]. Available: https://www.edmundoptics.de/knowledge-center/application-notes/optics/the-correct-material-for-infrared-applications/#:~:text=Index%20of%20Refraction%20%E2%80%93%20IR,N%2DSF11%2C%20BOROFLOAT%C2%AE%2C%20etc.)

[98] KOFFERMARKT. ”Max Koffer MAX540H245 Outdoor Case 2 Rollen leer schwarz”. KOFFERMARKT. Accessed: Jun. 20, 2024. [Online]. Available: https://www.koffermarkt.com/max-koffer-max540h245-outdoor-case-2-rollen/

[99] Iker Aldasoro. ”AllSkyCam4OLEODL package”. READ THE DOCS. Accessed: Aug. 22, 2024. [Online]. Available: https://allskycam4oleodl.readthedocs.io/en/latest/AllSkyCam4OLEODL.html

[100] ALLIED VISION. ”VmbPy repository”. GITHUB. Accessed: Aug. 22, 2024. [Online]. Available: https://github.com/alliedvision/VmbPy

[101] Iker Aldasoro. ”AllSkyCam4OLEODL repository”. GITHUB. Accessed: Aug. 22, 2024. [Online]. Available: https://github.com/Ikerald/AllSkyCam4OLEODL

[102] ALLIED VISION. ”Python API Manual”. READ THE DOCS. Accessed: Aug. 22, 2024. [Online]. Available: https://docs.alliedvision.com/Vimba_X/Vimba_X_DeveloperGuide/pythonAPIManual.html

[103] “Linearity Goldeye G/CL-008,” Allied Vision, Osnabrueck, Germany, Tech. Rep., May. 2024.

[104] S. Paris, P. Kornprobst, J. Tumblin, F. Durand *et al.*, “Bilateral filtering: Theory and applications,” *Foundations and Trends® in Computer Graphics and Vision*, vol. 4, no. 1, pp. 1–73, 2009.

[105] F. Durand and J. Dorsey, “Fast bilateral filtering for the display of high-dynamic-range images,” in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 257–266.

[106] N. Otsu *et al.*, “A threshold selection method from gray-level histograms,” *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[107] A. Nabil, M. Abdelfatah, A. Mousa, and G. El-Fiky, “A new model for reduction of azimuth asymmetry biases of tropospheric delay,” *NRIAG Journal of Astronomy and Geophysics*, vol. 8, no. 1, pp. 97–102, 2019.

[108] PANOTOOLS. "Fisheye Projection". PANOTOOLS. Accessed: Apr. 16, 2024. [Online]. Available: https://wiki.panotools.org/Fisheye_Projection

[109] K. Miller. "Python API Manual". MATHEMATICS STACK EXCHANGE. Accessed: Aug. 22, 2024. [Online]. Available: https://math.stackexchange.com/questions/1596513/find-the-bearing-angle-between-two-points-in-a-2d-space

[110] R. Paschotta. "Beam Divergence". RP Photonics AG. Accessed: Mar. 7, 2024. [Online]. Available: https://www.rp-photonics.com/beam_divergence.html#:~:text=For%20a%20diffraction,beam%20waist.

[111] CORNING. "https://math.stackexchange.com/questions/1596513/find-the-bearing-angle-between-two-points-in-a-2d-space". CORNING. Accessed: Mar. 7, 2024. [Online]. Available: https://www.corning.com/media/worldwide/coc/documents/Fiber/SMF-28%20ULL.pdf

[112] H. Yoda, P. Polynkin, and M. Mansuripur, "Beam quality factor of higher order modes in a step-index fiber," *Journal of lightwave technology*, vol. 24, no. 3, pp. 1350–1355, 2006.

[113] R. Paschotta. "Mode Radius". RP Photonics AG. Accessed: Mar. 7, 2024. [Online]. Available: https://www.rp-photonics.com/mode_radius.html

[114] ——. "Beam Radius". RP Photonics AG. Accessed: Mar. 7, 2024. [Online]. Available: https://www.rp-photonics.com/beam_radius.html

[115] R. Jaiswal, "Design and implementation of an integrated optical signal sensor device for telescopes for optical satellite data reception and measurements of signal variations," Master's thesis, Technische Universität Berlin (TUB) - Institute for Aeronautics and Astronautics, 2020.

[116] S. T. Popescu, P. S. Gheorghe, and A. Petris, "Measuring very low optical powers with a common camera," *Applied Optics*, vol. 53, no. 24, pp. 5460–5464, 2014.

Appendix A

A.1 Main.py

```
"""Main fuction of the project where everything is called.

Python version:     3.12.2.
Numpy version:      2.0.0.
Scipy version:      1.14.0.
Matplotlib version: 3.9.1.
Mplcursors version: 0.5.3.
Tkinter version:    0.1.0.
Pytz version:       2024.1.
OpenCV version:     4.10.0.84.


Author:
    Iker Aldasoro - 19.04.2024
"""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\02_FINAL_PROJECT\constants.py

from tkinter import ttk
from datetime import datetime
from typing import Tuple
import time
import matplotlib.pyplot as plt
import os

import AllSkyCam4OLEODL


def create_graph(
    elevation_in, payload
) -> Tuple[plt.figure, plt.axes, plt.hlines, list, list]:
    """Creates the live graph displayed in the GUI:

    1. Creates the plot with an specific size, position, title and labels.

    2. If a payload with full elevation range has been chosen, the graph will be smaller
    ↪
    to acomodate the link budget graph.

    3. Initializes the data for the graph.

    Args:
```

```python
        elevation_in (tk.StringVar): Container of the elevation mode (Individual or Full).
        payload (tk.StringVar): Container of the payload used (None, KIODO, OsirisV1, Osiris4CubeSat, CubeCat).

    Returns:
        tuple[plt.figure, plt.axes, plt.hlines, list, list]: fig (plt.figure): Figure of the created plot.

        ax (plt.axes): Axes of the created plot.

        line (plt.hlines): Lines of the created plot.

        xdata (list): X-axis data from the created plot.

        ydata (list): Y-axis data from the created plot.
    """
    # Create figure and axis objects
    fig, ax = plt.subplots()
    if elevation_in.get() == "Full" and payload.get() != "None":
        # Size and position of the intensity graph.
        fig.set_size_inches(5.38, 3.3)
        fig.canvas.manager.window.wm_geometry("+5+290")
        fig.subplots_adjust(left=0.11, right=0.95, top=0.92, bottom=0.13)
    else:
        fig.set_size_inches(8.9, 3.3)
        fig.canvas.manager.window.wm_geometry("+5+290")
        fig.subplots_adjust(left=0.08, right=0.97, top=0.92, bottom=0.13)

    (line,) = ax.plot([], [], lw=2)

    ax.set_ylim(0, 255)
    ax.grid()
    if payload.get() != "None":
        ax.set_title(
            f"{payload.get()} downlink on {datetime.now().strftime('%Y-%m-%d')}"
        )
    else:
        ax.set_title(f"Downlink on {datetime.now().strftime('%Y-%m-%d')}")
    ax.set_xlabel("Time [UTC]")
    ax.set_ylabel("Brightness")

    # Initialize empty data
    xdata, ydata = [], []

    return fig, ax, line, xdata, ydata


def main():
    AllSkyCam4OLEODL.print_preamble()
    cam_id = AllSkyCam4OLEODL.parse_args()

    with AllSkyCam4OLEODL.VmbSystem.get_instance():
        with AllSkyCam4OLEODL.get_camera(cam_id) as cam:
            AllSkyCam4OLEODL.print_preamble_settings()

            (
                gain_in,
                check_in,
                light_in,
```

```
                payload_in,
                h_ogs_in,
                zenith_in,
                elevation_in,
                elevation_angle_in,
                exposure_in,
                exposure_time_in,
                iso_in,
                root,
                exposure_time_entry,
                elevation_angle_entry,
            ) = AllSkyCam4OLEODL.create_menu()

            def start_streaming():
                # Validate inputs
                (
                    elevation_angle,
                    exposure_time_value,
                    zenith,
                    h_ogs,
                ) = AllSkyCam4OLEODL.checks(
                    elevation_in,
                    elevation_angle_in,
                    exposure_in,
                    exposure_time_in,
                    zenith_in,
                    h_ogs_in,
                )

                if payload_in.get() != "None":
                    AllSkyCam4OLEODL.link_budget(
                        elevation_in, elevation_angle, payload_in, zenith, h_ogs
                    )

                # Setup camera
                AllSkyCam4OLEODL.upload_lut(
                    cam, AllSkyCam4OLEODL.LUT_INDEX, gain_in
                )
                AllSkyCam4OLEODL.setup_camera(
                    cam, gain_in, exposure_in, exposure_time_value, iso_in
                )
                AllSkyCam4OLEODL.setup_pixel_format(cam)
                AllSkyCam4OLEODL.grab_frame(cam)
                AllSkyCam4OLEODL.print_start_stream()

                # Create graph
                fig, ax, line, xdata, ydata = create_graph(
                    elevation_in, payload_in
                )
                # plt.show(block=False)  # Show the plot window without blocking

                handler = AllSkyCam4OLEODL.Handler(
                    cam,
                    exposure_time_value,
                    check_in,
                    light_in,
                    gain_in,
                    iso_in,
                    payload_in,
                    elevation_in,
```

```python
                        fig,
                        ax,
                        line,
                        xdata,
                        ydata,
                    )
                    handler.create_camera_control_slider(root)
                    plt.ion()  # Turn on interactive mode
                    plt.show()

                    try:
                        cam.start_streaming(handler=handler, buffer_count=10)
                        while not handler.shutdown_event.is_set():
                            root.update()
                            time.sleep(
                                0.01
                            )  # Small delay to prevent high CPU usage
                    finally:
                        cam.stop_streaming()
                        AllSkyCam4OLEODL.print_end_stream()
                        handler.save_plot()  # Save the plot when streaming stops
                        root.quit()
                        root.destroy()
                        if os.path.isfile(AllSkyCam4OLEODL.BACKGROUND_FRAME_DIR):
                            os.remove(AllSkyCam4OLEODL.BACKGROUND_FRAME_DIR)

                start_button = ttk.Button(
                    root, text="Start Capture", command=start_streaming, width=60
                )
                # start_button.grid(row=13, column=0, columnspan=2, pady=10)
                start_button.grid(
                    row=13, column=0, columnspan=2, pady=10, sticky=""
                )

                # Initial call to set the correct state of the exposure and elevation
                AllSkyCam4OLEODL.update_entry(exposure_in, exposure_time_entry)
                AllSkyCam4OLEODL.update_entry(elevation_in, elevation_angle_entry)

                root.mainloop()


if __name__ == "__main__":
    main()
```

A.2 AllSkyCam4OLEODL Package

A.2.1 __init__.py

```python
# __init__.py

from .api import *
from .constants import *
from .gui import *
from .image_processing import *
from .input_checks import *
```

```python
from .link_budget import *
from .printer import *
```

