Chair of Astronautics
Prof. Prof. h.c. Dr. Dr. h.c.
Ulrich Walter

Master's Thesis

Experimental Analysis of Pose Tracking Performance of Visual and Infrared Stereo Cameras under Low Orbital Light Conditions

LRT-MA-2019/18

Author:
Lucio Franceschini

Supervisor:	Dr.-Ing. Martin Dziura Chair of Astronautics Technische Universität München
Examiner:	Prof. Prof. h.c. Dr. Dr. h.c. Ulrich Walter Chair of Astronautics Technische Universität München
Date of Submission:	2020-03-19

Acknowledgements

I want to thank, first of all, my supervisor Martin Dziura, for offering me the opportunity to develop this thesis even as an Erasmus student while also providing me with all the reasonable help, giving feedback whenever possible, and always being available for any problem. I additionally desire to thank my parents, that always supported me in my decision to develop my thesis as an Exchange student in Europe and through my studies at home.

Abstract

In this work, the stereo camera D435 and the solid-state LIDAR HPS-3D160, together with the previously present ZED camera, have been integrated into the RACOON-Lab laboratory at TUM. The resulting hardware structure enables a swift appending of future new sensors to the platform, while the new storage format enables storing all the results in an open and extensible format for current and future usage. The software framework also requires minimum changes for a new addition to the system, while also enabling the development of new algorithms and filters without the need of a full rewrite.

The completed setup has then been tested under orbital light conditions, where the laboratory simulated a short-range fly-around of a target satellite under the illumination of an Earth's albedo simulation and without the presence of external lights while changing the inclination of the target between the various measurements procedures. As external illumination factors do not influence the LIDAR, its HDR integration time has been changed between the two situations.

The resulting depth information recorded, as it was or after being filtered of the background, has been feed to the path tracking DIFODO algorithm, where the paths have been extracted. The resulting trajectories have been examined by applying a previously developed metric, and the computed parameters have been analyzed.

The resulting data show few dependencies from the position of the target and the quality of the metric parameters. For the selected settings of the tracking algorithm and the filter, elevated drift values have been found for the D435 and the HPS-3D160, while the ZED presents, overall, better results.

Contents

List of Figures

List of Tables

Symbols and Formulas

Symbol	Unit	Description	Symbol	Unit	Description
C_{F}	m	Difference between C_{K} and C_{G}	r_{F}	m	Difference between r_{K} and r_{G}
ε	deg	Elevation of the best-fit plane	R	m	Residuum
δ	deg	Drift	δ_{max}	deg	Maximum drift
C_{G}	m	Reference position of the circle	C_{K}	m	Center of the best-fit circle
r_{G}	m	Radius of rotation	r_{K}	m	Radius of the fitting circle
n_{G}	1	Versor of the reference rotation	n_{K}	1	Normal versor of the best-fit plane

Abbreviations

ADR	Active Debris Removal
COTS	Commercial Off-The-Shelf
CUDA	Compute Unified Device Architecture
D	Diagonal
DIFODO	DIFferential ODOmetry
ESA	European Space Agency
FoV	Field of View
GEO	Geostationary Earth Orbit
GPU	Graphics Processing Unit
GUI	Graphical User Interface
H	Horizontal
H. A.	High Accuracy
H. D.	High Density
HDF5	Hierarchical Data Format version 5
HDR	High Dynamic Range
HMI	Human Machine Interaction
ID	Identifier
IDE	Integrated Development Environment
I/O	Input/Output
IR	Infrared
LEO	Low Earth Orbit
LIDAR	LIght Detection And Ranging
MLI	Multi-Layer Insulation
MRPT	Mobile Robot Programming Toolkit
N/A	Not Available
NetCDF4	Network Common Data Form version 4
OOS	On-Orbit Servicing
PD	Payload Debris
PETG	Polyethylene Terephthalate Glycol-modified
PF	Payload Fragmentation Debris
PL	Payload
PM	Payload Mission Related Object
RACOON-Lab	Robotic Actuation and On-Orbit Navigation Laboratory

RB	Rocket Body
RD	Rocket Debris
RF	Rocket Fragmentation Debris
RM	Rocket Mission Related Object
ROI	Region of Interest
SDK	Software Development Kit
SLAM	Simultaneous Localization And Mapping
ToF	Time of Flight
UCAR	University Corporation for Atmospheric Research
UI	Unidentified
UUID	Universally Unique IDentifier
V	Vertical
VCSEL	Vertical Cavity Surface Emitting Laser
YAML	YAML Ain't Markup Language

1 Introduction

Since the first days of human activity in space, the presence of artificial objects orbiting the Earth has increased on a more than constant rate [1], as visible in fig. 1–1. The vast majority of these persist well beyond their expected useful life, increasing the number of so-called space debris, which pose a severe threat to present and future space activities near Earth, such as the risk of collision of larger objects between themselves and with active satellites. This cascade of collisions can trigger an out-of-control growth in the total number of objects in orbit even if further launches are completely halted, making it impossible to use some orbits due to the high risk of collision during the lifetime of the mission. This phenomenon is known as Kessler's syndrome [12] and, given the latest simulations [13] performed and the numerous collision and fragmentation events already recorded [1], seems to have already started. To interrupt this runaway reaction, it is therefore essential to remove the larger objects that cannot be autonomously deorbited and passivated, as they will be the primary source of future fragments.

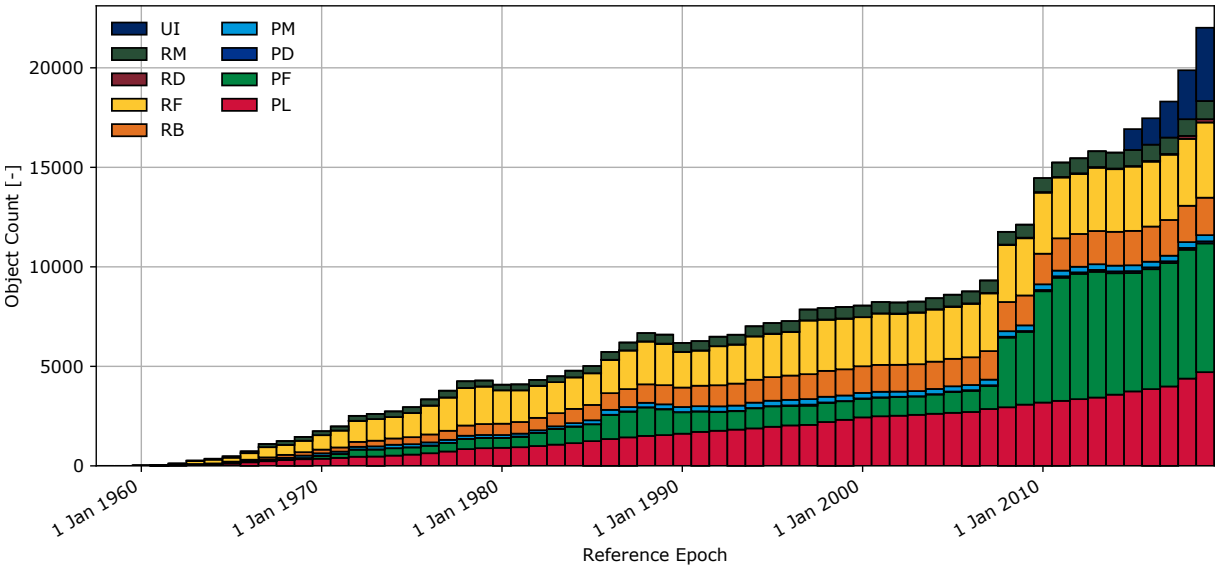


Fig. 1–1: Evolution of number of objects in all orbits [1]. All abbreviations are explained in the list of Abbreviations

Among the many challenges that these objects present in an Active Debris Removal (ADR) scenario, the lack of an autonomous attitude control system and a communication channel makes the docking procedure extremely complicated, as the chaser cannot obtain the target attitude directly from the target, which could also be tumbling.

In this context, the RACOON-Lab [4], a robotic platform developed to simulate close-range operations and On-Orbit Servicing (OOS) missions under the Chair of Astronautics at the Department of Aerospace and Geodesy under the Technical University of Munich, can be used to test and simulate the various aspects of the final approach in an ADR space missions.

Following previous theses developed within the RACOON-Lab group, the main objective of this work is to integrate Intel's RealSense D435 infrared stereo camera and the Hypersen Solid-State LIght Detection And Ranging (LIDAR) HPS-3D160 into the laboratory, to subsequently collect data for future use and then test its performance. To this end, the camera needs to detect, in a realistic 3D scenario where the camera is mounted on a tracker system, its position with respect to a target satellite full of glare and reflections generated from the simulated Sun available from the laboratory. The depth map so measured is then fed to an implementation of the DIFODO algorithm [14], which allows following the camera's position with respect to the target satellite. The resulting path is then compared with that obtained from the one calculated by the Stereolabs ZED stereo camera (following the same procedure) tested in the previous work conducted by Niklas Hab [15]. To be able to evaluate in a qualitative way the results, a metrics previously developed by Flavio Rehn [16] is used.

1.1 State of the art

As already mentioned, the growing number of space debris in Earth's orbit [1] has dramatically increased interest in being able to perform ADR operations and, more generally, OOS missions. An example of this attention can be seen in OHB's initially proposed e.Deorbit mission [17] for the removal of the Envisat satellite of the European Space Agency (ESA). Such a scenario, where a chaser needs to approach a target in orbit, requires a high degree of accuracy and robustness. Compared to docking and berthing procedures between active satellites, a space debris does not offer accurate knowledge about its attitude. Thus the chaser requires sensors to be able to detect the information mentioned above. It is also worth noting that these operations cannot be controlled via ground-based commands with ease, as this manual approach suffers from communication delays, low availability coverage [18], as it is also visible in complicated and time-sensitive e.Deobit angular-momentum synchronization procedure [19].

To fulfill such a task is available an extensive series of devices, as noted in fig. 1–2. Among these sensors, for OOS missions, it is typically considered the use of monocular or stereo camera or LIDAR systems, all of which are widely available as Commercial Off-The-Shelf (COTS) thanks to their extensive array of possible applications in many industries, as well as advances in machine learning capabilities able to improve Simultaneous Localization And Mapping (SLAM) usage [20].

The use of a monocular or stereo camera offers a low-cost solution with a high density of depth information, but the quality of the results depends heavily on image quality and illumination factor [21] because it is a passive sensor. On the other end, a LIDAR system is an active device that projects laser rays to the objects' surfaces and is then able to reconstruct their geometric shape by analyzing the light response. These sensors are more robust to the interference of the environment as they do not suffer from lack of external illumination (if it does not interfere with the laser) and offer a higher precision compared to the stereo camera. LIDAR systems are usually expensive and present, in their traditional form, a much more complex structure than a camera, as the laser beam needs to be physically moved via a mechanical mechanism, and its movement is a bottleneck to the detection speed. New developments in LIDAR technology [22],

prompted by increased interest in autonomous vehicles and aerial mapping, enabled the development of solid-state LIDAR, with no moving parts and much lower prices, as well as increased velocity scanning. More recent developments prompted by Microsoft Kinect's success combined active sensors and with passive optical sensors in the form of RGB-D cameras [23], merging the advantages of less dependency from external lights while also offering a more simple structure with no moving parts, low costs and a high density of information.