A.2.2 api.py

```python
"""This module contains the API of the Allied Vision Goldeye Camera.

It has been modified in order to allow both streaming and recording. It
allows processing of the frames without the need of writting them first.
"""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\02_FINAL_PROJECT\api.py

from vmbpy import *  # noqa: F403

from typing import Optional
from datetime import datetime, timedelta
from tkinter import ttk
from matplotlib import dates as mdates
import sys
import os
import threading
import cv2
import math
import pytz
import tkinter as tk
import matplotlib.pyplot as plt
import matplotlib.animation as animation

from . import image_processing as im
from . import constants as const
from . import printer as pri

# All frames will either be recorded in this format, or transformed to it before being
↪ displayed
opencv_display_format = PixelFormat.Bgr8  # noqa: F405


def feature_changed_handler(feature) -> None:
    """API propietary printing fuction to indicate a changed of value of a feature.

    Args:
        feature (_type_): Feature to be changed.
    """
    msg = "Feature '{}' changed value to '{}'"
    print(msg.format(str(feature.get_name()), str(feature.get())), flush=True)


def abort(reason: str, return_code: int = 1, usage: bool = False) -> None:
    """API propietary exiting fuction and indicate an error.

    Args:
        reason (str): Reason to abort operation.
        return_code (int, optional): Error code raised. Defaults to 1.
        usage (bool, optional): Bool to check if an argument has been parsed. Defaults
        ↪ to False.
    """
    print(reason + "\n")
```

```python

    if usage:
        pri.print_usage()

    sys.exit(return_code)


def parse_args() -> Optional[str]:
    """API propietary fuction to parse an argument.

    Returns:
        Optional[str]: Parsed argument.
    """
    args = sys.argv[1:]
    argc = len(args)

    for arg in args:
        if arg in ("/h", "-h"):
            pri.print_usage()
            sys.exit(0)

    if argc > 1:
        abort(
            reason="Invalid number of arguments. Abort.",
            return_code=2,
            usage=True,
        )

    return None if argc == 0 else args[0]


def get_camera(camera_id: Optional[str]) -> Camera:  # noqa: F405
    """API propietary fuction to obtain the camera.

    Args:
        camera_id (Optional[str]): Specific camera we want to connect to.

    Returns:
        Camera: Camera we have connected to.
    """
    with VmbSystem.get_instance() as vmb:  # noqa: F405
        # ip = vmb.GevDeviceForceIPAddress.get()
        if camera_id:
            try:
                return vmb.get_camera_by_id(camera_id)

            except VmbCameraError:  # noqa: F405
                abort("Failed to access Camera '{}'. Abort.".format(camera_id))

        else:
            cams = vmb.get_all_cameras()
            if not cams:
                abort("No Cameras accessible. Abort.")

            return cams[0]


def setup_camera(
    cam: Camera,  # noqa: F405
    gain: tk.StringVar,
```

```python
    exposure: tk.StringVar,
    exposure_time_value: int,
    iso: tk.StringVar,
) -> None:
    """Configures the camera based on user inputs:

    1. Checks camera mode, if the own camera's background is chosen, it needs be
    ↪  enabled.

    2. Sets the gain mode, either 0dB or 18dB.

    3. Selects the exposure mode, Auto or Manual. If Manual, sets the exposure value.

    4. Enables white balancing if camera supports it.

    5. Adjusts GeV packet size (just for PoE camera).

    Args:
        cam (Camera): Camera object from the VMBPY API module.
        gain (tk.StringVar): Container of the gain mode (0 [0 dB] or 1 [18 dB]).
        exposure (tk.StringVar): Container of the exposure mode (Manual or Auto).
        exposure_time_value (int): Specified exposure value for Manual mode.
        iso (tk.StringVar): Container of the camera mode: Normal, Hot-pixel
        ↪  substraction, Subtraction or Camera's BC.
    """
    with cam:
        # print(cam.LUTEnable)
        # lut = cam.LUTEnable.get()
        # print(cam.LUTEnable.get())
        # # lut = True
        # lut = True
        # print(cam.LUTEnable)
        # print(lut)
        # cam.LUTEnable.set(lut)
        # print(cam.LUTEnable)
        # print(cam.LUTEnable)
        # cam.LUTEnable.value(True)
        # print(cam.LUTEnable)

        # Default settings
        cam.BCMode.set("Off")
        cam.IntegrationMode.set("IntegrateWhileRead")
        # cam.IntegrationMode.set("IntegrateThenRead")
        print(cam.IntegrationMode)

        # Background Correction settings
        if iso.get() == "Camera's BC":
            cam.BCMode.set("On")
            cam.BCIntegrationStart.run()
        print(cam.BCMode)

        # lut_enable = cam.LUTEnable
        # print(lut_enable)
        # lut_enable.set(True)
        # print(lut_enable)

        # Gain settings
        if gain.get() == "1 [18 dB]":
            try:
                cam.SensorGain.set("Gain1")
```

```python
            except (AttributeError, VmbFeatureError):  # noqa: F405
                pass
        else:
            try:
                cam.SensorGain.set("Gain0")
            except (AttributeError, VmbFeatureError):  # noqa: F405
                pass
        print(cam.SensorGain)

        # Exposure settings
        if exposure.get() == "Manual":
            try:
                cam.ExposureAuto.set("Off")
                exposure_time = cam.ExposureTime
                exposure_time.set(exposure_time_value)
            except (AttributeError, VmbFeatureError):  # noqa: F405
                pass
        else:
            try:
                cam.ExposureAuto.set("Continuous")
            except (AttributeError, VmbFeatureError):  # noqa: F405
                pass

        # White balancing settings
        try:
            cam.BalanceWhiteAuto.set("Continuous")
        except (AttributeError, VmbFeatureError):  # noqa: F405
            pass

        # GeV packet size settings (only available for GigE Cameras)
        try:
            stream = cam.get_streams()[0]
            stream.GVSPAdjustPacketSize.run()
            while not stream.GVSPAdjustPacketSize.is_done():
                pass
        except (AttributeError, VmbFeatureError):  # noqa: F405
            pass


def setup_pixel_format(cam: Camera) -> None:  # noqa: F405
    """Configures the camera's pixel format:

    1. Retrieves all the camera's compatible color pixel formats.

    2. Filters out formats not compatible with OpenCV.

    3. Retrieves all the camera's compatible monochrome pixel formats.

    4. Filters out formats not compatible with OpenCV.

    5. Selects the OpenCV-compatible color pixel format. If none exist, attempts
    to convert an incompatible format to be compatible. If conversion is not possible,
    selects an OpenCV-compatible monochrome pixel format.

    Args:
        cam (Camera): Camera object from the VMBPY API module.
    """
    # Available color pixel formats. Prefer color formats over monochrome formats
    cam_formats = cam.get_pixel_formats()
    cam_color_formats = intersect_pixel_formats(  # noqa: F405
```

```
        cam_formats,
        COLOR_PIXEL_FORMATS,  # noqa: F405
    )
    convertible_color_formats = tuple(
        f
        for f in cam_color_formats
        if opencv_display_format in f.get_convertible_formats()
    )

    cam_mono_formats = intersect_pixel_formats(cam_formats, MONO_PIXEL_FORMATS)  # noqa: F405
    convertible_mono_formats = tuple(
        f
        for f in cam_mono_formats
        if opencv_display_format in f.get_convertible_formats()
    )

    # Use OpenCV color format
    if opencv_display_format in cam_formats:
        cam.set_pixel_format(opencv_display_format)

    # Else convert color to OpenCV format
    elif convertible_color_formats:
        cam.set_pixel_format(convertible_color_formats[0])

    # Else use OpenCV monochrome format
    elif convertible_mono_formats:
        cam.set_pixel_format(convertible_mono_formats[0])

    else:
        abort("Camera does not support an OpenCV compatible format. Abort.")


def upload_lut(
    cam: Camera,  # noqa: F405
    lut_dataset_selector_index: int,
    gain: tk.StringVar,
) -> None:  # noqa: F405
    """Uploads and enables the LUT:

    1. Checks the gain to select the correct LUT path.

    2. Opens the LUT file, loads it into the camera and runs it.

    3. Prints the directory and selected LUT.

    Args:
        cam (Camera): Camera object from the VMBPY API module.
        lut_dataset_selector_index (int): Index of the selected LUT.
        gain (tk.StringVar): Container of the gain mode for choosing the LUT (0 [0 dB] or 1 [18 dB]).
    """
    # Set directory
    if gain.get() == "1 [18 dB]":
        dir = const.LUT_DIR1
    else:
        dir = const.LUT_DIR0

    # Read LUT, upload to camera
    with cam:
```

```python
        with open(dir, mode="rb") as file:
            fileContent = file.read()

            cam.LUTDatasetSelector.set(lut_dataset_selector_index)
            cam.LUTValueAll.set(fileContent)

            cam.LUTDatasetSave.run()

    print(
        f"LUT from file {dir} loaded into LUT Nr.{lut_dataset_selector_index}."
    )


def grab_frame(cam: Camera) -> None:  # noqa: F405
    """Captures a frame to be used as a temporal frame:

    1. Grabs a frame from the camera.

    2. Saves the frame in the specified directory.

    Args:
        cam (Camera): Camera object from the VMBPY API module.
    """
    frame = cam.get_frame()
    cv2.imwrite(const.BACKGROUND_FRAME_DIR, frame.as_opencv_image())


class Handler:
    """Handles the mayority of the camera operation.

    Methods:

        __init__:                       Initializes the Handler class.

        update:                         Updates the live graph of the GUI.

        save_plot:                      Saves the plot stored in the instance.

        create_camera_control_slider:   Creates an slider, saves it in the instance.

        set_exposure:                   Updates the exposure value.

        set_min_max_value:              Updates the minimum or maximum spot size value.

        __call__:                       Between each frame, sends the frame for
        processing, prepares for the next one, and checks if the program has stopped.
    """

    def __init__(
        self,
        cam: Camera,  # noqa: F405
        exposure_time_value: int,
        check: tk.StringVar,
        light: tk.StringVar,
        gain: tk.StringVar,
        iso: tk.StringVar,
        payload: tk.StringVar,
        elevation_in: tk.StringVar,
        fig: plt.figure,
        ax: plt.axes,
```

```
        line: plt.hlines,
        xdata: list,
        ydata: list,
    ):
        """Initializes the Handler class with the specified camera and plotting
        parameters, and creates a directory for storing the frames:

        1. Sets up all necessary instances based on the chosen settings.

        2. Creates the directory for saving frames according to the specified settings
        (will be saves in the data directory on the root folder of the project,
        inside a folder called tracking_images).

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            cam (Camera): Camera object from the VMBPY API module.
            exposure_time_value (int): Specified exposure value for Manual mode.
            check (tk.StringVar): Container of the streaming mode.
            light (tk.StringVar): Container of the time of day.
            gain (tk.StringVar): Container of the gain mode (0 [0 dB] or 1 [18 dB]).
            iso (tk.StringVar): Container of the camera mode: Normal, Hot-pixel substraction, Subtraction or Camera's BC.
            payload (tk.StringVar): Container of the payload.
            elevation_in (tk.StringVar): Container of the elevation mode.
            fig (plt.figure): Figure for the GUI plot.
            ax (plt.axes): Axes for the GUI plot.
            line (plt.hlines): Lines for the GUI plot
            xdata (list): Data for the x-axis for the GUI plot
            ydata (list): Data for the y-axis for the GUI plot

        :no-index:
        """
        # Setting up instances
        self.shutdown_event = threading.Event()
        self.cam = cam
        self.exposure_slider = None
        self.min_value = 1
        self.max_value = const.H_SENSOR_SIZE * const.V_SENSOR_SIZE
        self.root = None

        # Input instances
        self.gain = 1
        self.counter = 0
        self.background = 0
        self.mode = 0
        self.cond = 0

        # Graph instances
        self.fig = fig
        self.ax = ax
        self.line = line
        self.xdata = xdata
        self.ydata = ydata

        # Graph's update
        self.ani = animation.FuncAnimation(
            self.fig,
            self.update_graph,
            interval=100,
```

```python
            blit=False,
            cache_frame_data=False,
        )

        # Directory where the tracking frames get saved: directory of script.
        d = r".\data"

        self.payload = payload.get()
        # self.elevation_mode = elevation_in.get()
        if gain.get() == "1 [18 dB]":
            self.gain = 1
        else:
            self.gain = 0

        if iso.get() == "Normal":
            self.background = 0
        elif iso.get() == "Subtraction":
            self.background = 1
        elif iso.get() == "Camera's BC":
            self.background = 2
        else:
            self.background = 3

        if light.get() == "Daytime":
            self.cond = 0
        else:
            self.cond = 1

        if check.get() == "Record":
            # Directory creation
            self.mode = 1
            now = datetime.now()
            if exposure_time_value == 1:
                foldername_NP = (
                    f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                    f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                    f"exp_AUTO_GAIN{self.gain}_NP"
                )
                if self.background == 0:
                    foldername = (
                        f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                        f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}"
                        f"_exp_AUTO_GAIN{self.gain}"
                    )
                elif self.background == 1:
                    foldername = (
                        f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                        f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                        f"exp_AUTO_GAIN{self.gain}_IS"
                    )
                elif self.background == 2:
                    foldername = (
                        f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                        f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                        f"exp_AUTO_GAIN{self.gain}_CAMERA_BC"
                    )
                else:
```

```python
                foldername = (
                    f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                    f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                    f"exp_AUTO_GAIN{self.gain}_HP"
                )
        else:
            foldername_NP = (
                f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                f"exp_{exposure_time_value}us_GAIN{self.gain}_NP"
            )
            if self.background == 0:
                foldername = (
                    f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                    f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                    f"exp_{exposure_time_value}us_GAIN{self.gain}"
                )
            if self.background == 1:
                foldername = (
                    f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                    f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                    f"exp_{exposure_time_value}us_GAIN{self.gain}_IS"
                )
            if self.background == 2:
                foldername = (
                    f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                    f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                    f"exp_{exposure_time_value}us_GAIN{self.gain}_CAMERA_BC"
                )
            else:
                foldername = (
                    f"tc_{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_"
                    f"{now.strftime('%H')}{now.strftime('%m')}{now.strftime('%S')}_"
                    f"exp_{exposure_time_value}us_GAIN{self.gain}_HP"
                )

        if elevation_in.get() == "Full" and self.payload != "None":
            foldername = foldername + "_" + self.payload
            foldername_NP = foldername_NP + "_" + self.payload

        self.pnp = os.path.join(d, "tracking_images", foldername_NP)
        self.p = os.path.join(d, "tracking_images", foldername)

        os.makedirs(self.p)
        os.makedirs(self.pnp)

    def update_graph(self, frame: Frame) -> plt.hlines:  # noqa: F405
        """Updates the live graph of the GUI with new data stored in the current instance (self):

        1. If data is available, plots it on the graph.

        2. Adjusts the graph's horizontal and vertical limits, extends the x-axis by 60
        seconds, and increase the y-axis limit by 10% of the maximum value.