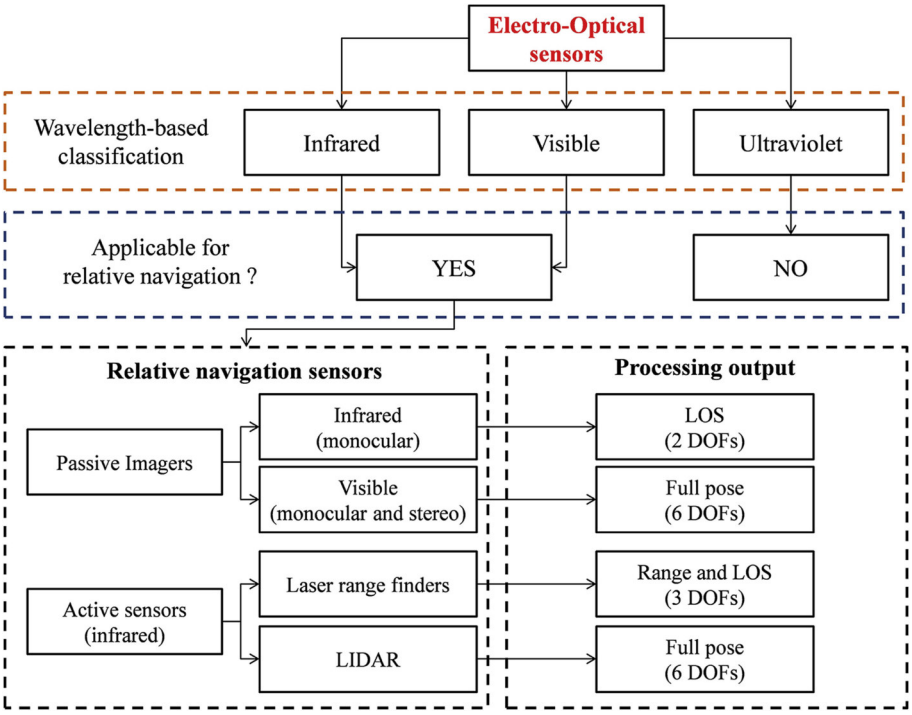


Fig. 1–2: Taxonomy of EO sensors for spacecraft applications [2]

Many publications have investigated the application application of these sensors to orbital operations, as visible, for example, in [18], [24], [25], [26] and [27]. These work, however, did not surpass a proof-of-concept stage, as they simulated the targeted spacecraft via software rendering or adopted a simple setup for their analysis and thus were only able to identified some mayor challenges [28], but not to characterize them in full detail. Among these, the presence of difficult illumination conditions that can be found in orbit or the widespread presence of highly reflecting surfaces on the target spacecraft could invalidate the values of the resulting depth map [3]. This degradation can significantly affect the performances of any tracking algorithm developed, as well as the quality of the 3D model obtainable via the SLAM. This research has proved that results are highly influenced by the light sources' position and the inclination of the spacecraft (relative to the observer device).

1.2 Objectives

The main objectives of this thesis are, first and foremost, the following:

1. Integration of the D435 infrared stereo camera and the HPS-3D160 LIDAR in the

RACOON-Lab, concerning the hardware side (mechanical supports and their incorporation in the existing platform) but also the software side, to allow recording operations and subsequent data analysis;

2. Test the results of previous work while the lab is simulating a realistic scenario in Low Earth Orbit (LEO) or Geostationary Earth Orbit (GEO), with a batch of tests under the Earth's albedo simulator, and another with a complete lack of external light. The inclination of the satellite needs to be changed to investigate its effects in the depth mapping algorithms;
3. Use the depth maps gathered with these measurements to an existing implementation of the DIFODO algorithm and apply an already developed metric to the resulting path, to then analyze the results.

After these primary objectives, several secondary objectives were planned during the development of this thesis, to integrate the sensors while taking into account not only the short-term needs of this work but also the long-term nature of the RACOON-Lab. This means:

- Provide a versatile support system that can accommodate future devices without requiring a substantial redesign of the platform;
- Development of a software framework to allow an easier integration of multiple devices, algorithms, and filters;
- Store the measurement campaign results in an open and extensible format for future extension and use;
- Convert of metrics from the MATLAB programming language to Python, for easier integration into future software frameworks.

1.3 Methodology

To successfully achieve the objectives of this thesis was necessary to record a series of measurements with the RACOON-Lab, and the procedure to be followed was based mainly on the one established in successful previous campaigns. The experiment consisted of recording the in-orbit simulated scenario as the satellite (inclined by a predetermined angle) is rotating around the laboratory's vertical axis. This procedure that is resembling a fly-around of a chaser satellited around a target has been repeated multiple times, continually changing the inclination of the satellite from $0°$ to $90°$ while the laboratory was kept under identical conditions. As the light of the room was changed two times, the entire procedure was repeated for each situation. For the first experience, the laboratory was left in the dark (to analyze the response of the devices under their active illumination system), while in the second case, the satellite was illuminated by the Earth's albedo simulator.

Additionally, for each situation, a set of reference recordings was taken where nothing was changed in the laboratory. This data was to be used to detect the whole toolchain measurement uncertainty.

All the recorded information was to be converted to an open data format, where tracking algorithms were able to read it and compute the fly around paths to be analyzed then under the previously developed metric. The resulting parameters obtained were then compared with each other to find possible trends and influences by the orbital conditions.

Before recording the data, a series of pre-tests had to be made, to be able to choose the correct parameters for the recording devices to be used during the measurement campaign. The settings to analyze were chosen based on the manufacturers' recommendations.

Several software applications, necessary to manage the massive flux of data, have been developed. These programs have been written in C++17, employing Microsoft's Integrated Development Environment (IDE) Visual Studio 2019, and Python 3.8, via Microsoft's Visual Studio Code. The analysis of the results (the pre-tests and the experiments) employed the Jupyter software framework, which was also running in the Visual Studio Code environment. Additionally, to visualize the internals of the developed data structure, the viewer HDF View, to be found in the HDF5 Software Development Kit (SDK) tools, has been adopted, while the sources were already accessible via their native SDK tools.

The design of the hardware support also required the aid of computer applications, and to this end, Dassault Systèmes' SolidWorks 2019 Student Edition has been used.

The entire software toolchain, as well as the used applications, were operated under Microsoft Windows 10.

1.4 Delimitations and Limitations

The computation of the experiments has not been done in realtime, as the data were first recorded and subsequentially consumed by the tracking algorithm. While this may seem like a limitation, as the final orbital application could require a realtime usage of the data [19], the computational time of the path took a similar time of the duration of the source video itself, and the DIFODO algorithm, as described by its creator [29], is perfectly able to estimate the path quickly. Because this computation is the most time-consuming portion of the entire toolchain, with the Input/Output (I/O) performances in the second rank, no significant problem appears to concern an adequate realtime implementation of the complete procedure.

The filter applied to remove the wall of the laboratory is very simplistic and does not consider its changing nature structure, nor the effects of the reflections. While this is true, it has already been proven [15] to be enough for a short analysis of the results, while for a more extensive examination of the data, it is clear that, given more time, more work is needed in this aspect. For the same reason, the parameters of the filters applied to each device, as well as settings of the path tracking algorithm, were kept as close as possible to each other, enabling an easier comparison.

2 Introduction of used Algorithms

2.1 Path Tracking

The DIFferential ODOmetry (DIFODO) algorithm [29] is a dense method able to compute a path via visual odometry from a 3D depth map, along with the linear and angular velocity of its point in a rigid environment. It is an algorithm that uses the range flow constraint equation and has no functionality to minimize the drift of position estimation over time. The algorithm is reported to being able to work in real-time at 60 Hz on a single CPU core, and it is available in an open-source implementation in the Mobile Robot Programming Toolkit (MRPT) library. Further information on its internal working can be found in its paper, and its internal mechanisms are not necessary for the understanding of this thesis, except for the following parameters:

- The framerate of the recorded source;
- The number of levels of the coarse-to-fine pyramid scheme;
- The pixel resolution of the finest level of the pyramid, calculated by dividing the input resolution by the number of levels of the coarse-to-fine pyramid scheme.

The ZED SDK has an integrated functionality that generates a path from a ZED recorded video. As it is a proprietary algorithm, not much is known, but, since it is generated by the camera without significant effort, it is worth comparing it with the results obtained by the DIFODO algorithm, since it offers a useful reference for comparison (when available). Its computations are not done directly on the camera, but it requires a computer equipped with a CUDA-enabled computer with an NVIDIA Graphics Processing Unit (GPU). The path tracking algorithm available under the Zed SDK 3.0.2 (from here on called ZedSDK), the latest available at the time of the experiment, has been used.

2.2 BoxFilter

The wall of the laboratory and the support structure of the target, despite their black envelopment, to minimize their reflectivity, are still on the resulting depth maps. The resulting points cloud, to minimize their influence on the DIFODO path tracking algorithm, is therefore filtered via a simple geometric filter, where each point must be within a selected area of the frame and must have a valid depth range. The resulting volume has the shape of a box, as visible in fig. 2–1, hence the name.

2.3 Metric

A metric is needed to compare the effect of the various parameters on the computed paths. To this end, the previously developed [16] best-fit circle metric has been reused for the comparison, albeit with a new software reimplementation. As it is the result of a previous thesis, it will not here receive a full mathematical description, but a summary of its parameter (to better understand the results) will be given.

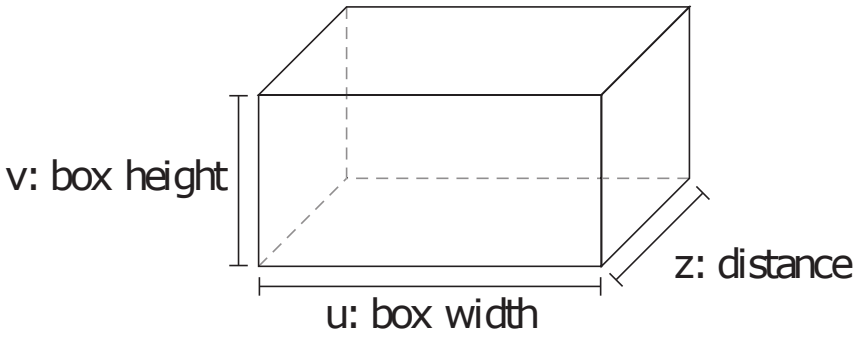


Fig. 2–1: Graphical representation of the BoxFilter [3]

As it is not possible to obtain a ground-truth reference from the laboratory due to its simulated nature, a circle that fits the results (from here on named best-fit-circle) has been calculated to reproduce it. This circle (denoted by the subscript K) and the plane that contains it are then matched, in $\mathbb{R}^3$, with the original points of the path computed via the algorithm, to extract a set of resulting parameters to compare the various configurations quantitatively. These are:

- C_F: Distance between the center point of the circle C_K and the real center of rotation C_G;
- r_F: Difference between the radius of the fitting circle r_K and the real radius of rotation r_G, which is also the distance of the camera to the center of rotation C_G;
- ε: Elevation between the circle plane and the reference plane, obtained by their normal vectors n_F and n_G;
- R: Mean residuum of the estimated path from the circle;
- δ: Drift of the estimated path from the expected position on the circle.

A graphical representation of these parameters is visible on fig. 2–2.

The references C_G, r_G, and n_G were not available due to the unavailability of the OptiTrack system of the laboratory, but, as the configuration did not significantly change from the previous thesis [15], the same values have been kept or adapted. As such, the distance r_K has been updated from $4.0\,\mathrm{m}$ to $4.1\,\mathrm{m}$, while the direction of the versor n_F has been kept normal to the xz plane since in each recording session the satellite was rotating around said axis. Finally, due to lack of data and following the previous metric implementation, the center of the reference C_G has been kept to the origin of the system.

Given the nature of these parameters, the ideal condition where the calculated path follows perfectly the theoretical one would show all values to zero.

It is also interesting to note that, while the first three parameters can only describe the path as a whole, the residuum and the drift can also illustrate each point of the calculated path, revealing how its accuracy change as time increase. These are also not related to parameters known previously, thus making them independent of related measurement errors.

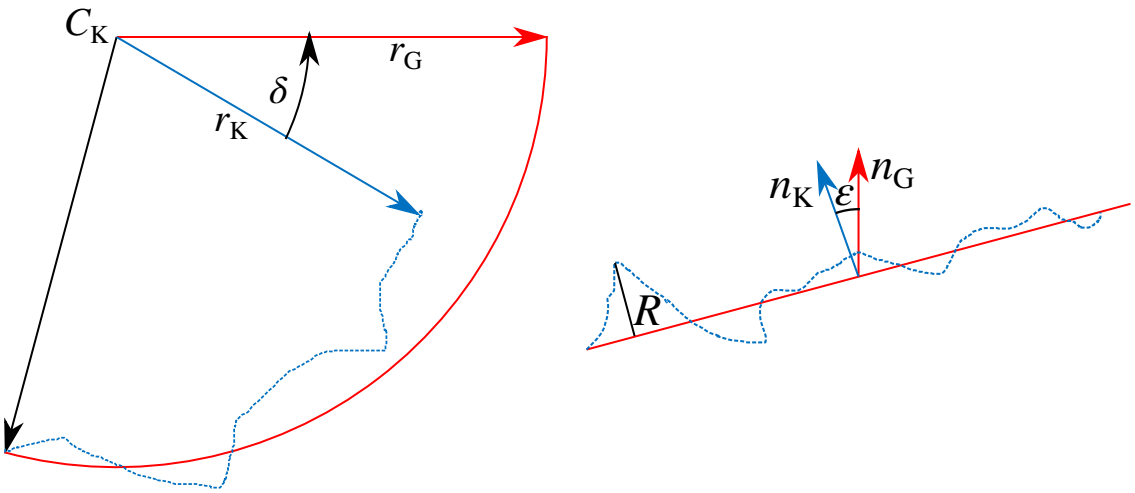


Fig. 2–2: Normal (left) and parallel views (right), relative to the best-fit plane, of the metric parameters. The blue dotted line represents the computed path, while the red line denotes the best-fit circle. The references are colored in red, while the measured parameters are visible in blue

As the metric has only been rewritten in Python without changes in its logic, due to lack of time and because it was not one the main objectives of this work, it suffers from the same problems of the previous implementation. Regarding this investigation, the computed position lacks behind the expected one by the metric, as each recording does not swipe a $360°$ angle but only a portion of it.

3 Experimental Environment in the RACOON-Lab

It is useful to describe its starting configuration, to illustrate adequately the changes made to the RACOON-Lab (visible in fig. 3–1), with a particular focus on the areas where changes have occurred.

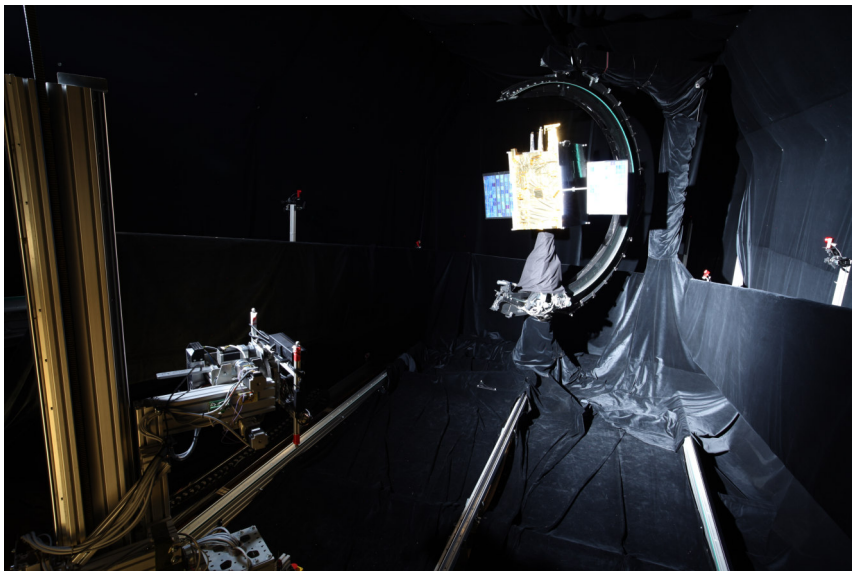


Fig. 3–1: View of target satellite and the chaser of the RACOON-Lab

3.1 General overview

The RACOON-Lab [4], is a robotic platform developed to simulate close-range operations and OOS missions. It allows, as visible in fig. 3–2, for end-to-end development and testing of all their aspects. The various scenarios make it possible to simulate the control segment, the communication between the Groundstation and the interaction with the chaser. It also emulates the scenario to be encounter in orbit, including the effects of Sunlight and Earth's albedo. The simulation can operate autonomous systems, but it is also able to include the human element and the relative Human Machine Interaction (HMI) required by the operator.

3.2 Chaser setup

The starting condition of the RACOON-Lab is visible on the fig. 3–3. On the lower bar, the Stereolabs ZED was attached, on the top of the structure was located the Microsoft Kinect, and between them, it was possible to find the LIDAR Hokuyo UST-2. They were held together with the arm of the laboratory by a series of plates screwed with it. These sensors (together with the new ones to be integrated) are later described in section 3.6.

The computer to be used for the acquisition, a ZOTAC MAGNUS EN970 (ZBOX-EN970), was located on the far right of the main structure (relative to the observer), and it was kept in place by cable ties and adhesive tape.

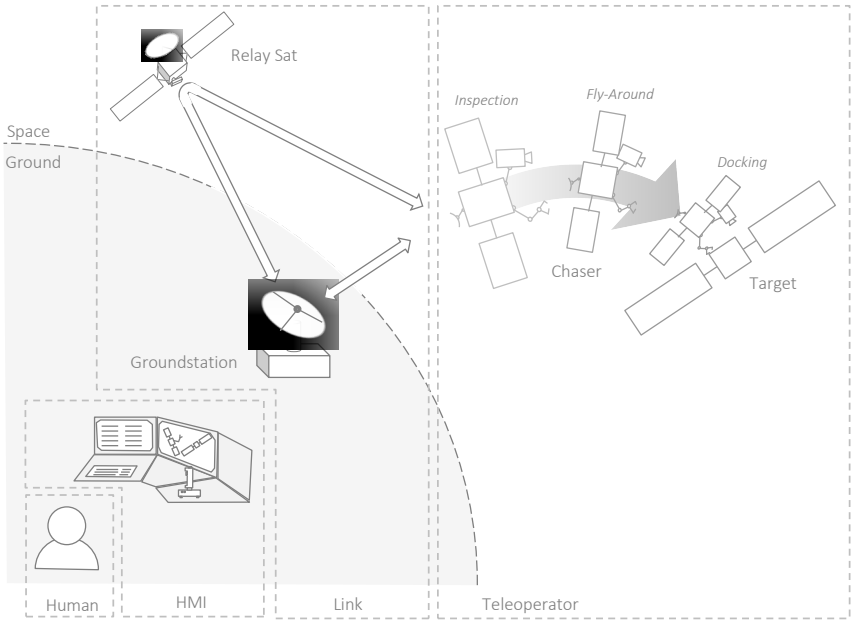


Fig. 3–2: Graphical representation of the RACOON-Lab capabilities [4]

A series of 3D printed supports (screwed in the structure) was holding all the power and data cable for all the previously mentioned devices. There were also support clamps to sustain the numerous adapters required for the Kinect, as visible on the vertical beam on fig. 3–3.

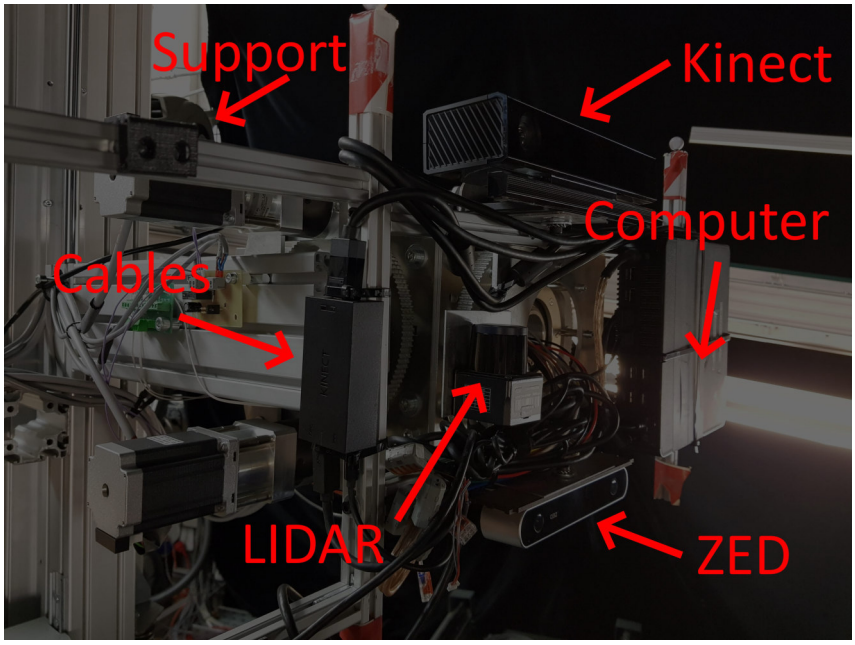


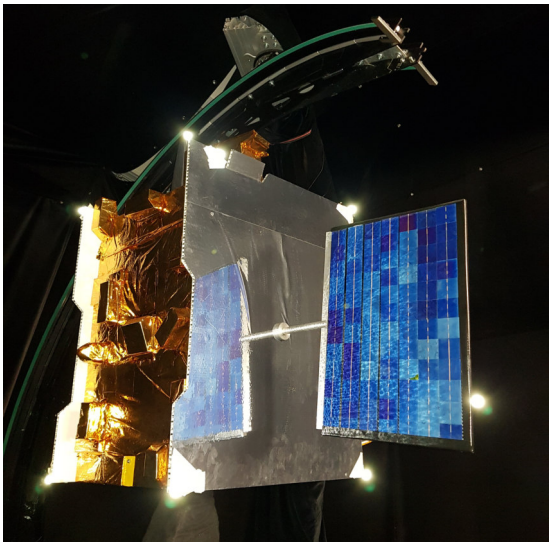
Fig. 3–3: Former configuration of the RACOON-Lab chaser

3.3 Target and chaser

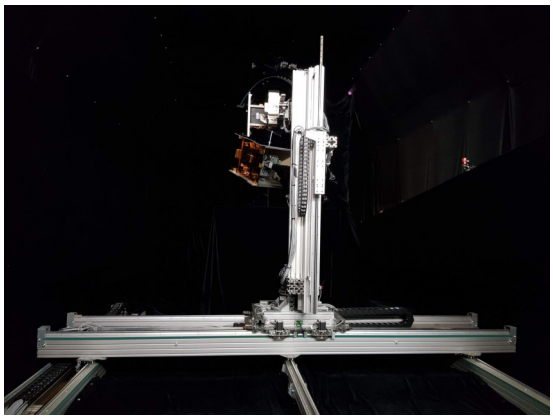
The laboratory, at the moment, includes a scaled replica of a generic telecommunication satellite usually located in GEO, and its surface is covered with reflective materials that simulate Multi-Layer Insulation (MLI) blankets. This material usually covers almost all exposed surfaces expected to be illuminated by the Sun, and its reflective characteristic decreases the thermal exchange with the satellite.

The edges of the target satellite were covered with reflective markers fig. 3–4a and were needed by the OptiTrack if it is to be used to detect the structure's position. As these were not necessary, they have been removed.

The chaser, visible in fig. 3–4b, is a mobile platform that simulates a chaser satellite around the target. It can change its position and orientation.



(a) Target with OptiTrack markers



(b) Chaser

Fig. 3–4: RACOON-Lab's target and chaser

3.4 Sun and Earth's Albedo Simulators

Key functionality for this thesis is the ability of the laboratory to reproduce the orbital light conditions that can be found in a range of situations. Along the ring around the chaser and target, it is possible to move two lamps that simulate the presence of the Sun and the effects of the Earth's albedo (visible in fig. 3–5), and offer a comparable spectrum. The position of these lamps can be changed by shifting them along the rail, while their intensity can be adjusted via software. At the moment, the Sun simulator's bulb suffered a catastrophic failure, making it unavailable.

3.5 Software Interface

Given the ever-changing nature of the RACOON-Lab, its control software is composed of a series of programs, each able to control a subset of its functionality.

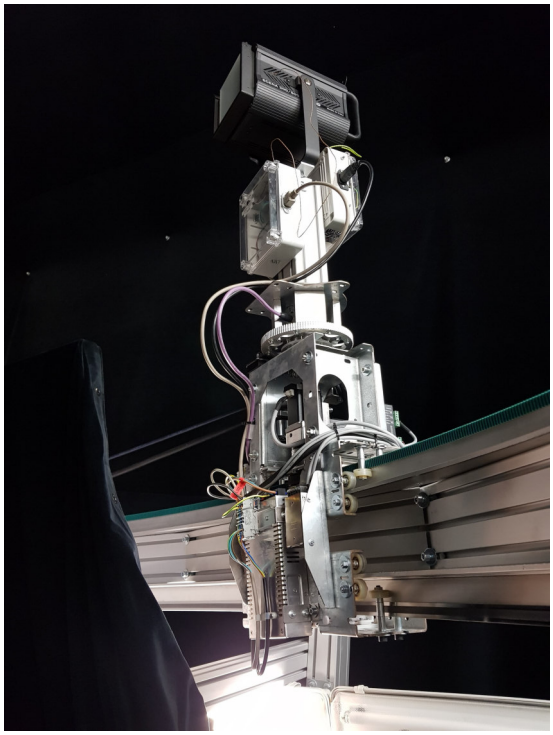


Fig. 3–5: Earth's albedo simulator

The main control interface of the laboratory is made of a LabVIEW Graphical User Interface (GUI) that shows the current state of the laboratory and empowers the user to modify the position of each controlled axis and to choose the velocity at which the selected configuration needs to be reached. Only the Sun and albedo simulator demand an additional program that can show the absolute value of their axes in meters (exes 10 and 12) for their position along the rail and degrees (axes 11 and 13) for their orientation relative to its tangent line. Additionally, a supplemental application is used to control the intensities of the lights, and allows to set a value (for each one) between $0\,\%$ and $100\,\%$. This percentage is relative to the intensity of the real reference source.