```

```
        3. Updates the graph's format as needed.

        4. Plots the new data on the graph.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            frame (Frame): Frame object from the VMBPY API module.

        Returns:
            plt.hlines: Updated graph instance.

        :no-index:
        """
        if not self.xdata:  # If no data, just return
            return (self.line,)

        # Ensure xdata is in UTC
        self.xdata = [x.astimezone(pytz.UTC) for x in self.xdata]

        self.line.set_data(self.xdata, self.ydata)
        ax = self.ax

        # Update axis limits
        ax.set_xlim(
            self.xdata[0],
            max(self.xdata[-1], self.xdata[0] + timedelta(seconds=60)),
        )

        if self.ydata:
            ax.set_ylim(0, max(255, max(self.ydata) * 1.1))

        # Update the formatter to show appropriate range
        locator = mdates.AutoDateLocator()
        # formatter = mdates.AutoDateFormatter(locator)
        formatter = mdates.DateFormatter("%H:%M:%S", tz=mdates.UTC)
        ax.xaxis.set_major_locator(locator)
        ax.xaxis.set_major_formatter(formatter)

        self.fig.canvas.draw()
        return (self.line,)

    def save_plot(self) -> None:
        """Saves the plot stored in the current instance (self):

        1. If recording mode is selected, sets the plot size, data, title, and labels.

        2. Adjust the format of the time as H:M:S.

        3. If a payload is chosen, its name will be added to the plot's title.

        4. Saves the plot in the same directory as the recorded frames (stored in the
        instance).

        Args:
            self (Instance): Current instance, provides access to attributes and methods.

        :no-index:
        """
```

```python
        if self.mode != 0:  # Only save if in recording mode
            plt.figure(figsize=(20, 8))
            plt.plot(self.xdata, self.ydata)

            # Enable the grid
            plt.grid(True)

            # Format the x-axis ticks
            time_format = mdates.DateFormatter("%H:%M:%S")
            plt.gca().xaxis.set_major_formatter(time_format)

            plt.xlabel("Time [UTC]")
            plt.ylabel("Brightness")
            plt.gcf().autofmt_xdate()  # Rotate and align the tick labels
            # Adjust layout manually for a tighter fit, but not as tight as plt.tightlayout()
            plt.subplots_adjust(left=0.05, right=0.95, top=0.95, bottom=0.1)
            # plt.tight_layout()
            if self.payload != "None":
                plt.title(
                    f"{self.payload} downlink on {datetime.now().strftime('%Y-%m-%d')}"
                )
                plt.savefig(
                    f"{self.p}/{datetime.now().strftime('%Y-%m-%d')}_{self.payload}_DL_plot.png"
                )
            else:
                plt.title(f"Downlink on {datetime.now().strftime('%Y-%m-%d')}")
                plt.savefig(
                    f"{self.p}/{datetime.now().strftime('%Y-%m-%d')}_DL_plot.png"
                )
            plt.close()

    def create_camera_control_slider(self, root: tk.Tk) -> None:
        """Creates an slider, saves it in the current instance (self) and sets its
        initial values:

        1. Creates a slider within the current instance using the CameraControlSlider
        class.

        2. Sets the initial values of the minimum and maximum spot size and the
        exposure with set_initial_values().

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            root (tk.Tk): Main window of the GUI menu.

        :no-index:
        """
        self.camera_control_slider = CameraControlSlider(root, self)
        current_exposure = self.cam.ExposureTime.get()
        self.camera_control_slider.set_initial_values(
            current_exposure, self.min_value, self.max_value
        )

    def set_exposure(self, exposure_time: int) -> None:
        """Updates the exposure value stored in the current instance (self).

        Args:
```

```python
            self (Instance): Current instance, provides access to attributes and methods.
            exposure_time (int): Exposure time value.
        :no-index:
        """
        try:  # Tries to update the exposure time
            self.cam.ExposureTime.set(exposure_time)
        except (AttributeError, VmbFeatureError):  # noqa: F405
            print("Failed to set exposure time")

    def set_min_max_value(self, value: int, siz: int) -> None:
        """Updates the minimum or maximum spot size value stored in the current
        instance (self).

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            value (int): Final minimum or maximum spot size value.
            siz (int): Integer to differenciate if we want to change the minimum or the maximum spot size value,
            1 for the maximum, 0 for minimum.

        :no-index:
        """
        if siz == 1:  # Max value
            self.max_value = value
            print(f"Max value set to: {value}")
        else:
            self.min_value = value
            print(f"Min value set to: {value}")

    def __call__(self, cam: Camera, stream: Stream, frame: Frame) -> None:  # noqa: F405
        """Between each frame, sends the current frame for processing, prepares for
        the next one, and checks if the program has stopped:

        1. If program shutdown is activated, exits the fuction.

        2. If a frame has been grabbed, increments the counter and starts image
        processing.

        3. Prepares for the next frame and checks if program is stopped (by
        pressing q / Q)

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            cam (Camera): Camera object from the VMBPY API module.
            stream (Stream): Stream object from the VMBPY API module.
            frame (Frame): Frame object from the VMBPY API module.

        :no-index:
        """
        if self.shutdown_event.is_set():
            return

        if frame.get_status() == FrameStatus.Complete:  # noqa: F405
            self.counter += 1
            # Image Processing fuction
            im.frame_processing(self, cam, frame)
```

```

            cam.queue_frame(frame)

            # Check for 'q' key press to stop recording
            if cv2.waitKey(1) & 0xFF == ord("q") or cv2.waitKey(1) & 0xFF == ord(
                "Q"
            ):
                self.shutdown_event.set()
                if self.root:
                    self.root.quit()


class CameraControlSlider(tk.Toplevel):
    """Handles the sliders of the GUI menu.

    Methods:

        __init__: Initializes the CameraControlSlider class.

        setup_exposure_slider: Creates the exposure slider.

        setup_min_value_slider: Creates the minimum spot size value slider.

        setup_max_value_slider: Creates the maximum spot size value slider

        update_exposure: Updates the exposure value.

        update_min_value: Updates the minimum spot size value.

        update_max_value: Updates the maximum spot size value.

        update_from_exposure_entry: Updates the exposure value parsed through the
        ↪   button.

        update_from_min_entry: Updates the minimum spot size value parsed through the
        ↪   button.

        update_from_max_entry: Updates the maximum spot size value parsed through the
        ↪   button.

        set_initial_values: Sets the intial values for all the varibales of the sliders.

    Args:
        tk (tk.Toplevel): Window where the slider should be created.
    """

    def __init__(self, master: tk.Tk, camera_control: Handler):
        """Initializes the CameraControlSlider class with the specified
        GUI parameters, and creates the sliders:

        1. Sets up all necessary instances for the slider elements in the menu.

        2. Configures exposure, minimum, and maximum spot size using the appropriate
        ↪   functions.

        Args:
            self (Instance): Current instance, provides access to attributes and
            ↪   methods.
            master (tk.Tk): Tinker GUI window.
            camera_control (Handler): Parsed through handler.
```

```python
        """
        # Allows the child class to invoke the constructor of its parent class.
        super().__init__(master)
        self.camera_control = camera_control
        self.title("Camera Control Sliders")
        # Dimensions adn position of the control sliders (WidthxHeight+X+Y).
        self.geometry("370x285+900+360")

        # Creates a frame to hold min and max sliders side by side
        self.min_max_frame = ttk.Frame(self)
        self.min_max_frame.pack(fill="x", expand=True, pady=10)

        # Slider setup
        # Exposure slider
        self.setup_exposure_slider()
        # Min value slider
        self.setup_min_value_slider()
        # Max value slider
        self.setup_max_value_slider()

    def setup_exposure_slider(self) -> None:
        """Creates the exposure slider:

        1. Initializes all slider elements and converts the exposure to a logarithmic
        ↪  scale.

        2. Sets up the displayed exposure value and its slider.

        3. Configures the button to manually change the exposure value.

        Args:
            self (Instance): Current instance, provides access to attributes and
            ↪  methods.
        """
        # Initial values are converted to a logarithmic scale
        self.min_exposure = 10
        self.max_exposure = 3000000  # 1 second
        self.log_min_exposure = math.log10(self.min_exposure)
        self.log_max_exposure = math.log10(self.max_exposure)

        self.current_exposure = tk.DoubleVar(value=self.log_min_exposure)
        ttk.Label(self, text="Exposure Control").pack(pady=(10, 0))

        self.exposure_label = ttk.Label(
            self, text=f"Exposure: {self.min_exposure:.2f} µs"
        )
        self.exposure_label.pack(pady=(0, 5))

        # Exposure slider settings
        self.exposure_slider = ttk.Scale(
            self,
            from_=0,
            to=1000,
            orient="horizontal",
            length=600,
            command=self.update_exposure,  # Calls the fuction on each value change
            variable=self.current_exposure,
        )
        self.exposure_slider.pack(pady=5, padx=20, fill="x")

```

```python
        self.exposure_entry = ttk.Entry(self, width=10)
        self.exposure_entry.pack(pady=5)
        self.exposure_entry.bind("<Return>", self.update_from_exposure_entry)

        self.exposure_set_button = ttk.Button(
            self, text="Set Exposure", command=self.update_from_exposure_entry
        )
        self.exposure_set_button.pack(pady=5)

    def setup_min_value_slider(self) -> None:
        """Creates the minimum spot size value slider:

        1. Initializes all slider elements and converts the minimum spot size value to a logarithmic scale.

        2. Sets up the displayed minimum value and its slider.

        3. Configures the button to manually change the minimum spot size value.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
        """
        self.min_value = 1
        self.max_value = const.H_SENSOR_SIZE * const.V_SENSOR_SIZE
        self.log_min_value = math.log10(self.min_value)
        self.log_max_value = math.log10(self.max_value)

        self.current_min = tk.DoubleVar(value=self.log_min_value)

        min_frame = ttk.Frame(self.min_max_frame)
        min_frame.pack(side="left", expand=True, fill="x", padx=10)

        ttk.Label(min_frame, text="Minimum Value Control").pack()
        self.min_label = ttk.Label(
            min_frame, text=f"Min. Beam Spot Size: {self.min_value:.0f}"
        )
        self.min_label.pack()

        self.min_slider = ttk.Scale(
            min_frame,
            from_=0,
            to=1000,
            orient="horizontal",
            command=self.update_min_value,
            variable=self.current_min,
        )
        self.min_slider.pack(fill="x")

        self.min_entry = ttk.Entry(min_frame, width=10)
        self.min_entry.pack()
        self.min_entry.bind("<Return>", self.update_from_min_entry)

        self.min_set_button = ttk.Button(
            min_frame, text="Set Min", command=self.update_from_min_entry
        )
        self.min_set_button.pack()

    def setup_max_value_slider(self) -> None:
        """Creates the maximum spot size value slider:
```

```

        1. Initializes all slider elements and converts the maximum spot size value to a
        ↪  logarithmic scale.

        2. Sets up the displayed maximum value and its slider.

        3. Configures the button to manually change the maximum spot size value.

        Args:
            self (Instance): Current instance, provides access to attributes and
            ↪  methods.
        """
        self.current_max = tk.DoubleVar(value=self.log_max_value)

        max_frame = ttk.Frame(self.min_max_frame)
        max_frame.pack(side="right", expand=True, fill="x", padx=10)

        ttk.Label(max_frame, text="Maximum Value Control").pack()
        self.max_label = ttk.Label(
            max_frame, text=f"Max Beam Spot Size: {self.max_value:.0f}"
        )
        self.max_label.pack()

        self.max_slider = ttk.Scale(
            max_frame,
            from_=0,
            to=1000,
            orient="horizontal",
            command=self.update_max_value,
            variable=self.current_max,
        )
        self.max_slider.pack(fill="x")

        self.max_entry = ttk.Entry(max_frame, width=10)
        self.max_entry.pack()
        self.max_entry.bind("<Return>", self.update_from_max_entry)

        self.max_set_button = ttk.Button(
            max_frame, text="Set Max", command=self.update_from_max_entry
        )
        self.max_set_button.pack()

    def update_exposure(self, event=None) -> None:
        """Updates the exposure value parsed through the slider:

        1. Gets the exposure from the slider and converts it to a logarithmic scale.

        2. Sends the updated exposure value to the camera.

        Args:
            self (Instance): Current instance, provides access to attributes and
            ↪  methods.
            event (_type_, optional): Nothing, just compatibility purposes. Defaults to
            ↪  None.
        """
        log_value = (
            self.exposure_slider.get()
            / 1000
            * (self.log_max_exposure - self.log_min_exposure)
            + self.log_min_exposure
```

```python
        )
        print(f"the log_value in update_exposure is {log_value}")
        print(
            f"the self.exposure_slider.get() in update_exposure is {self.exposure_slider.get()}"
        )
        print(
            f"the self.log_max_exposure in update_exposure is {self.log_max_exposure}"
        )
        print(
            f"the self.log_min_exposure in update_exposure is {self.log_min_exposure}"
        )
        exposure = round(10**log_value)
        self.exposure_label.config(text=f"Exposure: {exposure:.2f} µs")
        self.camera_control.set_exposure(exposure)
        self.exposure_entry.delete(0, tk.END)
        self.exposure_entry.insert(0, str(exposure))

    def update_min_value(self, event=None) -> None:
        """Updates the minimum spot size value parsed through the slider:

        1. Gets the minimum spot size value from the slider and converts it to a logarithmic scale.

        2. If the miniumum sleected spot size value is bigger than the maximum
        spot size value, the maximum spot size value is selected as the minimum
        spot size value. If a negative value is selected, it will be converted to 0.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            event (_type_, optional): Nothing, just compatibility purposes. Defaults to None.
        """
        log_value = (
            self.min_slider.get()
            / 1000
            * (self.log_max_value - self.log_min_value)
            + self.log_min_value
        )
        min_value = round(10**log_value)
        self.min_value = max(0, min(self.max_value - 1, min_value))
        self.min_label.config(text=f"Min Value: {self.min_value:.2f}")
        # Updates min value
        self.camera_control.set_min_max_value(self.min_value, 0)
        self.min_entry.delete(0, tk.END)
        self.min_entry.insert(0, str(self.min_value))

    def update_max_value(self, event=None) -> None:
        """Updates the maximum spot size value parsed through the slider:

        1. Gets the maximum spot size value from the slider and converts it to a logarithmic scale.

        2. The maximum selected spot size value will be the maximum value between
        the minimum spot size value and the smaller value between the maximum spot
        size value selected from the slider and whole dimensions of the image.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
```

```
            event (_type_, optional): Nothing, just compatibility purposes. Defaults to None.
        """
        log_value = (
            self.max_slider.get()
            / 1000
            * (self.log_max_value - self.log_min_value)
            + self.log_min_value
        )
        max_value = round(10**log_value)
        self.max_value = max(
            self.min_value,
            min((const.H_SENSOR_SIZE * const.V_SENSOR_SIZE), max_value),
        )
        self.max_label.config(text=f"Max Value: {self.max_value:.2f}")
        self.camera_control.set_min_max_value(self.max_value, 1)
        self.max_entry.delete(0, tk.END)
        self.max_entry.insert(0, str(self.max_value))

    def update_from_exposure_entry(self, event=None) -> None:
        """Updates the exposure value parsed through the button:

        1. Gets the exposure value and converts it to a logarithmic scale if it
        is in the correct range. If not, it raises an error.

        2. Updates the exposure value calling the update_exposure() function.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            event (_type_, optional): Nothing, just compatibility purposes. Defaults to None.

        Raises:
            ValueError: Exposure time inputed in the text-box is either bigger than the
            maximum exposure time or smaller than the smallest exposure time.
        """
        try:
            exposure = float(self.exposure_entry.get())
            if self.min_exposure <= exposure <= self.max_exposure:
                log_value = (
                    (math.log10(exposure) - self.log_min_exposure)
                    / (self.log_max_exposure - self.log_min_exposure)
                    * 1000
                )
                self.exposure_slider.set(log_value)
                # Updates the exposure value
                self.update_exposure(None)
            else:
                raise ValueError
        except ValueError:
            self.exposure_entry.delete(0, tk.END)
            self.exposure_entry.insert(
                0, str(round(10 ** self.current_exposure.get()))
            )

    def update_from_min_entry(self, event=None) -> None:
        """Updates the minimum spot size value parsed through the button:

        1. Gets the minimum spot size value and converts it to a logarithmic scale if
```

```python
        it is bigger than zero but smaller than the maximum value.

        2. Updates the minimum spot size value calling the update_min_value() function.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            event (_type_, optional): Nothing, just compatibility purposes. Defaults to None.

        Raises:
            ValueError: Minimum spot size value is either smaller than zero or bigger
            than the maximum spot size value.
        """
        try:
            min_value = int(self.min_entry.get())
            if 0 <= min_value < self.max_value:
                self.min_value = min_value
                log_value = (
                    (math.log10(max(1, min_value)) - self.log_min_value)
                    / (self.log_max_value - self.log_min_value)
                    * 1000
                )
                self.min_slider.set(log_value)
                # Updates the min spot value
                self.update_min_value(None)
            else:
                raise ValueError
        except ValueError:
            self.min_entry.delete(0, tk.END)
            self.min_entry.insert(0, str(self.min_value))

    def update_from_max_entry(self, event=None) -> None:
        """Updates the maximum spot size value parsed through the button:

        1. Gets the maximum spot size value and converts it to a logarithmic scale
        if it is bigger or equal to the minimum spot size values and smaller or equal
        than the full image.

        2. Updates the maximum spot size value calling the update_max_value() function.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            event (_type_, optional): Nothing, just compatibility purposes. Defaults to None.

        Raises:
            ValueError: Maximum spot size value is either bigger than the dimensions of
            the whole frame or smaller than the minimum spot size value.
        """
        try:
            max_value = int(self.max_entry.get())
            if max_value >= self.min_value and max_value <= (
                const.H_SENSOR_SIZE * const.V_SENSOR_SIZE
            ):
                self.max_value = max_value
                log_value = (
                    (math.log10(max_value) - self.log_min_value)
                    / (self.log_max_value - self.log_min_value)
```

```
                    * 1000
                )
                self.max_slider.set(log_value)
                # Updates the max spot value
                self.update_max_value(None)
            else:
                raise ValueError
        except ValueError:
            self.max_entry.delete(0, tk.END)
            self.max_entry.insert(0, str(self.max_value))

    def set_initial_values(
        self, exposure: float, min_value: int, max_value: int
    ) -> None:
        """Sets the intial values for all the varibales of the sliders.

        1. Gets the maximum and minimum spot size and exposure values and converts them to a logarithmic scale.

        2. Sets initial values of the sliders.

        3. Calls the update fuctions to set an inicial value for the variables.

        Args:
            self (Instance): Current instance, provides access to attributes and methods.
            exposure (float): Initial exposure time.
            min_value (int): Initial minimum spot size value.
            max_value (int): Initial maximum spot size value.
        """
        log_exposure = (
            (
                math.log10(
                    max(self.min_exposure, min(self.max_exposure, exposure))
                )
                - self.log_min_exposure
            )
            / (self.log_max_exposure - self.log_min_exposure)
            * 1000
        )
        self.min_value = min_value
        self.max_value = max_value
        log_min_value = (
            (math.log10(max(1, self.min_value)) - self.log_min_value)
            / (self.log_max_value - self.log_min_value)
            * 1000
        )
        log_max_value = (
            (math.log10(self.max_value) - self.log_min_value)
            / (self.log_max_value - self.log_min_value)
            * 1000
        )
        # Sets inital values of the sliders
        self.exposure_slider.set(log_exposure)
        self.min_slider.set(log_min_value)
        self.max_slider.set(log_max_value)
        # Sets inital values
        self.update_exposure(None)
        self.update_min_value(None)
```

```
        self.update_max_value(None)
```

A.2.3 constants.py

```
"""This module contains all the constants needed to allow the project to work
↪ correctly."""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\02_FINAL_PROJECT\constants.py

import numpy as np

# Directories
HOT_PIXEL0_DIR: str = r".\data\references\gain0\hot_500000.tiff"
HOT_PIXEL1_DIR: str = r".\data\references\gain1\hot_50000.tiff"
BACKGROUND_FRAME_DIR: str = r".\temp_.tiff"
TEMP_FRAME_DIR: str = r".\temp.tiff"

LUT_DIR0: str = r".\data\lut\Goldeye-G-CL-008_LinLUT_Gain0.bin"
LUT_DIR1: str = r".\data\lut\Goldeye-G-CL-008_LinLUT_Gain1.bin"

HOT_PIXEL0_THRES_VAL: int = 45
LUT_INDEX: int = 4

# Calibration factors
# CALIBRATION_THOR: float = 0.139602768 / 166
# CALIBRATION_THOR_GRID: float = 0.139602768 / 1426
# CALIBRATION_EDMU: float = 0.139602768 / 166
# MIN_BEAM_SIZE = 1
# MAX_BEAM_SIZE = 10

GRID_RADIUS: int = 2

# Lens parameters
LENS_FOCAL_LENGTH: float = 3.5  # mm
V_SENSOR_SIZE: int = 256  # Pixels
H_SENSOR_SIZE: int = 320  # Pixels
PIXEL_SIZE: float = 0.03  # mm (30um)

FISH_CENTER = [161, 131]

K = np.array(
    [
        [123.48774151225143, 0.0, 160.36138044001243],
        [0.0, 123.21716405041697, 130.60363126947752],
        [0.0, 0.0, 1.0],
    ]
)
D = np.array(
    [
        [-0.04444806603723004],
        [0.003674670475966278],
        [0.003063454507676400 2],
        [-0.0019588697364114325],
    ]
)


######################  LINK BUDGET  ######################
```

```python
# Fixed constants
C = 3e8  # m/s - speed of light
H = 6.626e-34  # m²-kg/s - Planck's constant
R_E = 6370e3  # m - Earth's radius

# Environment constants
PSI = 0.3
# PSI = 0  # a change according to el is not yet regarded
# PSI = 0.1
# PSI = 1E-8
P_THR = 0.1  # loss_fraction for ScintiLoss

# Satellite constants
# el = 15  # ° - Elevation of the satellite
# P_tx = 30  # dBm - Transmited power
# 1W mean was used in FLP-OSIRISv1 experiments with OCAM
# 100mW or 50mW mean we expect in KIODO, since the 20dBm mentioned in the book might be peak-power
```

A.2.4 gui.py

```python
"""This module manages the GUI settings menu that appears when the program is first run."""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\02_FINAL_PROJECT\gui.py

from tkinter import ttk
from typing import Tuple
import tkinter as tk


def update_entry(variable: tk.StringVar, entry: tk.ttk.Entry) -> None:
    """Checks for the parsed state of a variable to enable
    or disable its text box.

    Args:
        variable (tk.StringVar): Container of the variable to check.
        entry (tk.ttk.Entry): Container of the value and its state.
    """
    if variable.get() == "Auto" or variable.get() == "Full":
        # If exposure is automatic, the button will be greyed-out
        entry.config(state="disabled")
    else:
        entry.config(state="normal")


def create_menu() -> (
    Tuple[
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
        tk.StringVar,
```

```
        tk.StringVar,
        tk.Tk,
        tk.ttk.Entry,
        tk.ttk.Entry,
    ]
):
    """Creates the GUI menu to configure the initial settings:

    1. Creates the variables windows with an specific size and position.

    2. Creates varibales to store the inputs.

    3. Creates dropdown menus with default values.

    4. Defines a lambda function to parse the exposure and elevation varibales to update_time_entry().

    Returns:
        tuple: gain_var (tk.StringVar): Container of the gain mode (0 [0 dB] or 1 [18 dB]).

        check_var (tk.StringVar): Container of the streaming mode (Stream or Record).

        light_var (tk.StringVar): Container of the time of the day (Daytime or Nighttime).

        payload_var (tk.StringVar): Container of the payload used (None, KIODO,
        OsirisV1, Osiris4CubeSat or CubeCat).

        h_ogs_var (tk.StringVar): Container of the height of the OGS used (IKN-OP or GSOC-OP).

        zenith_var (tk.StringVar): Container of the zenith attenuation (Bad
        1550nm [0.891], Good 1550nm [0.986], Bad 850nm [0.705], Good 850nm
        [0.950] or CubeCat 20240822 [0.963])

        elevation_var (tk.StringVar): Container of the elevation mode (Individual or Full).

        elevation_angle_var (tk.StringVar): Container of the elevation angle (if manual).

        exposure_var (tk.StringVar): Container of the exposure mode (Auto or Manual).

        exposure_time_var (tk.StringVar): Container of the exposure value (if manual).

        iso_var (tk.StringVar): Container of the main camera mode (Normal, Hot-pixel
        substraction, Subtraction ormCamera's BC).

        root (tk.Tk): Main window of the GUI menu.

        exposure_time_entry (tk.ttk.Entry): Value of the exposure.

        elevation_angle_entry (tk.ttk.Entry): Value of the elevation.
    """
    root = tk.Tk()
    root.title("IR Camera for Satellite Tracking")
    # Dimensions and position of the settings menu (WidthxHeight+X+Y).
    root.geometry("370x320+900+5")