3.6 Sensors

For this data gathering campaign, three sensors have been taken into consideration: the Stereolabs ZED camera (the subject of previous theses and to be used as reference), the Intel® RealSense™ Depth Camera D435 and the Hypersen Solid-State LIDAR HPS-3D160. For this campaign was also dropped the previously integrated Microsoft Kinect for Xbox One, as Microsoft [30] does not anymore support it, and the LIDAR Hokuyo UST-2, as it was no longer required.

3.6.1 Stereolabs ZED

The ZED is visible in fig. 3–6. As it was subject to previous studies [15], it will be only briefly reintroduced. It is a 3D camera for depth sensing and motion tracking [5]. It uses its two optical sensors to detect the scene from two different perspectives and then merge the results in a depth map. It can record video with a resolution up to

1920×1080, and its system of reference is centered on its left optic.

Fig. 3–6: Stereolabs ZED [5]

The software can communicate with the camera via the Stereolabs SDK.

3.6.2 Intel® RealSense™ Depth Camera D435

The Intel® RealSense™ D435 is an infrared stereo camera able to produce a depth camera of the scene. It can achieve such a task thanks to the use of the D430 module [9], made, as visible in fig. 3–7, of two identical Infrared (IR) sensor (left imager and right imager) and an infrared projector, situated between them, able to project a non-visible static IR pattern. A color sensor is also available on the side of this module. During the acquisition process, the IR sensors capture the scene, and the integrated depth imaging processor calculates depth values for each pixel in the frame, by correlating points on the left image with the same one recorded via the right image. This process is done in real-time and on the camera itself, so it is not possible to record the video streams for a later elaboration (like what happens with the ZED camera). The pattern projected by the IR projector is also not required for the process to occur (unlike other sensors, for example, the original Microsoft Kinect [31]), but its pattern increase the quality of the depth map if the scene presents a featureless scene or it has a low illumination. The color sensor is used to provide texture information to be superimposed on the depth image to create a cloud of color points but not for its creation. Since it can record a significant amount of data, the device offers a USB Type-C interface, where it is also able to draw power for its operations.

Fig. 3–7: Intel©RealSense™D435 [6]

The center of the system of reference of the resulting depth map is, by default, located at the center of the left IR camera, near the color sensor.

The main features and various Field of View (FoV) of the camera modules are given in table 3–1.

Like with the ZED, the software can communicate with the camera via the RealSense SDK.

Tab. 3–1: Main D435's parameters [9]

Parameter	Value
Depth FoV HD	H: $87° \pm 3°$ / V: $58° \pm 1°$ / D: $95° \pm 3°$
Depth FoV VGA	H: $75° \pm 3°$ / V: $62° \pm 1°$ / D: $89° \pm 3°$
IR Projector FoV	H: $90°$ / V: $63°$ / D: $99°$
IR Resolution	1280×800
IR maximum frame rate	90 FPS
IR F Number	$f/2.0$
IR Focal Length	1.93 mm
IR Projector Power	360 mW average, 4.25 W peak
Color Resolution	1920×1080
Color maximum framerate	60 FPS
Color F Number	$f/2.0$
Color Focal Length	1.88 mm

3.6.3 Hypersen Solid-State LIDAR HPS-3D160

The Hypersen Solid-State LIDAR HPS-3D160 is a solid-state LIDAR sensor, visible in fig. 3–8, based on the Time of Flight (ToF) principle. Thanks to its infrared optical lens, the producer claim it can measure distances up to $12\,\mathrm{m}$ on $90\,\%$ reflective white targets. It can communicate with a computer, among other options, thanks to a USB 2.0 interface.

It is composed of two sections: the lower one (near the cable connector) stores the infrared Vertical Cavity Surface Emitting Laser (VCSEL), while the upper one hosts the IR detector and is where the center of the system of reference of the resulting depth map is located.

The main features of the LIDAR are given in table 3–2. The sensors can enable various types of High Dynamic Range (HDR) to improve the quality of the results.

Like with the other sensors, the software can communicate with the sensor via the HPS SDK.

Fig. 3–8: HPS-3D160 [7]

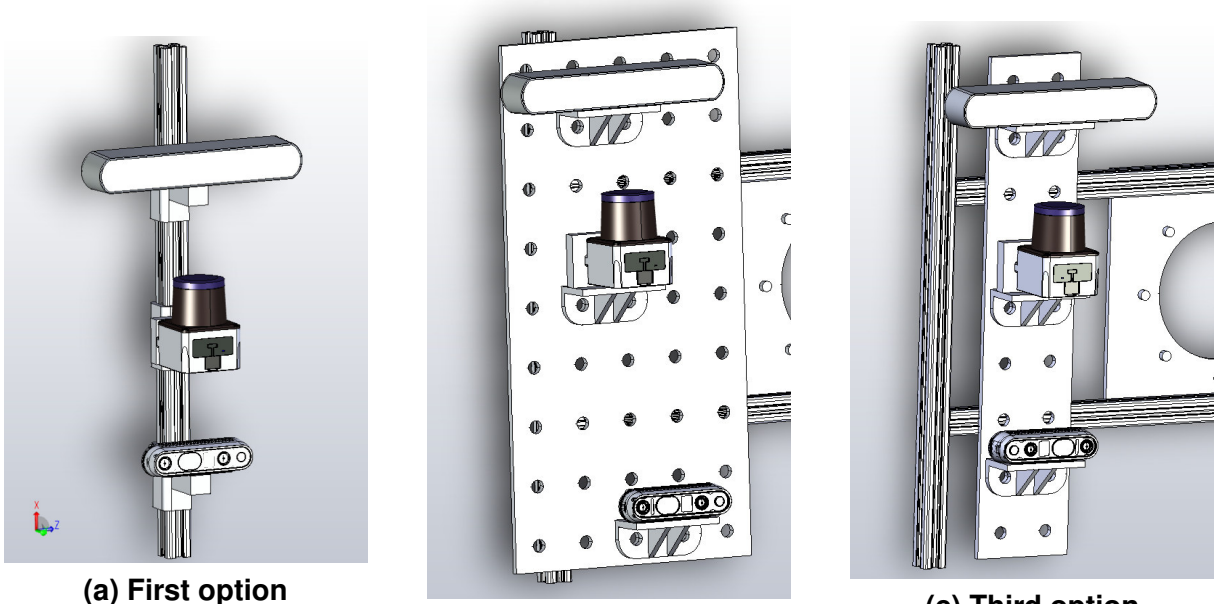
Tab. 3–2: Main HPS-3D160's parameters [10]

Parameter	Value
Resolution:	H: $160\times$ V: 60
Power consumption	From 0.7 W to 6 W
Infrared VCSEL emitter	850 nm
Emitting angle	H: $76°\times$ V: $32°$
Measurable distance range	From 0.25 m to 12 m
Output frame rate	Up to 35 FPS
Operating mode	Normal / Auto-HDR / Super-HDR / Simple-HDR

4 Hardware system design

4.1 Device supports

As already mentioned, given the ever-changing nature of the laboratory, a flexible support system had to be implemented that could handle previous sensors and future expansions of further devices in a simple but flexible way. After analyzing the existing structure, a preliminary design of a series of options (as visible in fig. 4–1) was designed. In all cases, the existing structure was maintained to support the new mounting system, but with different purposes. In the first case, the outermost vertical beam was to be used as a sliding track where the sensors were to be mounted, thus allowing for a single degree of freedom in its position. In the second and third configurations (different only in size), a flat plate was designed to support the various systems, while the previous vertical beam was only used as a reinforcement structure capable also of accommodating the cable assembly. This plate allowed to place the sensors on a bi-dimensional plane, thus allowing for more flexibility in their positions.



(a) First option

(b) Second option

(c) Third option

Fig. 4–1: 3D models of the possible hardware solutions analyzed in the preliminary phase

After considering future expansion scenarios and the available mean of production, an evolution of the last two ideas has been developed. The second iteration of this configuration is visible in fig. 4–2. The plate presents a grid-like system that fills all the available space, where each anchoring point presents a circular and an elongated slit. The first feature is meant as a support structure, where a bolt can be inserted for holding the device's mounting in place, where the second one can receive the support's ledge to keep the whole structure aligned with the rest of the robot. Each sensor

requires a 3D-printed support to be held in place, and it can be custom-built to best suit the required configuration.

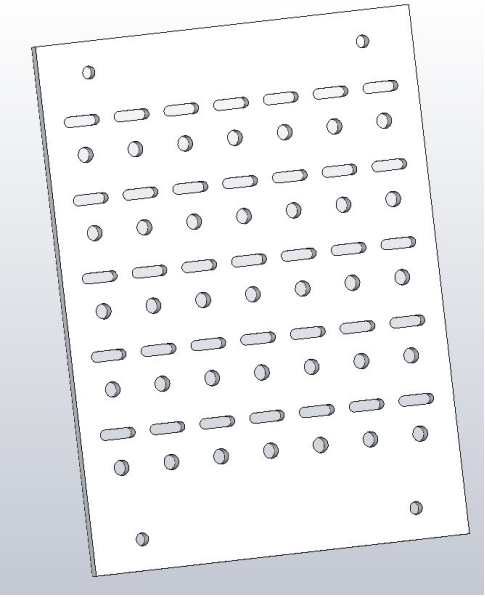


Fig. 4–2: First implementation of the plate support

While this implementation could theoretically support any orientation of the devices thanks to their associated 3D-printed counterpart, a further design has been developed to allow for more flexibility. In this final version, visible on fig. 4–3, the grid has been adjusted to have an equal spacing of $2\,\mathrm{cm}$ between each central hole. In this way, it is possible to turn the support of $\pm 90°$ or $180°$.

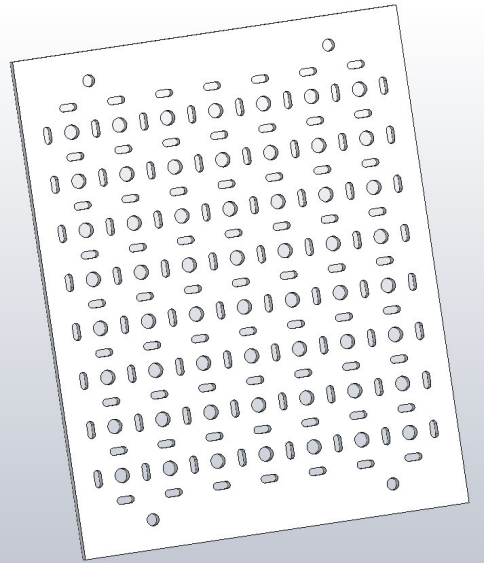


Fig. 4–3: Final implementation of the plate support

The plate is made of acrylic plate laser-cut with a depth of $5\,\mathrm{cm}$ (as these plates are readily available in the laboratory, to ensure the structural strength on the plate), while

the support is printed of Polyethylene Terephthalate Glycol-modified (PETG). The combination of the plate and some supports is visible in fig. 4–4.