```

```python
    # Variables
    gain_var = tk.StringVar(value="1")
    check_var = tk.StringVar(value="S")
    light_var = tk.StringVar(value="D")
    payload_var = tk.StringVar(value="")
    h_ogs_var = tk.StringVar(value="IKN-OP")
    zenith_var = tk.StringVar(value="B_1")
    elevation_var = tk.StringVar(value="F")
    elevation_angle_var = tk.StringVar(value="")
    exposure_var = tk.StringVar(value="M")
    exposure_time_var = tk.StringVar(value="")
    iso_var = tk.StringVar(value="N")

    # Camera title
    ttk.Label(text="Camera settings", font=(14)).grid(
        row=0, column=0, pady=5, sticky=""
    )
    # LB title
    ttk.Label(text="LB Settings", font=(14)).grid(
        row=0, column=1, columnspan=1, pady=5
    )

    # Dropdown menus
    ttk.Label(text="Gain:").grid(row=1, column=0, sticky="")
    gain_combo = ttk.Combobox(
        textvariable=gain_var, values=["0 [0 dB]", "1 [18 dB]"]
    )
    gain_combo.grid(row=2, column=0)
    gain_combo.set("1 [18 dB]")

    ttk.Label(text="Payload:").grid(row=1, column=1, sticky="")
    payload_combo = ttk.Combobox(
        textvariable=payload_var,
        values=[
            "None",
            "KIODO",
            "OsirisV1",
            "Osiris4CubeSat",
            "CubeCat",
        ],
    )
    payload_combo.grid(row=2, column=1)
    payload_combo.set("None")

    ttk.Label(text="Capture Mode:").grid(row=3, column=0, sticky="")
    check_combo = ttk.Combobox(
        textvariable=check_var, values=["Stream", "Record"]
    )
    check_combo.grid(row=4, column=0)
    check_combo.set("Stream")

    ttk.Label(text="OGS:").grid(row=3, column=1, sticky="")
    h_ogs_combo = ttk.Combobox(
        textvariable=h_ogs_var, values=["IKN-OP", "GSOC-OP"]
    )
    h_ogs_combo.grid(row=4, column=1)
    h_ogs_combo.set("IKN-OP")

    ttk.Label(text="Zenith-attenuation:").grid(row=5, column=1, sticky="")
    zenith_combo = ttk.Combobox(
```

```
        textvariable=zenith_var,
        values=[
            "Bad 1550nm [0.891]",
            "Good 1550nm [0.986]",
            "Bad 850nm [0.705]",
            "Good 850nm [0.950]",
            "CubeCat 20240822 [0.963]",
        ],
    )
    zenith_combo.grid(row=6, column=1)
    zenith_combo.set("Bad 1550nm [0.891]")

    ttk.Label(text="Time of the day:").grid(row=5, column=0, sticky="")
    light_combo = ttk.Combobox(
        textvariable=light_var, values=["Daytime", "Nighttime"]
    )
    light_combo.grid(row=6, column=0)
    light_combo.set("Daytime")

    ttk.Label(text="Elevation:").grid(row=7, column=1, sticky="")
    elevation_combo = ttk.Combobox(
        textvariable=elevation_var, values=["Full", "Individual"]
    )
    elevation_combo.grid(row=8, column=1)
    elevation_combo.set("Full")

    ttk.Label(text="Elevation Angle (°):").grid(row=9, column=1, sticky="")
    elevation_angle_entry = ttk.Entry(textvariable=elevation_angle_var)
    elevation_angle_entry.grid(row=10, column=1)

    ttk.Label(text="Exposure:").grid(row=7, column=0, sticky="")
    exposure_combo = ttk.Combobox(
        textvariable=exposure_var, values=["Auto", "Manual"]
    )
    exposure_combo.grid(row=8, column=0)
    exposure_combo.set("Manual")

    ttk.Label(text="Exposure Time (µs):").grid(row=9, column=0, sticky="")
    exposure_time_entry = ttk.Entry(textvariable=exposure_time_var)
    exposure_time_entry.grid(row=10, column=0)

    ttk.Label(text="Tecnique Used:").grid(row=11, column=0, sticky="")
    iso_combo = ttk.Combobox(
        textvariable=iso_var,
        values=[
            "Normal",
            "Hot-pixel substraction",
            "Subtraction",
            "Camera's BC",
        ],
    )
    iso_combo.grid(row=12, column=0)
    iso_combo.set("Normal")

    # lambda to parse exposure and elevation states
    exposure_var.trace_add(
        "write",
        lambda *args: update_entry(exposure_var, exposure_time_entry),
    )

```

```python
    elevation_var.trace_add(
        "write",
        lambda *args: update_entry(elevation_var, elevation_angle_entry),
    )

    return (
        gain_var,
        check_var,
        light_var,
        payload_var,
        h_ogs_var,
        zenith_var,
        elevation_var,
        elevation_angle_var,
        exposure_var,
        exposure_time_var,
        iso_var,
        root,
        exposure_time_entry,
        elevation_angle_entry,
    )
```

A.2.5 image_processing.py

```python
"""This module manages the processing of the frames taken by the camera."""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\02_FINAL_PROJECT\image_processing.py

from datetime import datetime
from typing import Tuple
import cv2
import os
import csv
import math
import numpy as np

from . import constants as const


def dark_frame_setup(frame: np.array) -> np.array:
    """Prepares a frame for processing:

    1. A threshold of 45 is applied to a previously recorded image of the camera's hot
    ↪ pixels.

    2. The resulting thresholded image is normalized, resulting in an image with values
    ↪ [0, 1].

    3. Finally the normalized image is scaled, obtaining an image with values [0, 255].

    Args:
        frame (np.array): Frame with the hot pixels present.

    Returns:
        np.array: Image after normalization, thresholding and scaling.
    """
    # Apply threshold to create a binary image
    _, thresh = cv2.threshold(
```

```python
        frame, const.HOT_PIXEL0_THRES_VAL, 255, cv2.THRESH_BINARY
    )

    # Normalize the binary image
    normalized = cv2.normalize(
        thresh, None, 0, 1.0, cv2.NORM_MINMAX, dtype=cv2.CV_32F
    )
    # Scale the image to [0, 255]
    normalized = normalized * 255
    normalized = normalized.astype(np.uint8)

    return normalized


def subtract_frames(frame: np.array, frame_substracted: np.array) -> np.array:
    """Subtracts one frame from another using OpenCV:

    1. Checks if both frames have the same dimensions and format.

    2. Substracts both of the frames.

    Args:
        frame (np.array): Original frame.
        frame_substracted (np.array): Frame to substract.

    Returns:
        np.array: Image obtained after frame subtraction.
    """
    # Ensure images are the same size and format
    assert (
        frame.shape == frame_substracted.shape
    ), "Both image frames must have the same dimensions"
    assert (
        frame.dtype == frame_substracted.dtype
    ), "Both image frames must have the same data type"

    # Subtract the dark frame using OpenCV
    subtracted_image = cv2.subtract(frame, frame_substracted)

    return subtracted_image


def write_csv(
    self,
    frame_mean_pixel_value: int,
    frame_number: str,
    time: str,
    exposure: float,
    r: float,
    elevation: float,
    azimuth: float,
    fov: float,
    location: Tuple,
    pvalue: float,
    pvalue_grid: np.uint32,
    intensity: float,
    intensity_grid: np.float64,
) -> None:
    """Writes a csv file with all the parameters needed to perform the analysis of the
    ↪ final satellite pass:
```

```python

    1. Creates the dictionaries and fields.

    2. Writes the csv file.

    Args:
        self (Instance): Parsed instance from Handler in the api module, provides access to attributes and methods.
        frame_mean_pixel_value (int): Mean Pixel Value of the entire frame.
        frame_number (str): Frame number.
        time (str): Current time where the frame as been recorded.
        exposure (float): Exposure time used for that particular frame.
        r (float): Radial position of the brightest point.
        elevation (float): Elevation of the brightest point
        azimuth (float): Azimuth of the brightest point.
        fov (float): Field Of View of the camera, in degrees, at the brightest point
        location (tuple): Location of the brightest point: [0] = x-axis, [1] = y-axis.
        pvalue (float): Pixel Value of the brightest point [0-255].
        pvalue_grid (np.uint32): Pixel Value of the whole brightest contour.
        intensity (float): Intensity value of the brightest point.
        intensity_grid (np.float64): Intensity value of the whole brightest contour.
    """
    if self.gain == 1:
        gain = "Gain 1 [18dB]"
    else:
        gain = "Gain 0 [0dB]"
    exposure = exposure.get()
    mydict = [
        {
            "Frame": frame_number,
            "Gain Mode": gain,
            "Time [CEST]": time,
            "Exposure [us]": exposure,
            "Location [x, y]": location,
            "Elevation [°]": elevation,
            "Azimuth [°]": azimuth,
            "FOV": fov,
            "R": r,
            "Mean Pixel Value [DN]": frame_mean_pixel_value,
            "Brightest Pixel Value [DN]": pvalue,
            "Grid Brightest Pixel Value [DN]": pvalue_grid,
            "Intensity [uW/m^-2]": intensity,
            "Grid Intensity[uW/m^-2]": intensity_grid,
        }
    ]

    fields = [
        "Frame",
        "Gain Mode",
        "Time [CEST]",
        "Exposure [us]",
        "Location [x, y]",
        "Elevation [°]",
        "Azimuth [°]",
        "FOV",
        "R",
        "Mean Pixel Value [DN]",
        "Brightest Pixel Value [DN]",
        "Grid Brightest Pixel Value [DN]",
        "Intensity [uW/m^-2]",
```

```python
        "Grid Intensity[uW/m^-2]",
    ]

    if self.payload != "None":
        filename = f"{self.p}/{datetime.now().strftime('%Y-%m-%d')}_{self.payload}_DL_csv.csv"
    else:
        filename = f"{self.p}/{datetime.now().strftime('%Y-%m-%d')}_DL_csv.csv"
    file_exists = os.path.isfile(filename)

    # Write csv
    with open(filename, "a", newline="") as csvfile:
        writer = csv.DictWriter(csvfile, fieldnames=fields)
        if not file_exists:
            writer.writeheader()
        writer.writerows(mydict)


def brightest_V2(
    self,
    frame: np.array,
    exposure: float,
) -> Tuple[Tuple, float, float, float, float, float]:
    """Calculates the brightest point of the frame based on an specified minimum and maximum spot size.

    1. Setups the calibration factor.

    2. Depending if the daylight or nighttime is selcted a different process will be applyed:

        In case of Daytime: Bilateral filter 3 200x200 -> OTSU Thresholding.

        In case of Nighttime: Gaussian blur 3x3 -> Threshold based on black values.

    3. Finds the contours of the figure (zones of the fram with similar pixel values).

    4. Bounds the contours and filters them based on the minimum and maximum spot sizes values.

    5. Select the contour with the highest mean pixel value.

    6. Obtains the brightest pixel and the sum of all the values from the brightest contour.

    7. Converts pixel value to intensity using the correction factor.

    Args:
        self (Instance): Current instance, provides access to attributes and methods.
        frame (np.array): Frame from where the brightest point will be obtained.
        exposure (float): Exposure time used for that particular frame.

    Returns:
        tuple[tuple, float, float, float, float, float]: max_loc (tuple):Location of the brightest point:
        [0] = x-axis, [1] = y-axis.

        max_val (float): Pixel value of the brightest point [0-255].

        intensity_brightest (float): Intensity value of the brightest point.
```

```python

            max_mean_pixel_value (float): Mean pixel value of the whole brightest contour.

            sum_max_mean_pixel_value[0] (float): Summed pixel value of the whole brightest contour.

            intensity_brightest_grid (float): Summed intensity value of the whole brightest contour.
        """
        max_mean_pixel_value = 0
        max_val = 0
        max_loc = [0, 0]
        intensity_brightest = 0
        intensity_brightest_grid = 0
        sum_max_mean_pixel_value = [0, 0, 0]
        # sum_max_mean_pixel_value[0]

        exposure = exposure.get()  # Exposure in us

        # Calibration factors
        calibration_THOR = 0.139602768 / (0.0001 * exposure + 18.545)
        calibration_THOR_grid = 0.139602768 / (0.0008 * exposure + 443.22)
        if self.gain == 1:
            calibration_THOR_gain = 0.139602768 / (1.4788 * exposure - 82.62)
        else:
            calibration_THOR_gain = 0.139602768 / (0.2427 * exposure + 1.67)

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        if self.cond == 0:
            # Processing for Daytime - Bilateral filter and otsu thresholding
            # filter = cv2.bilateralFilter(gray, 3, 100, 100)
            filter = cv2.bilateralFilter(gray, 3, 200, 200)
            # filter = cv2.GaussianBlur(gray, (5, 5), 0)
            # filter = cv2.GaussianBlur(gray, (3, 3), 0)

            _, th = cv2.threshold(
                filter, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU
            )
        else:
            # Processing for Nighttime - Gaussian filter and thresholding based on dark pixels pixel value
            thresh_val = 0.0004 * exposure + 13.113

            filter = cv2.GaussianBlur(gray, (3, 3), 0)
            _, th = cv2.threshold(filter, thresh_val, 255, cv2.THRESH_BINARY)

        contours, _ = cv2.findContours(th, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

        min_size = self.min_value
        max_size = self.max_value

        for contour in contours:
            _, _, w, h = cv2.boundingRect(contour)
            # Filter contours based on min and max spot size
            if min_size <= w * h and max_size >= w * h:
                mask = np.zeros(gray.shape, np.uint8)
                cv2.drawContours(mask, [contour], 0, 255, -1)

                mean_pixel_value = cv2.mean(gray, mask=mask)[0]