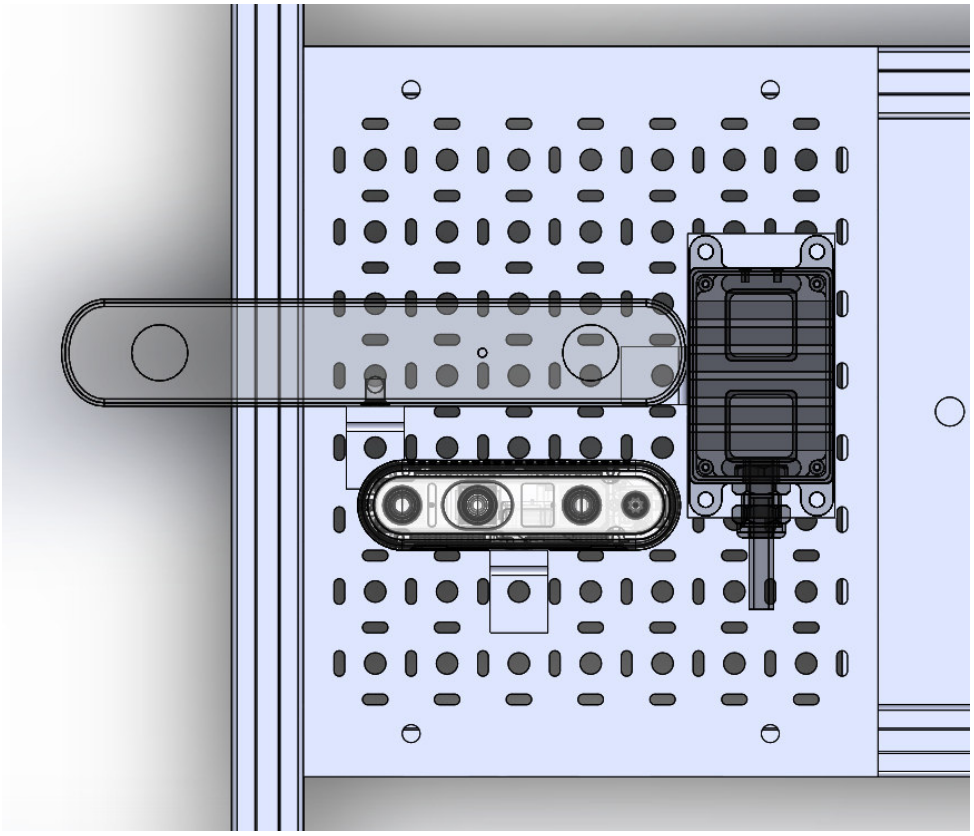


Fig. 4–4: Plate with supports, ZED, D435, and HPS-3D160 in their final disposition

4.2 Computer support

In order to allow easier management of the chaser's computer, a new support system had to be developed in order to replace the already installed cable ties and adhesive tape. After various considerations, a double 3D printed support system (made of the same material as the mountings of the sensors) was developed. Each piece is attached to one of the horizontal beams and wraps three sides of the computer. This shape prevents obstruction of the numerous ports and openings on the machine, and the physical separation between the two halves permits a secure installation of the entire system, without the need of precise manufacturing of a contact zone between the two parts. An image of it, together with the complete implementation of the support, can be seen on fig. 4–5.

The final setup, with all the devices and relative supports in place, is visible in fig. 4–6.

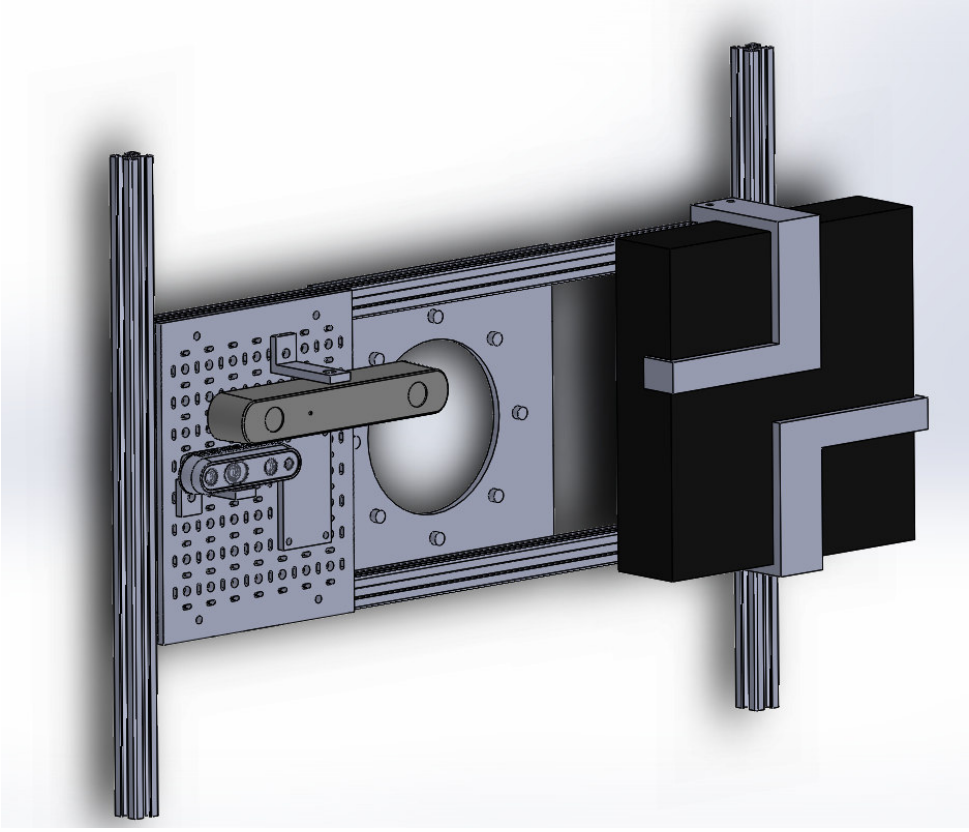


Fig. 4–5: 3D of the supporting structure with the plate, the sensors in a valid position (but not the chosen for the experiment), the computer and its support

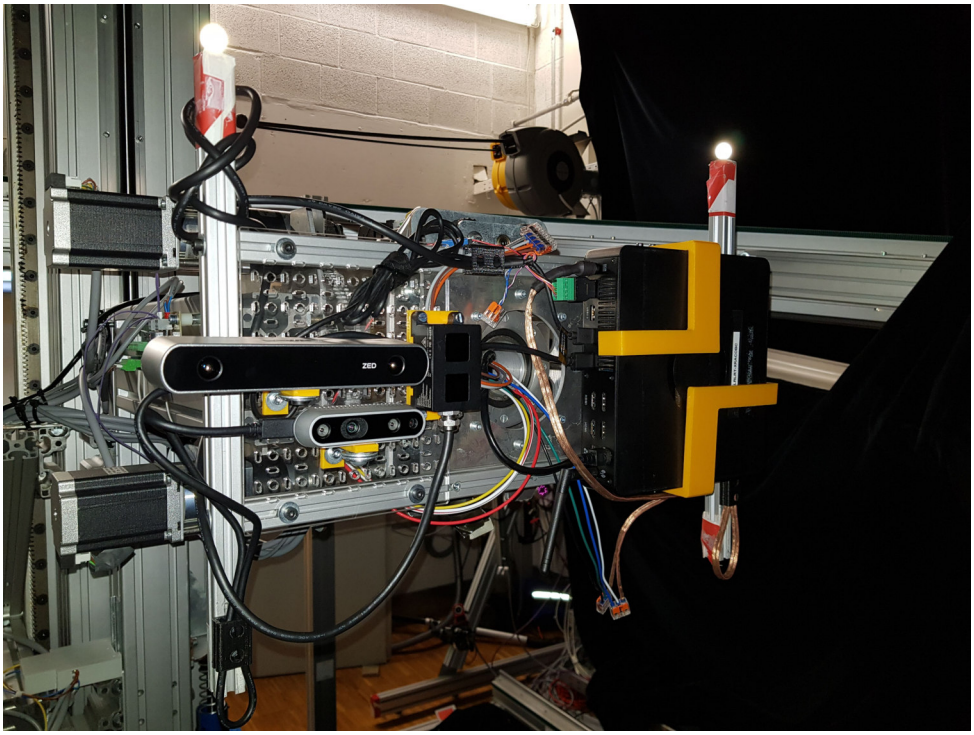


Fig. 4–6: Final hardware setup with the support plate and the sensor on the left and the computer and its supports on the right

5 Software system design

5.1 Data storage format

A considerable amount of time has been spent to choose and implement the right data management system, as a large amount of data that is expected to be recorded and processed. Since each device can produce data in its format and the results of the various calculations must be stored, a unified file format was deemed necessary to present a consistent interface with all the various types of data available. Equally important, this format needed to be able to store the information obtained for future use without the need for all the various dependencies required to read the sources and, therefore, without being tied to a closed proprietary file format.

After much consideration and empirical evidence, the previous mix of text-based files and single-frame images were discarded, and a structure based on HDF5 was chosen. Developed by The HDF Group [32], HDF5 is a hierarchical binary data format designed to store and organize large amounts of data in a self-documenting way. In such a format, as visible in fig. 5–1, each node of the data structure is called a group, which can host additional metadata and links to other groups and datasets (where the data is stored). Integrated utilities with the HDF5 library also allow for lossless compression of the datasets, while keeping access to the stored information transparent to the applications (with a time delay related to the chosen compression).

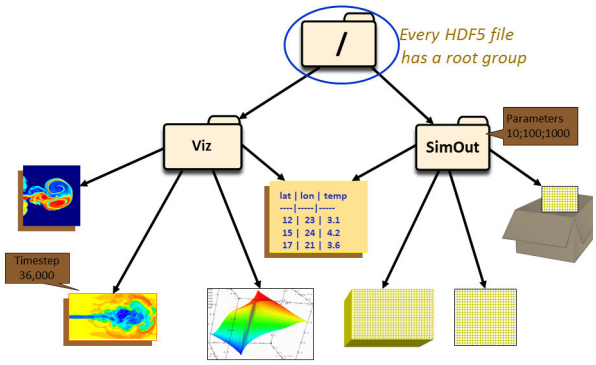


Fig. 5–1: Example of a HDF5 file, with the root, some connected groups and their relative datasets and attributes [8]

To allow for a more straightforward interpretation of the data and to increase the number of analysis tools able to read the information, some sections of this system use the Network Common Data Form version 4 (NetCDF4), developed by the University Corporation for Atmospheric Research (UCAR) [33]. A NetCDF4 file is also a HDF5 file. However, it uses only a subset of the many available features, thus simplifying the organization of the data but without compromising the access mode or adding additional dependency only for reading (or, if demanded, writing) the dataset. While also offering groups, in a NetCDF4 file, the data of the single measurements are stored in

variables as a N-dimensional matrix, where each axis has an associated dimension that describes the data.

The main disadvantage with the embedding of NetCDF4 is that it prohibits links to external files. Due to the expected final size of the final dataset of more than a terabyte, a single file to store all the data was considered unwise (fearing data loss or information corruption in the future), giving preference to a system where multiple files are interconnected. While the HDF5 library supports this distributed system in a completely transparent way (allowing the user to work with the data as if it were in a single file), the NetCDF4 library is unable to decode the resulting files with links to others but has no problem reading the target file. So a tree structure has been developed, where a HDF5 root file hosts the main structure of the dataset, but the final groups (with all the data) are in NetCDF4. In this way, reading the root, one can follow the data structure to the required group, where the location of the hosting file can be retrieved and shared with the NetCDF4 library.

The final configuration of the root file (without presenting the metadata and thus to avoid excessive clutter of the tree) is visible in fig. 5–2. The gray groups represent the external links, while the dashed arrows indicate the presence of several nodes simultaneously for various sessions or calculated paths.

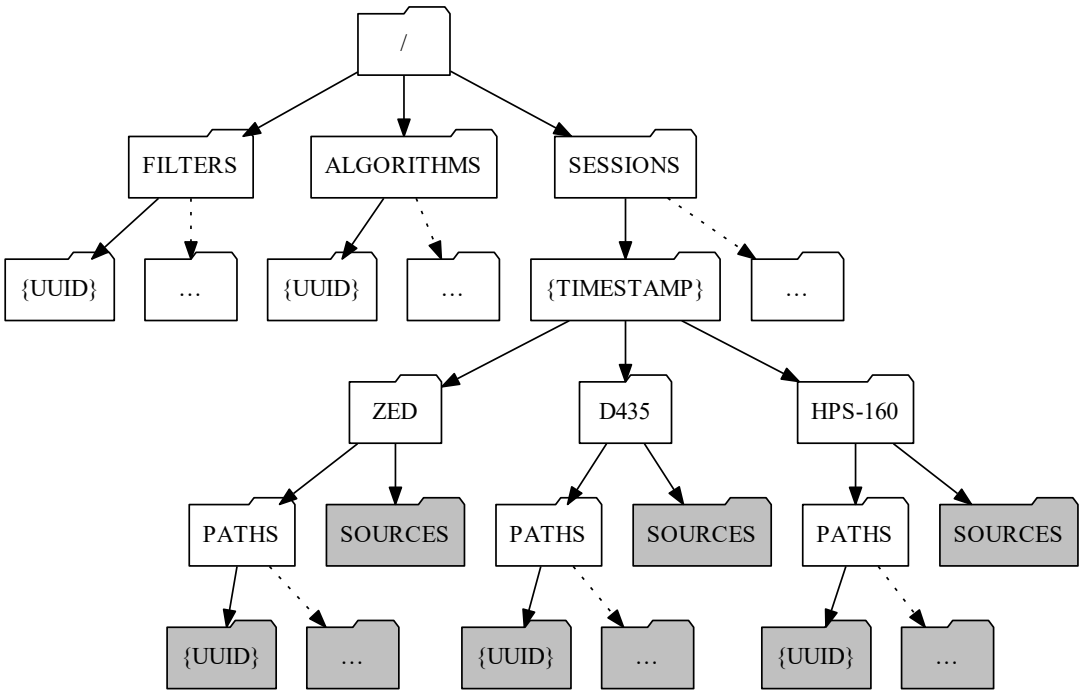


Fig. 5–2: Data structure of the root file; each gray group is stored on a different file

From the root group, it is possible to reach three different subgroups: *FILTERS*, *ALGORITHMS* and *SESSIONS*. Under the first one are stored new groups, one for each

available filter, together with the metadata needed for their application. Their quantity and what they store are filter-specific, but each of them has the field Identifier (ID) (a unique Integer number to identify each filter among others) and a field *alias*, which describes the type of filter and help the software to classify it (e.g., BoxFilter or None at the moment). The ALGORITHMS group follows the same logic but stores information about the algorithms used to calculate the paths (for example, DIFODO or ZedSDK).

The *SESSIONS* group stores a new group for each recorded data session and contains metadata describing the state of the laboratory at the time of the recording, such as the session timestamp (also visible on the group name), lamps positions, intensity and other information, and a group for each recording device. For each stored instrument are available sources recorded under the group *SOURCES*, stored in a different file, and all paths calculated under *PATHS*. Among the metadata related to them (each path is stored on a separate file, as its size could be substantial if additional information is stored), the ID of the filter and the algorithm that were used to calculate their points are recorded.

For each session, all recorded data is stored in a single NetCDF4 file but, thanks to the links available from the HDF5 format, it appears as part of the primary root and can be reached from different paths along the tree. Each sensor has its specific structure and can have (but is not required, if these fields are not useful) the groups visible in figs. 5–3 to 5–5. Rectangular fields indicate datasets, while a dotted one implies that it is also a dimension relative to another dataset. Similarly, on fig. 5–6, one can see the group for a path.

It is crucial to notice that, even if the data structure is the same for each sensor, no conversion has been carried out between the original file and the information here stored. In this way, while it can sometimes be inconvenient to read a particular format of data (like the depth map) and conversions need to occur on the fly, no data loss occurs in the storage process, thus allowing the original output to be analyzed without further uncertainty.

As mentioned above, each axis of a dataset is associated with a dimension that describes it. Since all recordings are time-dependent, the first axis of each matrix is described by a timestamp dimension (represented as the number of seconds from a start date stored in an attribute). If the matrix stores an image, the dimensions *u* and *v* indicate the number of pixels along each dimension, and if each position stores the color channels, it is possible to read which one is it thanks to the field (red, green, blue or transparency).

The position and orientation found on the path are also organized by time, while an additional dimension describes the order of storage of the 3D/4D space (x, y, z, and, for a quaternion, r).

To increase the IO performances of the data structure, and to enable the possibility to compress the stored data, the chunking of the NetCDF4 datasets has been enabled. Each chunk has the size of the expected frame (being it 2D or 3D) and a selected number of timestamps. Given the expected number of frames to be stored (between

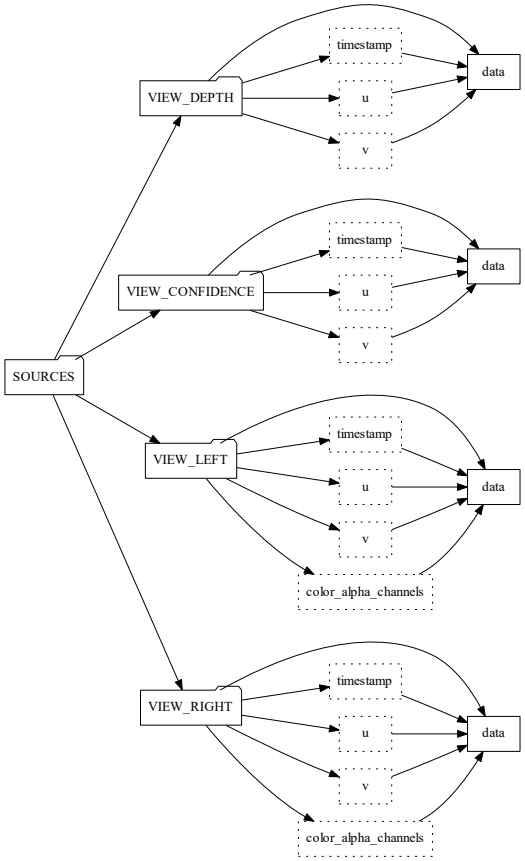


Fig. 5–3: Structure of the ZED groups

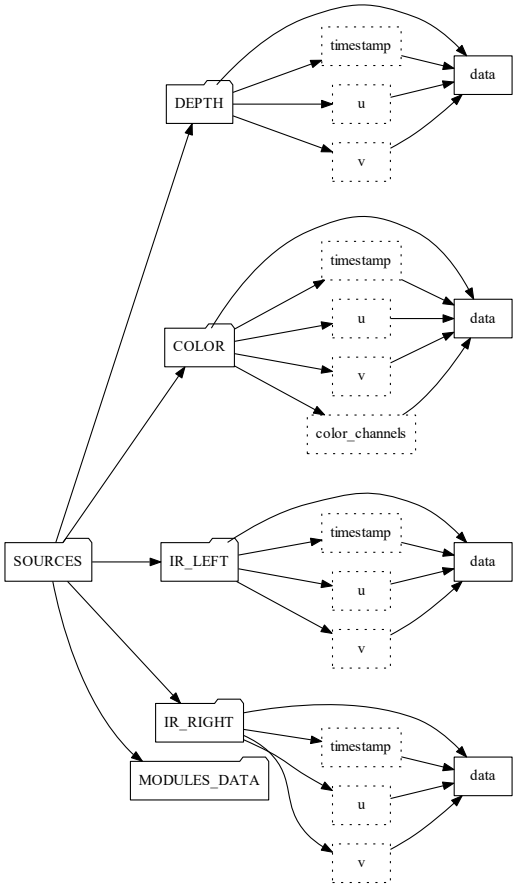


Fig. 5–4: Structure of the D435 groups

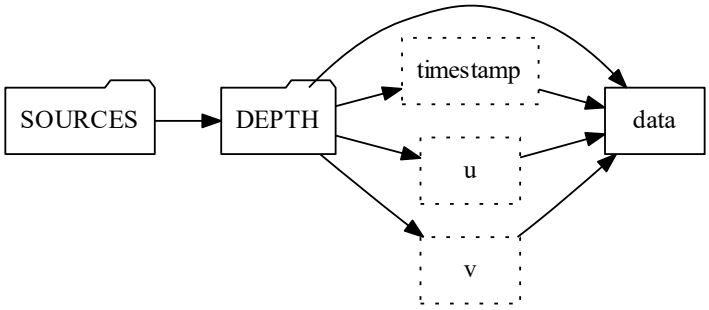


Fig. 5–5: Structure of the HPS-160 groups

2500 and 3000), each chunk can thus store 500 frames. While this choice can occur in disproportionate file size for the shorter recordings (as a multiple of 500 frames will always be stored on the storage device), the writing operations need to allocate additional space only six times, while the compression algorithm benefits from bigger

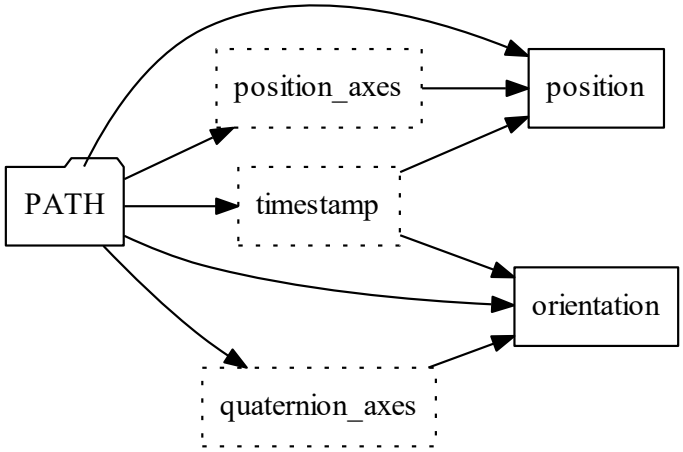


Fig. 5–6: Structure of the path group

chunking in creating smaller files. It is still worth mentioning that this value has not been subject to rigorous measurements, as optimal IO performances were not one of the objectives of the thesis, and possible better values for access time or compression levels may be present.

As it may be visible, the results of the metric are not mentioned in the data structure. The various parameters are not stored since each path requires a fraction of a second to compute its parameters by calling the relative Python function. The location of the stated procedure is mentioned in section 5.2.9. The computed values are very fast to compute and involve a minuscule amount of data, and because of the expected use case of the parameters is their visualization or their immediate analysis, this approach guarantees quick usability without burdening the data structure of more complexity or the need to travel the data tree to retrieve the results. In light of future development of said metric (as is further discussed in section 9.1), this also prevents confusion in recording the used implementation for the computation of the parameters. This choice does not hinder the future reproducibility of the results, as the current implementation of the metric will always generate the same results.

5.2 Software Utilities for Data Management

Thanks to the data structure just described and the fact that it is implemented in an open format, a set of software tools has been developed to manipulate this structure, starting from the original implementation of the previous works. These programs have been developed in C++17, where there was previously an original program and for performance reasons, while for new software, Python 3.8 has been used to simplify code interpretation and speed up development time. To store settings and additional metadata (in a temporary format until the final data are saved in the HDF5 file), the

YAML Ain't Markup Language (YAML) file format is used. The various interconnections between them are visible in fig. 5–7.

The Python utility *dataset_editor.py* is the main tool for editing the root HDF5 file and its various NetCDF4 dependencies. It allows the user to create new filters/algorithms, link converted sources data, and export paths and videos in human-readable representation.

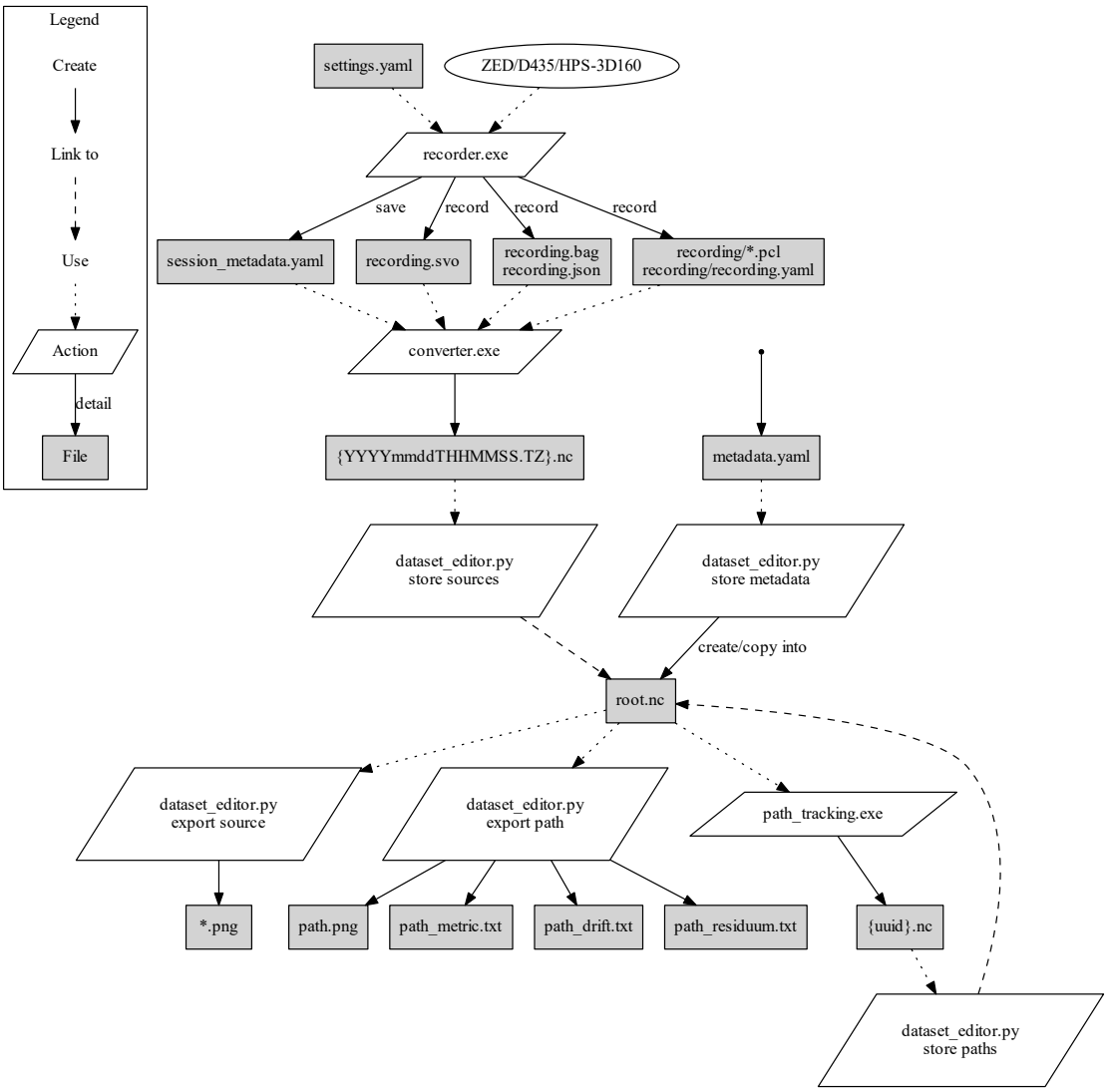


Fig. 5–7: Software architecture and data flow of the entire data management toolchain. The external library with additional functionalities is not displayed in this graph, as it is already employed by the various command-line tools here visualized

5.2.1 Recorder Application

The C++ program *recorder.exe* is tasked with the realtime recording of the native files from the various sensors. It requires, as an input, a configuration file where are stored

the devices settings and generic metadata. These, together with the recordings, are saved on a folder with the recording timestamp as name, formatted following the ISO 8601 [34] basic format representation.