```

```
            # Compares current contour with the brightest one
            if mean_pixel_value > max_mean_pixel_value:
                max_mean_pixel_value = mean_pixel_value

                # Obtain brightest point and full brightness of the contour
                masked_gray = cv2.bitwise_and(gray, gray, mask=mask)
                sum_max_mean_pixel_value = cv2.sumElems(masked_gray)
                _, max_val, _, max_loc = cv2.minMaxLoc(masked_gray)

        # Convert to intensity
        intensity_brightest = max_val * calibration_THOR
        intensity_brightest_grid = (
            sum_max_mean_pixel_value[0] * calibration_THOR_grid
        )

    return (
        max_loc,
        max_val,
        intensity_brightest,
        max_mean_pixel_value,
        sum_max_mean_pixel_value[0],
        intensity_brightest_grid,
    )


def calculate_el_azi(max_loc: Tuple) -> Tuple[float, float, float, float]:
    """Calculates the elevation and azimuth of the brightest point of the frame,
    making  use of the fisheye projections. Equidistant, equisolid and stereographic
    can be selected.

    1. Calculates the distance from the center of the image to the brightest point.

    2. Applies the projection to obtain the fov of the lens in the brightest point.

    3. Obtains the elevation based on the fov.

    4. Calculates the azimuth based on the center of the lens.

    Args:
        max_loc (tuple): Location of the brightest point: [0] = x-axis, [1] = y-axis.

    Returns:
        tuple[float, float, float, float]: elevation (float): Elevation of the brightest point.

        fov (float): Field Of View of the camera, in degrees, at the brightest point.

        r (float): Radial position of the brightest point.

        azimuth (float): Azimuth of the brightest point.
    """
    # print(max_loc)
    # print(type(max_loc))
    # max_loc = np.array([[[max_loc[0], max_loc[1]]]], dtype=np.float32)
    # # Undistort the point
    # undist_max_loc = cv2.fisheye.undistortPoints(max_loc, const.K, const.D, P=const.K)
    # undist_max_loc = undist_max_loc[0][0]
    # max_loc = (int(round(undist_max_loc[0])), int(round(undist_max_loc[1])))
    # print(max_loc)

```

```python
    F = const.LENS_FOCAL_LENGTH
    r = math.sqrt(
        (max_loc[0] - const.FISH_CENTER[0]) ** 2
        + (max_loc[1] - const.FISH_CENTER[1]) ** 2
    )

    # Lens Projections
    # EQUIDISTANT PROJECTION
    # fov_rad = (r * const.PIXEL_SIZE) / F  # rad - FOV in radians
    # EQUISOLID PROJECTION
    fov_rad = 2 * math.asin((r * const.PIXEL_SIZE) / (2 * F))
    # STEREOGRAPHIC PROJECTION
    # fov_rad = 2 * math.atan((r * const.PIXEL_SIZE) / (2 * F))
    # RECTILINEAR PROJECTION
    # fov_rad = math.atan((r * const.PIXEL_SIZE) / F)

    fov = (fov_rad * (180 / math.pi)) * 2  # ° - FOV in degrees
    elevation = (180 - fov) / 2

    # Azimuth at the exact center of the frame
    if max_loc[1] == const.FISH_CENTER[1]:
        azimuth = 0
    else:
        azimuth = (
            math.atan(
                (max_loc[0] - const.FISH_CENTER[0])
                / (max_loc[1] - const.FISH_CENTER[1])
            )
        ) * (180 / math.pi)

    # If point is in the upper part of the frame
    if max_loc[1] >= const.FISH_CENTER[1]:
        azimuth = 180 + azimuth
    else:
        # If point is the bottom-right part of the frame
        if max_loc[0] >= const.FISH_CENTER[0]:
            azimuth = 360 + azimuth

    return elevation, fov, r, azimuth


def frame_draw(
    frame: np.array,
    time: str,
    exposure: float,
    rad: int,
    max_bright_loc: Tuple,
    max_bright_val: float,
    int_bright_val: float,
    mean_bright_grid_val: np.uint32,
    max_bright_grid_val: np.uint32,
    int_bright_grid_val: np.float64,
    min_size: int,
    max_size: int,
    elevation: float,
    azimuth: float,
) -> None:
    """Draws the overlays on top of the frame:

    Args:
```

```
        frame (np.array): Frame where the overlays will be drawn.
        time (str): Actual time in that particular frame.
        exposure (float): Exposure time used for that particular frame.
        rad (int): Radius of the intensity grid.
        max_bright_loc (tuple): Location of the brightest point: [0] = x-axis, [1] = y-axis.
        max_bright_val (float): Pixel value of the brightest point [0-255].
        int_bright_val (float): Intensity value of the brightest point.
        mean_bright_grid_val (np.uint32): Mean pixel value of the whole brightest contour.
        max_bright_grid_val (np.uint32): Summed pixel value of the whole brightest contour.
        int_bright_grid_val (np.float64): Summed intensity value of the whole brightest contour.
        min_size (int): Minimum spot size value used.
        max_size (int): Maximum spot size value used.
        elevation (float): Elevation of the brightest point.
        azimuth (float): Azimuth of the brightest point.
    """
    # Draw the time, exposure, coordinates and crosshair on the frame
    cv2.putText(
        frame,
        time,
        (1, 7),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.3,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        f"Exposure: {exposure.get()} us",
        (1, 38),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        f"Min Size: {min_size:.0f}",
        (1, 47),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        f"Max Size: {max_size:.0f}",
        (1, 56),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        f"El.: {elevation:.0f}",
        (1, 201),
```

```
            cv2.FONT_HERSHEY_SIMPLEX,
            0.25,
            (255, 255, 255),
            1,
        )
        cv2.putText(
            frame,
            f"Az.: {azimuth:.0f}",
            (1, 210),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.25,
            (255, 255, 255),
            1,
        )
        cv2.putText(
            frame,
            f"Brightness at {max_bright_loc}: {int_bright_val:.3f} uw/m^2 ->
            ↪ {max_bright_val}",
            (1, 238),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.25,
            (255, 255, 255),
            1,
        )
        if max_bright_val == 255:
            cv2.putText(
                frame,
                "Saturated! Lower Exposure",
                (115, 20),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.25,
                (255, 255, 255),
                1,
            )
        cv2.putText(
            frame,
            f"Brightness {rad*2+1}x{rad*2+1} grid: {int_bright_grid_val:.3f} uw/m^2 "
            f"-> {max_bright_grid_val} (mean: {mean_bright_grid_val:.1f})",
            (1, 245),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.25,
            (255, 255, 255),
            1,
        )
        cv2.putText(
            frame,
            "N",
            (160, 7),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.4,
            (255, 255, 255),
            1,
        )
        cv2.putText(
            frame,
            "S",
            (160, 254),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.4,
            (255, 255, 255),
```

```
        1,
    )
    cv2.putText(
        frame,
        "E",
        (1, 128),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.4,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "W",
        (310, 128),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.4,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "SW",
        (290, 242),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.4,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "(225)",
        (280, 227),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "(248)",
        (295, 190),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "NW",
        (290, 17),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.4,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "(338)",
        (225, 10),
```

```
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "(315)",
        (280, 29),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "(293)",
        (295, 66),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "SE",
        (30, 242),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.4,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "(135)",
        (35, 227),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "(113)",
        (11, 190),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.25,
        (255, 255, 255),
        1,
    )
    cv2.putText(
        frame,
        "NE",
        (30, 17),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.4,
        (255, 255, 255),
        1,
    )
    cv2.putText(
```

```python
            frame,
            "(45)",
            (35, 29),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.25,
            (255, 255, 255),
            1,
        )
        cv2.putText(
            frame,
            "(68)",
            (11, 66),
            cv2.FONT_HERSHEY_SIMPLEX,
            0.25,
            (255, 255, 255),
            1,
        )

        # Circle for the brightest point
        cv2.circle(frame, max_bright_loc, 5, (255, 255, 255), 1)

        # Crosshair
        # cv2.line(frame, (159, 128), (161, 128), (255, 255, 255), 1)
        # cv2.line(frame, (160, 127), (160, 129), (255, 255, 255), 1)


    def frame_processing(self, cam, frame) -> None:
        """Procceses the frame.

        1. Grabs a temporal frame.

        2. Depending on the selected mode by the user:

        - Hot-pixel removal.
            A frame with the hot pixels will threshold and normalized by the
            dark_frame_setup() fuction and then subtracted to the the taken frame
            with the subtract_frames() function.

        - Own background correction.
            Substracts the temporal frame to the next grabbed frame. The
            subtract_frames() fuction is applied for substracting the temporal frame,
            just grabbed, with the next frame.

        - Normal operation.
            The temporal frame will be used directly.

        - Camera's own background correction.
            The temporal frame will be used directly.

        3. Obtains the brightest point thanks to the brightest_V2() function.

        4. Obtains the elevation and azimuth of the brightest pixel with the calculate_el_azi() fuction.

        5. Draws all the desired values on top of the frame using the frame_draw() fuction.

        6. Just in case the Record mode is being used, both the processed and unprocessed frames,
        besides the csv file, will all be saved.