The program operates via a command-line interface that accepts various parameters, as visible in the following list:

- duration: Unsigned integer that indicates how many milliseconds the recorder needs to record the data. It is an optional parameter and, if it is not provided, allows the program to record until the Enter key is pressed;
- output_path: String that indicates where the resulting folder with all the recorded data will be created;
- settings_path: Path to the settings file (in a YAML format) with the devices settings and additional metadata to be stored with results;
- key: String parameter that can be absent or repeated multiple times and allows to store the name of metadata in addition to the ones indicated in the YAML settings file;
- value: String parameter that must be used the same number of times of the key parameter and allows to store the value of metadata in addition to the ones indicated in the YAML settings file. The final association with the key parameters follow their order;
- hps160_name: Name of the HPS-3D160 device for the operating system. It is not required but allows the user to speed up the starting of the program.

An example of its use is visible in listing 5.1:

Listing 5.1: Example of a program recording 90 seconds of data

```
recorder.exe`
  --duration=90000`
  --output_path=D:\sources`
  --settings_path=settings.yaml`
  --key=albedo_position --value=0.0`
  --key=albedo_position_unit --value=deg`
  --hps160_name=\\.\COM3
```

The file settings.yaml is a YAML document that indicates which devices should record and with which configuration. It also allows specifying metadata that needs to be stored with the results. An example of its structure, where all the devices are enabled, is visible on listing A.1.

5.2.2 Converter Application

Starting from the data recorded by the recorder program, the C++ application *converter.exe* can convert and collect all the various native formats and metadata to a single NetCDF4 file, ready to be connected to the root file HDF5 with another application.

Like the recorder, the converter is operated via a command-line interface with the following parameters:

- source_dir: Path where the recorder can find previously recorded source data;
- output_dir: Path to which the recorder will save the newly converted dataset;
- ZED_channels: List of channels (separated by a comma) to be exported from the ZED file. It is possible to choose between VIEW_DEPTH for the depth map, VIEW_LEFT for the left camera video, VIEW_RIGHT for the right camera and VIEW_CONFIDENCE to get the confidence of the depth values determined by the algorithm;
- D435_channels: It has the same meaning as the ZED_channels parameter, but it works for the D435 stereo camera. It is possible to select DEPTH for the depth map, LEFT_INFRARED for the left camera IR video, RIGHT_INFRARED for the one from the right camera and COLOR to get the images from the color camera;
- HPS160_channels: It has the same meaning of the ZED_channels and D435_channels parameters, but it applies to the HPS-3D160 LIDAR. It supports only the DEPTH keyword.

In addition to the various channels' names, it is also possible to use the keyword NONE to disable the exporter for a specific device.

An example of the converter used is visible in listing 5.2:

Listing 5.2: Example of a program exporting data

```
converter.exe`
  --source_dir=D:\sources\20200210T100154.0631443+0000`
  --output_dir=D:\data`
  --ZED_channels=VIEW_DEPTH`
  --D435_channels=DEPTH,COLOR`
  --HPS160_channels=NONE
```

5.2.3 Path Tracking Application

The C++ program *path_tracking.exe* is tasked to calculate the path followed by the device, making use of the converted NetCDF4 dataset. It requires the user to specify the filter that will be applied to the data before feeding them to the selected algorithm that should be used to compute the result.

The output of this program is also a NetCDF4 file, and it stores the computed path and orientation of the device, together with information about the algorithm and filter employed (and other possible extracted information). It allows for the following configurations:

- sensor: Name of the sensor to analyze (to choose between ZED, D435, and HPS-160);
- root_path: Path where to find the root HDF5 file;

- output_folder: Where to store the calculated path
- session_name: Name (timestamp) of the session to analyze;
- filter_id: ID of the filter to be used;
- algorithm_id: ID of the algorithm to be used;
- svo_file: Path to the SVO file that needs to be used if the algorithm choose is the ZED SDK;
- from_timestamp and to_timestamp: Optional timestamp range (in the full format ISO 8601 with timezone) where the path should be calculated.

An example of its use is visible in listing 5.3:

Listing 5.3: Example of the path tracking program usage

```
path_tracking.exe‘
  --svo_file=D:\sources\20200210T100154.0631443+0000\recording.svo‘
  --sensor=ZED‘
  --root_path=D:\root.nc‘
  --output_folder=D:\recordings\results\‘
  --session_name=20200210T100154.063144300+0000‘
  --filter_id=0 --algorithm_id=0
```

It is essential to notice that, while determining the paths, missing depth frames can be found in the sources. In these events, the program will skip the computation for the selected step. More on the consequences of this aspect will be later analyzed in the results.

5.2.4 Store Metadata Utility

The *store metadata* command allows the user to create the root file (if it does not exists on the specified path) and to add additional filters and algorithms information to it. It requires the path to the HDF5 root and to the file where the metadata to add are specified. An example of its calling convention is visible in listing 5.4.

The data file, an example of which is visible in listing A.2, is written in YAML. For each algorithm/filter entry an *alias* field is present that indicates its type, while all the other fields are implementation-specific for the entry.

If the root file does not exists, it is created by this command, together with its main groups *SESSIONS*, *ALGORITHMS* and *FILTERS*.

For each entry in the YAML file, a group is created on the respective root (*ALGORITHMS* or *FILTERS*) with an Universally Unique IDentifier (UUID) as name (as each group requires an unique identifier). All fields are then copied from the file, with the addition of an additional unique incremental ID to be used as the primary key for reference.

Listing 5.4: Example of how to store metadata on the root file

```
dataset_editor.py store metadata‘
  D:/recordings/root.nc data.yaml
```

5.2.5 Store Sources Utility

The *store sources* command allows connecting the previously converted sources data to their relative positions in the main tree of the root file. It requires the path to the root file and the session file created by the *converter.exe* program. An example of its calling convention is visible in listing 5.5.

Listing 5.5: Example of how to store metadata on the root file

```
dataset_editor.py store paths‘
  D:/recordings/root.nc D:/recordings/results
```

For each directory in the specified path, the program creates, if it does not already exist, a new group in the *SESSIONS* group for the session. The name for each session is its timestamp. Another group called *SOURCES* is then created inside of it. Inside this last added group, a link for each sensor group to it via an external link is then added.

5.2.6 Store Paths Utility

The *store paths* command, in a similar manner as *store sources*, allows to store the previously computed paths in their relative position on the root tree. An example of its calling convention is visible in listing 5.6.

Listing 5.6: Example of how to store paths on the root file

```
dataset_editor.py store paths‘
  D:/recordings/root.nc D:/recordings/data
```

As with the sources, for each path in the folder, the program creates a new group in *SESSIONS*. Another group called *PATHS* is then created inside of it (if it was not created previously). The path file is then connected here with an external link (with an UUID as name).

5.2.7 Export Source Utility

The *export source* command allows the user to export a channel of a device of a session in a sequence of *png* images. An example of its calling convention is visible in listing 5.7. It requires, in order, the following parameters:

1. Path to the root dataset;
2. Path where the results will be saved;
3. Name of the session to export;
4. Name of the sensor to export;
5. Name of the channel to export;

6. from_value and to_value: first and last value of the colorband (only for fields with a single value for each pixel);
7. scale_factor: how much the width and height of the image have to be scaled (1 by default).

Listing 5.7: Example of how to export the depth channel of a HPS-3D160 from a session

```
dataset_editor.py export sources‘
  D:/recordings/root.nc D:/recordings/pngs/‘
  20200219T103238.261749100+0000 HPS-160 DEPTH‘
  --scale_factor=6 --from_value=0 --to_value=7
```

5.2.8 Export Path Utility

The *export sources* command allows the user to export information on the path, such as plotting or metrics data. An example of its calling convention is visible in listing 5.7. It requires, in order, the following parameters:

1. Path to the root dataset;
2. Path where the results will be saved;
3. Name of the session to export;
4. Name of the sensor to export;
5. path_name: Optional name of the path to export. If this field is not provided, the whole series of paths of the selected configuration will be exported.

Listing 5.8: Example of how to export information of a path

```
dataset_editor.py export path‘
  D:/recordings/root.nc D:/recordings/metrics‘
  20200219T102226.4088197+0000 D435
```

5.2.9 Additional libraries

In addition to the stated software tools, a series of Python modules are available. These libraries host all the internal logic used by the *dataset_editor.py* commands, plus additional functionality available for developing additional scripts. The implementation of said modules is based on various open-source libraries listed in the *Pipfile*, among which are worth mentioning:

- *NumPy* [35]/*SciPy* [36] for matrix manipulation;
- *h5py* [37] for managing the HDF5/NetCDF4 files;
- *Pandas* [38] and xarray [39] to analyze the datasets in memory;
- *Dask* [40] to parallelize operations and manage the bigger datasets;
- *Matplotlib* [41]/*Altair* [42] to visualize the results.

The libraries are stored in the *utility.py* for general utilities (like converting the old data format to NetCDF4 files) and in *plotting.py* for plotting.

Among the available functions, the most important is *bestfit_circle*, listed in listing 5.9, that calculate the metric parameter for a path. The parameter *ds* stores a xarray dataset of a path group, while *angular_velocity* saves the angular velocity of the satellite, even when the frames, for the analyzed trajectory, were skipped.

Listing 5.9: Definition of the function that calculate the bestfit-circle parameters

```
def bestfit_circle(
        ds: xr.Dataset,
        angular_velocity: float
) -> xr.Dataset
```

6 Experiment Design

6.1 Selection of Device Parameters

The devices tested offer various degrees of configurations to fit better the fulfillment of the requirements and the condition of the environment in which they need to operate. Their parameters needed to be determined before the gathering of the data, as they deeply influence their quantity and quality.

The selection of a valid combination of parameters has been made in two phases: in the first one, a subset of significant parameters for each device has been chosen after reading the documentation available. This selected list of field tests has then been tested in combination with the partial measurement pipeline, to finally select the configurations to be used in the measurement campaign.

6.1.1 D435

The D435 stereo camera allows changing more than 100 parameters for its infrared cameras, depth sensor, color camera, projector, and depth map format. The settings selection procedure had two targets: to check which configuration was able to gather the more uniform flux of data (without losing frames) and which gave the best quality of the results. The parameters tested and their options are visible in table 6–1.

Tab. 6–1: Parameters to be tested for the D435

Parameter	Tested values
Color Resolution	1280×720, 960×540, 848×480, 640×360, 424×240
Color framerate	90 FPS, 60 FPS, 30 FPS
Depth resolution	1280×720, 848×480, 640×480, 640×360, 480×270, 424×240
Depth scale factor	$0.0001\,\mathrm{m}^{-1}$, $0.000\,152\,\mathrm{m}^{-1}$, $0.000\,457\,\mathrm{m}^{-1}$
Depth framerate	90 FPS, 60 FPS, 30 FPS, 15 FPS
Depth preset	High Accuracy (H. A.), High Density (H. D.)
Projector power	0 mW, 180 mW

The depth sensor alone requires the configuration of approximately 40 parameters. The documentation does not offer a clear description of each of them but, to simplify the selection of satisfactory values, presents a collection of pre-configured sets of settings called Presets (which values have been chosen by Intel via Machine Learning methods) that best fits particular scenarios. Among the one directly offered by the SDK and visible in table 6–2, the High Accuracy presets and the High Density preset are considered for the tests, as, following their description, they best suit most scenarios.