```

```python
    7- Finally the frames will be display and the first temporal frame will be removed.

    Args:
        self (Instance): Current instance, provides access to attributes and methods.
        cam (Camera): Camera object from the VMBPY API module.
        frame (Frame): Frame object from the VMBPY API module.
    """
    # Get the current time with ms.
    now = datetime.now()
    current_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S.%f")[:-3]
    exposure_time = cam.ExposureTime
    msg = "Stream from '{}'. Press <q> to stop stream."

    print("{} acquired {}".format(cam, frame), flush=True)

    # Create a temporal frame where the proccessing will be made.
    cv2.imwrite(const.TEMP_FRAME_DIR, frame.as_opencv_image())
    frame_temp = cv2.imread(const.TEMP_FRAME_DIR)

    # Hot-pixel removal
    if self.background == 3:
        # Setup of the hot pixels of the wanted exposure.
        hot = cv2.imread(const.HOT_PIXEL0_DIR)
        normalized = dark_frame_setup(hot)
        # Removal of the dark pixels from the taken frame.
        frame_subs = subtract_frames(frame_temp, normalized)
    # Background substraction.
    elif self.background == 1:
        normalized = cv2.imread(const.BACKGROUND_FRAME_DIR)
        # background = background.astype(np.uint8)
        frame_subs = subtract_frames(frame_temp, normalized)
        # frame_subs = subtract_frames(normalized, frame_temp)  #   Chroma effect
    # Normal operation or Background correction mode
    else:
        frame_subs = cv2.imread(const.TEMP_FRAME_DIR)

    # Calculate the brightest point, elevation and azimuth
    mean_val_grid = 0
    (
        max_loc,
        max_val,
        intensity_brightest,
        mean_val_grid,
        max_val_grid,
        intensity_brightest_grid,
    ) = brightest_V2(self, frame_subs, exposure_time)

    elevation, fov, r, azimuth = calculate_el_azi(max_loc)

    # Append data for update the live graph of the GUI
    self.xdata.append(now)
    self.ydata.append(max_val)

    # Draw on-top of the frame
    frame_draw(
        frame_subs,
        current_time,
        exposure_time,
        const.GRID_RADIUS,
        max_loc,
```

```
            max_val,
            intensity_brightest,
            mean_val_grid,
            max_val_grid,
            intensity_brightest_grid,
            self.min_value,
            self.max_value,
            elevation,
            azimuth,
        )

        # We only write the frame if the mode selected is Record.
        if self.mode != 0:
            cv2.imwrite(
                f"{self.p}/{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_{now.strftime('%H')}"
                f"{now.strftime('%M')}{now.strftime('%S')}_frame_{str(self.counter)}.tiff",
                frame_subs,
            )
            cv2.imwrite(
                f"{self.pnp}/{now.strftime('%Y')}{now.strftime('%m')}{now.strftime('%d')}_{now.strftime('%H')}"
                f"{now.strftime('%M')}{now.strftime('%S')}_frame_{str(self.counter)}.tiff",
                frame_temp,
            )

            # Calculate mean pixel value of the frame and generate csv
            frame_mpv = frame_subs.mean()
            frame_num = f"frame_{str(self.counter)}"
            write_csv(
                self,
                frame_mpv,
                frame_num,
                current_time,
                exposure_time,
                r,
                elevation,
                azimuth,
                fov,
                max_loc,
                max_val,
                max_val_grid,
                intensity_brightest,
                intensity_brightest_grid,
            )

        # Create the window for the frames
        window = np.concatenate((frame_subs, frame_temp), axis=1)
        winname = msg.format(cam.get_name())
        cv2.namedWindow(winname)
        cv2.moveWindow(winname, 70, 5)
        cv2.imshow(winname, window)  # Show the frame

        if os.path.isfile(const.TEMP_FRAME_DIR):
            os.remove(const.TEMP_FRAME_DIR)
```

A.2.6 input_checks.py

```python
"""This modules manages the checking and validation of the user inputs."""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\02_FINAL_PROJECT\input_checks.py

from typing import Tuple

def checks(
    elevation_in,
    elevation_angle_in,
    exposure_in,
    exposure_time_in,
    zenith,
    h_ogs,
) -> Tuple[int, int, float, int]:
    """Based on the selected values in the GUI it preapres the exposure, elevation and
    zenith attenuation we will finally use.

    1. If manual exposure mode is selected, retrieves the exposure time, ensuring that
    the value is non-negative.

    2. If individual elevation mode is selected, retrieves the elevation angle, ensuring
    that the selected value is between 0 and 90 degrees of elevation.

    3. Selects the value of the atmospheric zenith attenuation.

    Args:
        elevation_in (tk.StringVar): Container of the chosen elevation mode.
        elevation_angle_in (tk.StringVar): Container of the chosen elevation angle (if manual).
        exposure_in (tk.StringVar): Container of the chosen exposure mode.
        exposure_time_in (tk.StringVar): Container of the chosen exposure value (if manual).
        zenith (tk.StringVar): Container of the atmospheric zenith attenuation.
        h_ogs (tk.StringVar): Container of the height of the OGS used.

    Raises:
        ValueError: Exposure time value is a lower than zero.
        ValueError: Elevation angle is lower than 0 or bigger than 90.

    Returns:
        tuple[int, int, float, int]: elevation_angle (int): Final selected elevation angle (if individual).

        exposure_time_value (int): Final selected exposure value (if manual).

        zenith (float): Final selected zenith attenuation value.

        h_ogs (int): Final height of the selected OGS.
    """
    # Validate exposure time
    if exposure_in.get() == "Manual":
        try:
            exposure_time_value = int(exposure_time_in.get())
            if exposure_time_value <= 0:
                raise ValueError
        # Error validation
        except ValueError:
```

```python
            print("[ERROR] Invalid exposure time. Must be a positive integer.")
            return
    else:
        exposure_time_value = 1  # Default value for Auto mode

    # Validate elevation
    if elevation_in.get() == "Individual":
        try:
            elevation_angle = int(elevation_angle_in.get())
            print(elevation_angle)
            if elevation_angle <= 0 or elevation_angle >= 90:
                raise ValueError
        # Error validation
        except ValueError:
            print(
                "[ERROR] Invalid elevation angle. Must be a whole angle between 1 and 89
                ↪  degrees."
            )
            return
    else:
        elevation_angle = 15  # Default value for Auto mode

    if zenith.get() == "Bad 1550nm [0.891]":
        zenith = 0.891
    elif zenith.get() == "Good 1550nm [0.986]":
        zenith = 0.986
    elif zenith.get() == "Bad 850nm [0.705]":
        zenith = 0.705
    elif zenith.get() == "Good 850nm [0.950]":
        zenith = 0.950
    else:
        zenith = 0.963

    if h_ogs.get() == "IKN-OP":
        h_ogs = 650
    else:
        h_ogs = 600

    return (
        elevation_angle,
        exposure_time_value,
        zenith,
        h_ogs,
    )
```

A.2.7 link_budget.py

```python
"""This modules manages the creation and display of the link budget."""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\03_SCRIPTS\05_LINK_BUDGET\link_budget.py

from typing import Union
from scipy.special import erfinv

import numpy as np
import mplcursors
import matplotlib.pyplot as plt

```

```python
import math

from . import constants as const


def printer_lb(
    el: Union[np.ndarray, int],
    elevation_mode,
    sat,
    a_tx: int,
    p_tx: int,
    ppb: int,
    teta_tx: float,
    a_rx: int,
    leng: Union[np.ndarray, float],
    g_tx: float,
    a_fsl: Union[np.ndarray, float],
    i_axial: Union[np.ndarray, float],
    area_rx: float,
    a_atm: Union[np.ndarray, float],
    a_bw: int,
    g_rx: float,
    p_rx: Union[np.ndarray, float],
    int_ogs_lin: Union[np.ndarray, float],
    int_ogs_lin_loss: Union[np.ndarray, float],
    p_ogs_mean: Union[np.ndarray, float],
    p_ogs_mean_loss: Union[np.ndarray, float],
    p_rx_lin: Union[np.ndarray, float],
    wl: float,
    p_rfe_lin: float,
    a_sci: int,
) -> None:
    """Prints the graph or the results and summary of the link budget.

    1. If the elevation mode is "Full" it will just print the graph, if not
    it will print the result of the link budget.

    Args:
        el (np.ndarray | int): Elevation of the satellite
        elevation_mode (tk.StringVar): Container of the elevation mode.
        sat (tk.StringVar): Container of the payload used.
        a_tx (int): Transmitter optical loss.
        p_tx (int): Transmitter power.
        ppb (int): Photons/bit.
        teta_tx (float): Transmitter divergence.
        a_rx (int): Receiver optical loss onto RFE.
        leng (np.ndarray | float): Length of the link.
        g_tx (float): Transmitter antenna gain.
        a_fsl (np.ndarray | float): Free-space loss.
        i_axial (np.ndarray | float): Axial intensity at OGS-distance.
        area_rx (float): Receiver antenna area.
        a_atm (np.ndarray | float): Atmospheric attenuation loss.
        a_bw (int): Mean BeamWander loss.
        g_rx (float): Reveiver antenna gain.
        p_rx (np.ndarray | float): Received power.
        int_ogs_lin (np.ndarray | float): Intensity onto OGS-apertue exc. losses.
        int_ogs_lin_loss (np.ndarray | float): Intensity onto OGS-apertue inc. losses.
        p_ogs_mean (np.ndarray | float): Power into the OGS-apertue - no additional
        ↪  RX-losses.
        p_ogs_mean_loss (np.ndarray | float): Power into the OGS-apertue including
        ↪  RX-losses.
```

```
        p_rx_lin (np.ndarray | float): RxPower onto RFE-detector incl all losses.
        wl (float): Wavelength.
        p_rfe_lin (float): RFE-sensitivity for an specific Photons/bit.
        a_sci (int): Scintillation loss.
    """
    if elevation_mode.get() == "Full":
        print(int_ogs_lin_loss)
        el = np.arange(5, 90)
        print(el)
        plt.figure()

        plt.plot(el, int_ogs_lin_loss * 1e6, "-", color="r", linewidth=2)
        plt.ylabel("Intensity onto Camera-apertue  / µW/m²")
        plt.xlabel("Elevation / 1°")
        plt.subplots_adjust(left=0.17, right=0.95, top=0.92, bottom=0.13)
        plt.title(f"{sat.get()} intensity Link Budget")
        plt.grid(True)

        fig_manager = plt.get_current_fig_manager()

        # Dimensions and position of the link budget window (WidthxHeight+X+Y)
        fig_manager.window.wm_geometry("355x372+545+290")

        mplcursors.cursor()
        plt.show(block=False)
    else:
        print("/////////////////////////////////////////////////////")
        print()
        print(f"Transmit-power = {p_tx} dBm")
        print(f"Divergence = {((teta_tx*1E6*10)/10)} µrad")
        print(f"Optical loss Tx = {a_tx} dB")

        print(
            f"Optical loss onto RFE (incl. splitting) = {a_rx} dB - We are not "
            f"taking into account optical loss as the receiver is a camera"
        )
        # if sat == "OsirisV1":
        #     print(
        #         f"Optical loss onto RFE (incl. splitting) = {a_rx} dB - in OSIRISv1 from FLP -7,5dB were measured in 30cm telescope towards PowerSensor"
        #     )
        # print(f"Optical loss Rx (incl. splitting) = {a_rx} dB - in KIODO only 4% of Rx-light was on RFE-APD")

        print()
        print(f"Link-distance  = {(leng/100)*.1} km")
        print(f"Tx-antenna gain = + {g_tx} dB")
        print(f"Freespace Loss  = {a_fsl} dB")
        print(
            f"   # axial Intensity a OGS-distance = {i_axial*1E6} µW/m^2, after "
            f"only distance and Tx-internal losses"
        )
        print(f"   # Area of Rx-antenna = {area_rx} m^2")
        print(
            f"   # power into Rx-aperture [no a_atmo nor a_pointing, only a_Tx, "
            f"a_fsl, g_Rx]  = {1E6*i_axial * area_rx} µW"
        )
        print()
        print(f"atmosph. atten. = {a_atm} dB")
        print(
```

```python
            f"mean BeamWander loss = {a_bw} dB - Being the receiver a camera, we are "
            f"not taking into acount BeamWander losses"
        )

        print(
            f"scinti-loss     = {a_sci} dB - Once again we suppose Scintillation loss as cero"
        )

        print()
        print(f"Rx-antenna gain = + {g_rx} dB")
        print(f"optical loss Rx = {a_rx} dB, includes splitting for Tracking")
        print(f"RxPower on RFE with all losses  = {p_rx} dBm")
        print(
            f"   # intensity onto OGS-apertue incl atmosphere but excl. "
            f"Rx-losses = {int_ogs_lin *1E6} µW/m^2"
        )
        print(
            f"   # intensity onto OGS-apertue incl atmosphere including "
            f"Rx-losses = {int_ogs_lin_loss *1E6} µW/m^2"
        )
        print(
            f"   # power into the OGS-apertue - no additional RX-losses"
            f"= {(10**(p_ogs_mean/10)/1000)*1E6} µW"
        )
        print(
            f"   # power into the OGS-apertue including RX-losses = "
            f"{(10**(p_ogs_mean_loss/10)/1000)*1E6} µW"
        )
        print(
            f"RxPower onto RFE-detector incl all losses =  {p_rx_lin*1E9} nW, "
            f"sufficient for {(p_rx_lin/ppb/(const.H*const.C/wl))/1E9} Gbps at {ppb} Photons/bit"
        )
        print(
            f"RFE-sensitivity for {ppb} Ppb   =  {p_rfe_lin*1E9} nW or "
            f"{math.log10(p_rfe_lin*1000)*10} dBm"
        )
        print()
        print(f"Link Margin: {p_rx - math.log10(p_rfe_lin*1000)*10 } dBm")
        print()

        print("/////////////////// Summary ////////////////////")
        if elevation_mode.get() == "Individual":
            print(f"mean source power  +{p_tx} dBm")
            print(f"Tx-internal losses {a_tx} dBm")
            print(f"Tx-antenna gain    +{g_tx} dB")
            print(f"pointing loss      {a_bw} dB")
            print(f"Distance           {leng/1000} km")
            print(f"freespace loss     {a_fsl} dB")
            print(f"Atmospheric loss   {a_atm} dB")
            print(f"scintillation loss {a_sci} dB")
            print(f"Rx-antenna gain    +{g_rx} dB")
            print(f"Power into Rx-Aper {p_ogs_mean} dBm")
            print(f"Rx-internal losses {a_rx} dB")
            p_rx_ = (
                p_tx + a_tx + g_tx + a_bw + a_fsl + a_atm + a_sci + g_rx + a_rx
            )
            print(f"Power onto detectr {p_rx_} dBm")
            print(
```

```python
                f"Sensitivity of RFE {10*math.log10(p_rfe_lin*1000)} dBm  / {p_rfe_lin*1E9} nW"
            )
            print(
                f"Link Margin        {p_rx_ - 10*math.log10(p_rfe_lin*1000)} dB"
            )
            print()

            print("/////////////////// Intensity ///////////////////")
            print(
                f"Axial Intensity at OGS-distance = {i_axial*1E6} µW/m^2, after only "
                f"distance and Tx-internal losses"
            )
            print(
                f"Intensity onto OGS-apertue incl atmosphere but excl. Rx-losses = "
                f"{int_ogs_lin *1E6} µW/m^2"
            )
            print(
                f"Intensity onto OGS-apertue incl atmosphere including Rx-losses = "
                f"{int_ogs_lin_loss *1E6} µW/m^2"
            )
            print()

            print("///////////////////// Power /////////////////////")
            print(
                f"Power into Rx-aperture [no a_atmo nor a_pointing, only a_Tx, a_fsl, "
                f"g_Rx]  = {1E6*i_axial * area_rx} µW"
            )
            print(
                f"Power into the OGS-apertue - no additional RX-losses = "
                f"{(10**(p_ogs_mean/10)/1000)*1E6} µW"
            )
            print(
                f"Power into the OGS-apertue including RX-losses = "
                f"{(10**(p_ogs_mean_loss/10)/1000)*1E6} µW"
            )


def link_budget(elevation_mode, el: int, payload, zenith: float, h_ogs: int) -> None:
    """Performs the link budget based on the parameters from the GUI:

    1. Selects specific parameters based on the payload chosen.

    2. Specifies the losses.

    3. Calculates antennas dimensions.

    4. If the elevation mode is "Full", the elevation angle will be converted to an array from 0 to 90 degrees.

    5. The rest of the parameters dependent on the elevation are calculated based on if
    the elevation is just a point or the full range.

    6. Computes the final received power at the receiver.

    7. Calculates the intensity onto the receiver using its area.

    Args:
        elevation_mode (tk.StringVar): Container of the elevation mode.
        el (int): Elevation angle.
```

```
        payload (tk.StringVar): Container of the payload used.
        zenith (float): Final selected zenith attenuation value.
        h_ogs (int): Final height of the selected OGS.
    """
    if payload.get() == "OsirisV1":
        h_orbit = 595  # km - Satellite height
        wl = 1545e-9  # m - Wavelenght of the downlink
        p_tx = 30  # dBm - Transmited power
        teta_tx = 1e-3  # rad - OSIRISv1: 1.0E-3 oder 1.2E-3 mrad - die Dokumente sagen immer 1,2 aber CF meinte f rs CNES-Paper 1,0
        a_tx = -1  # dB - Optical Transmissor losses (Tx)
        dr = 39e6  # bps - datarate in KIODO, 39Mbps in OSIRIS-FLP for OCAM and some tests to OP
        ppb = 250  # ppb - Photons per bit required bei RFE at BER=1E-3 at first OSIRIS with APD-RFE-100-OLD, 320Photons for RFE-300-NEW
    # diese Formel muss noch durch WL erg nzt werden:   D_tx=0.2; teta_tx = 100E-6 * 0.01/D_tx; % estimate for near-optimum cut-gauss Tx
    elif payload.get() == "KIODO":
        h_orbit = 610  # km - Satellite height
        wl = 847e-9  # m - Wavelenght of the downlink
        p_tx = 20  # dBm - Transmited power
        # p_tx = 16.99
        teta_tx = 5.5e-6  # rad - OICETS-Kirari-LUCE FWHM beam divergence
        a_tx = -1  # dB - Optical Transmissor losses (Tx)
        dr = 39e6  # bps - datarate in KIODO, 39Mbps in OSIRIS-FLP for OCAM and some tests to OP
        ppb = 250  # ppb - Photons per bit required bei RFE at BER=1E-3 at first OSIRIS with APD-RFE-100-OLD, 320Photons for RFE-300-NEW
    elif payload.get() == "CubeCat":
        h_orbit = 455  # km - Satellite height
        wl = 1545e-9  # m - Wavelenght of the downlink
        p_tx = 24.7712  # dBm - Transmited power [300 mW]
        # p_tx = 16.99
        teta_tx = 104E-6  # rad - collimator F220FC-1550
        a_tx = 0  # dB - Optical Transmissor losses (Tx)
        dr = 39e6  # bps - datarate in KIODO, 39Mbps in OSIRIS-FLP for OCAM and some tests to OP
        ppb = 250  # ppb - Photons per bit required bei RFE at BER=1E-3 at first OSIRIS with APD-RFE-100-OLD, 320Photons for RFE-300-NEW
        # diese Formel muss noch durch WL ergnzt werden:   D_tx=0.2; teta_tx = 100E-6 * 0.01/D_tx; % estimate for near-optimum cut-gauss Tx
    else:
        h_orbit = 595  # km - Satellite height
        wl = 1545e-9  # m - Wavelenght of the downlink
        p_tx = 30  # dBm - Transmited power
        teta_tx = 1e-3  # mrad - OSIRISv1: 1.0E-3 oder 1.2E-3 mrad - die Dokumente sagen immer 1,2 aber CF meinte frs CNES-Paper 1,0
        a_tx = -1  # dB - Optical Transmissor losses (Tx)
        dr = 39e6  # bps - datarate in KIODO, 39Mbps in OSIRIS-FLP for OCAM and some tests to OP
        ppb = 250  # ppb - Photons per bit required bei RFE at BER=1E-3 at first OSIRIS with APD-RFE-100-OLD, 320Photons for RFE-300-NEW
        # diese Formel muss noch durch WL ergnzt werden:   D_tx=0.2; teta_tx = 100E-6 * 0.01/D_tx; % estimate for near-optimum cut-gauss Tx

    h_orbit_m = h_orbit * (10**3)  #  m - Satellite height
    # 1W mean was used in FLP-OSIRISv1 experiments with OCAM
    # 100mW or 50mW mean we expect in KIODO, since the 20dBm mentioned in the book might be peak-power

```

```python
sigma_jit = 0.85 * teta_tx / 2  # erzeugt dann -3dB BW-loss
# sigma_jit = 1E-9 # 0.85*teta_tx/2 # erzeugt dann -3dB BW-loss
# sigma_jit = teta_tx/4 # irrelevant when we set beta later below

beta = teta_tx**2 / sigma_jit**2 / (8 * math.log(2))
# beta = 8
# beta = 2 # produces a pointing loss of -1.7dB
# beta=1000 # no pointing loss for the plot in fig.10

############  RECEIVER SETTINGS  ############   -> Check standalone script for noncam values
# LOSSES
a_rx = 0  # dB - Optical Receiver losses (Tx)  -
a_bw = 0  # dB - Beam Wander losses
a_sci = 0  # dB - Scintillation losses

###########  ANTENA'S DIMENTIONS  ###########
d_rx_o = 2.5e-3  # m - Diameter Rx-apertur in meter - Infra-FE4.41.0-17
# d_rx_o = 5.6e-3  # m - Diameter Rx-apertur in meter - Infra-FE5.61.0-
area_rx = math.pi * (d_rx_o / 2) ** 2
# m^2 - Area of a fisheye lens based on its aperture

# alpha = (180/math.pi) * math.asin( (const.R_E+h_ogs)/(const.R_E+h_orbit) * math.sin((90+el)*math.pi/180) )
# gamma = 90 - el - alpha;
# leng = math.sqrt( (const.R_E+h_ogs)**2 + (const.R_E+h_orbit_m)**2 - 2 * (const.R_E+h_ogs)*(const.R_E+h_orbit_m)*math.cos(gamma*math.pi/180) )
if elevation_mode.get() == "Individual":
    a = math
    grad = el * math.pi / 180
else:
    el = np.arange(5, 90)
    a = np
    grad = np.radians(el)

leng = a.sqrt(
    (const.R_E + h_ogs) ** 2 * a.sin(grad) ** 2
    + 2 * (h_orbit_m - h_ogs) * (const.R_E + h_ogs)
    + (h_orbit_m - h_ogs) ** 2
) - (const.R_E + h_ogs) * a.sin(grad)

a_fsl = 10 * a.log10((wl / (4 * math.pi * leng)) ** 2)
# dB - Freespace losses

a_atm = 10 * a.log10(zenith ** (1.0 / a.sin(math.pi * el / 180)))
# dB - Athmosperic Attenuation
# It is using degrees now

g_tx = 10 * math.log10((3.33 / teta_tx) ** 2)
# dB - Gain of thetransmissor antena

# weitere Berechnung - sollte selbes rauskommen:  i_axial = (0.693/pi) * 10**( (a_tx+p_tx) /10)/1000  / (leng*teta_tx/2)**2 # W/m^2
i_axial = 0.001 * 10 ** (
    (
        p_tx
        + a_tx
        + g_tx
        + a_fsl
        + 10 * math.log10(4 * math.pi * 1 / (wl**2))
```

```
        )
        / 10
    )  # W/m^2 - Axial Intensity

    g_rx = 10 * math.log10(4 * math.pi * area_rx / (wl**2))
    # dB - Gain of the receiver antena
    # g_rx = 0

    p_rx = (
        p_tx + a_tx + g_tx + a_fsl + a_bw + a_atm + a_sci + g_rx + a_rx
    )  # dBm - RxPower on RFE with all losses

    p_ogs_mean = p_rx - a_rx
    # dBm - power onto OGS-aperture - no Rx-internal losses
    p_ogs_mean_loss = p_rx
    # dBm - power onto OGS-aperture - WITH Rx-internal losses
    int_ogs_lin = (10 ** ((p_ogs_mean) / 10) / 1000) / area_rx
    int_ogs_lin_loss = (10 ** ((p_ogs_mean_loss) / 10) / 1000) / area_rx
    p_rx_lin = (10 ** (p_rx / 10)) / 1000  # W
    p_rfe_lin = ppb * const.H * const.C * dr / wl  # W

    printer_lb(
        el,
        elevation_mode,
        payload,
        a_tx,
        p_tx,
        ppb,
        teta_tx,
        a_rx,
        leng,
        g_tx,
        a_fsl,
        i_axial,
        area_rx,
        a_atm,
        a_bw,
        g_rx,
        p_rx,
        int_ogs_lin,
        int_ogs_lin_loss,
        p_ogs_mean,
        p_ogs_mean_loss,
        p_rx_lin,
        wl,
        p_rfe_lin,
        a_sci,
    )
```

A.2.8 printer.py

```
"""This script manages all the printing functions needed to display the results
↪  correctly."""

# C:\Users\alda_ik\Documents\04_PROGRAMMING\02_FINAL_PROJECT\constants.py


def print_preamble() -> None:
```

```python
    """Printing fuction - prints program preamble."""
    print("///////////////////////////////////////////////////")
    print("/// IR Camera System for Satellite Observation ///")
    print("///////////////////////////////////////////////////\n")


def print_preamble_settings() -> None:
    """Printing fuction - prints settings preamble."""
    print("//////////////////// Settings ////////////////////")


def print_start_stream() -> None:
    """Printing fuction - prints the start of the stream."""
    print()
    print("/// Stream started. Press <q> to stop stream ///")


def print_end_stream() -> None:
    """Printing fuction - prints the end of the stream."""
    print("/////////////////// Stream ended ///////////////////")
    print("///////////////////////////////////////////////////\n")


def print_usage() -> None:
    """Printing fuction - prints the usage."""
    print("Usage:")
    print("    python asynchronous_grab_opencv.py [camera_id]")
    print("    python asynchronous_grab_opencv.py [/h] [-h]")
    print()
    print("Parameters:")
    print(
        "    camera_id   ID of the camera to use (using first camera if not specified)"
    )
    print()
```