Tab. 6–2: Available presets for the D435 [11]

Preset Name	Official Description
High Density	Higher Fill factor, sees more objects. (Ex. BGS and 3D Enhanced Photography, Object recognition)
Medium Density	Balance between Fill factor and accuracy.
High Accuracy	High confidence threshold value of depth, lower fill factor. (Ex. Object Scanning, Collision Avoidance, Robots)
Hand	Good for Hand Tracking, Gesture recognition, good edges
Default	Best Visual appeal, Clean edges, Reduced PointCloud Spraying

As the D435 was chosen because of its ability to produce depth maps, their quality was deemed the primary target in the selection procedure. Because the quality of the depth projection is not a quantifiable metric [43, 44], it is necessary to define one. In the so-called *White wall test*, as visible in fig. 6–1, a flat uniform surface that emits no glare but has high reflectivity is placed at a known distance from the camera is considered, and three parameters are used as a target:

- Z-accuracy: ability to evaluates the depth data with accuracy;
- Fill Rate: percentage of the depth coverage of the image;
- RMS error: spatial depth uniformity.

The RealSense SDK comes with an integrated tool that can calculate and export these parameters, and it has been used to gather information for the analysis procedure.

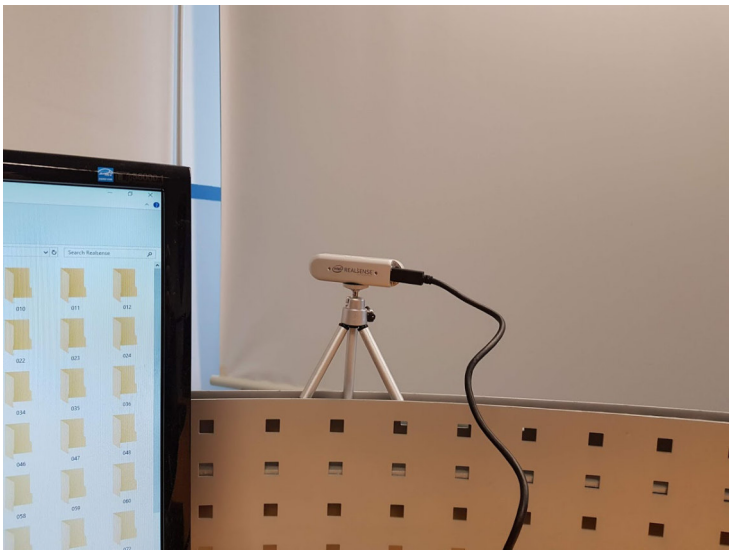


Fig. 6–1: D435 set up to perform the White wall test, where the camera was put in front of a flat projection screen to analyze the quality of the resulting depth map while detecting a known surface

The camera can produce a vast amount of data in a brief window of time, as at full resolutions and framerate, the device can transmit $3.8624\,\mathrm{Gbit/s}$. While this value is within the $5.0\,\mathrm{Gbit/s}$ theoretical limit of the USB 3.0 protocol, sub-sequential tests exhibited considerable drops in the recorded framerate. A series of pre-tests were then made to prevent this phenomenon from happening during the final data-gathering campaign, where significant parameters (see table 6–1) able to change the size of the data were investigated.

The results of the SDK's tool of the *White wall tests* are listed in table B.1.

As visible on fig. 6–2 and confirmed by the Intel guidelines [43], the resolution of 848×480 offers, on average, the best Fill-Rate and the lowest Plane Fit error for both presets, while at the same time also offering one of the highest counts of available pixels (and information).

While the results denote better performances for the H. D. preset, the H. A. has been in the end chosen. This because the metric is not a perfect representation of the final scenario, and, as seen in the previous thesis [15], it is expected much lower confidence of the depth estimations in many sections of the images on the final data.

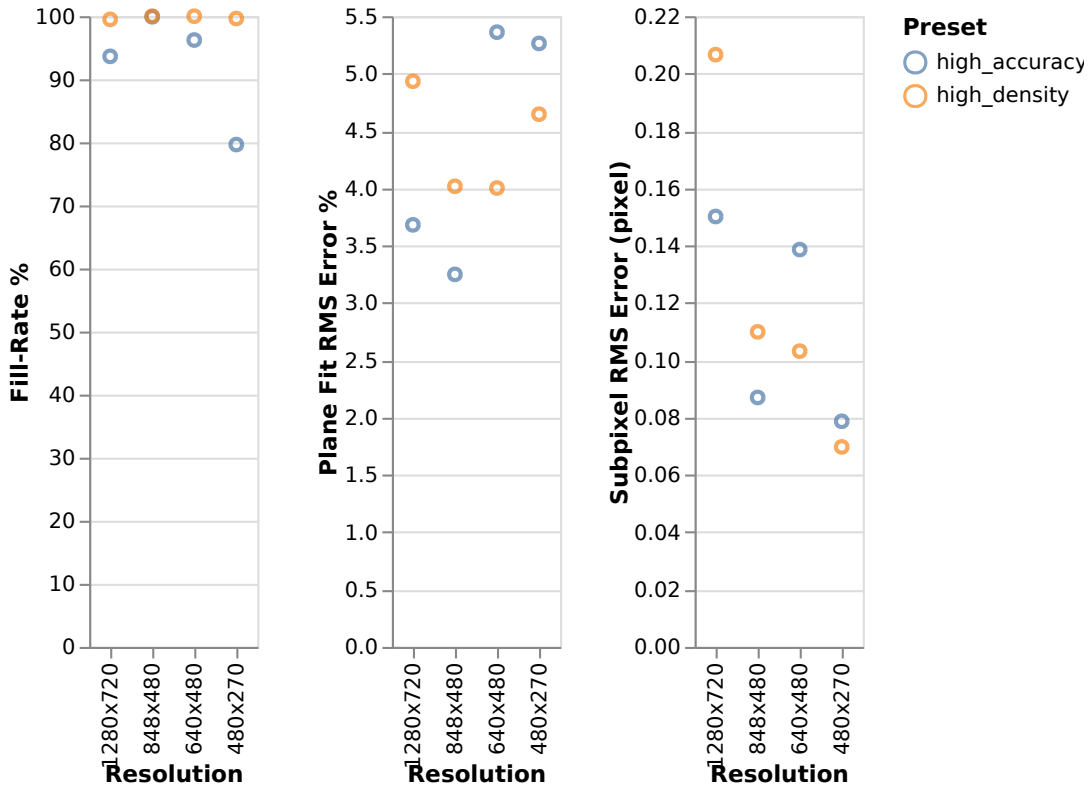


Fig. 6–2: D435 White wall test results for the various metric parameters and both tested presents

Concerning the dropped frames, from the same dataset have been counted the dropped frames in the various configuration, and the results are visible in table B.2, while a summary is represented in fig. 6–3. It is worth noticing that a frame is considered dropped

Tab. 6–3: Standard deviation between different White wall tests for the D435

Resolution	Preset	Fill-Rate	Plane Fit RMS Error	Subpx RMS Error
1280×720	H. A.	0.442 637 %	0.022 718 %	0.001 080 px
	H. D.	0.015 010 %	0.013 778 %	0.000 537 px
424×240	H. A.	2.385 435 %	0.004 284 %	0.000 237 px
480×270	H. A.	2.886 093 %	0.052 493 %	0.000 888 px
	H. D.	0.017 307 %	0.052 567 %	0.000 844 px
640×360	H. A.	0.032 845 %	0.016 918 %	0.000 422 px
640×480	H. A.	0.135 728 %	0.030 329 %	0.001 090 px
	H. D.	0.000 438 %	0.028 044 %	0.000 639 px
848×480	H. A.	0.067 005 %	0.037 973 %	0.001 054 px
	H. D.	0.000 144 %	0.005 990 %	0.000 281 px

if the difference between its number, given by the camera, and the one from the previous is higher than one. As it is possible to see in the plot, in general, the greater the amount of data and the higher is the number of dropped frames. Configuration 2 is the one with the higher information density that does not lose frames with the employed toolchain.

Finally, no significant variations have been found regarding in quality of the result (measured as the percent of the depth map filled with values relative to the whole surface) in the selection of the depth scale factor. The data gathered while analyzing this parameter are visible in table B.3.

After considering the available data, the configuration in tables 6–4 and 6–5 has been chosen. The resolutions and the framerates have been selected to gather the highest possible amount of data without losing frames in the process. At the same time, the selected depth scale factor can cover up to approximately $13\,\mathrm{m}$, thus covering all possible areas of interest but with increased precision compared to the default value of $0.001\,\mathrm{m}^{-1}$ (where the range would be up to $65\,\mathrm{m}$).

6.1.2 ZED

Since the ZED has been tested in the previous thesis, it has been kept with the same configuration. The list of parameters is visible in table 6–6. The only difference has been the change of the depth mode from *PERFORMANCE* to *ULTRA* (as there was no need to prioritize for time since the computation was not done on realtime) and the disablement of the self-calibration procedure, that change the calibration of the camera between each session and make repeatability less precise.

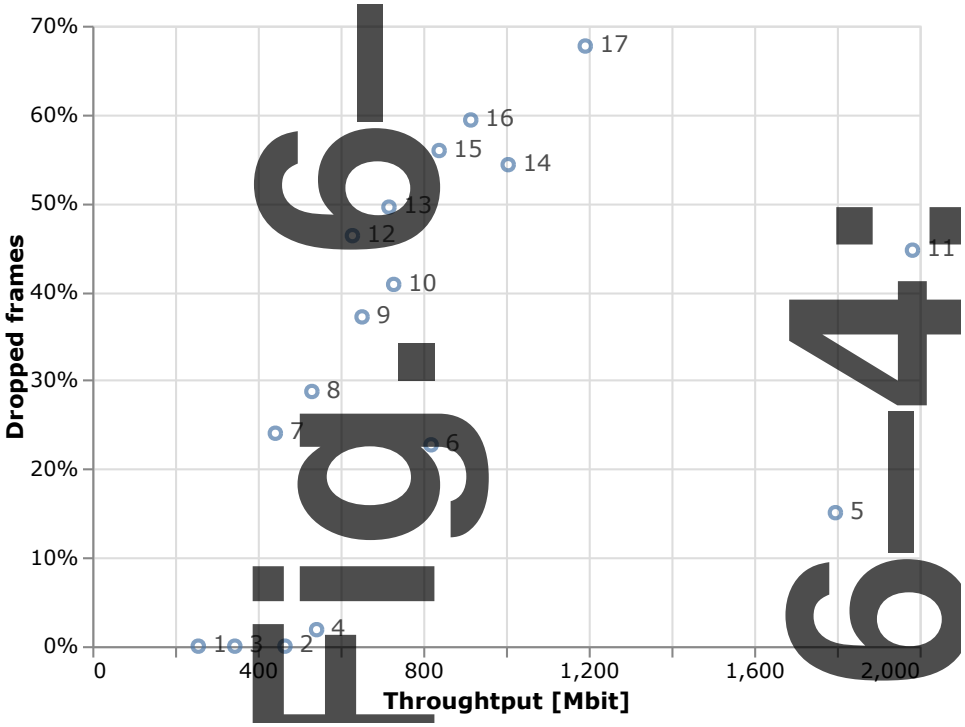

70%
60%
50%
40%
30%
20%
10%
0%
Dropped frames
0
400
800
1,200
1,600
2,000
Throughtput [Mbit]
17
16
15
14
13
12
11
10
9
8
7
6
5
4
1
3
2

Fig. 6.4:

Tab. 6–6: Selected parameters for the ZED

Parameter	Value
Framerate	30 FPS
Resolution	HD720
Depth Mode	ULTRA
Depth Minimum Distance	-1
Sensing Mode	STANDARD
Reference Frame	CAMERA
Disable Self Calibration	True
Image Flip	False
Enable Right Side Measure	False
Depth Stabilization	True
Compression Mode	LOSSLESS

more, the available documentation for this sensor is not as accurate as it is for other devices, and only partial information from the SDK of the sensor is accessible.

From the four modes available for the HDR, two modes (Simple HDR and Auto HDR) are listed are discouraged. Of the remaining two (Super HDR and HDR Disabled), it is unclear how their parameters change the results. After analyzing the results visible from the producer's Client, the Super HDR has been selected with a frame integration period of 4 frames and two possible maximum integration periods of 5000 µs and 15 000 µs for the first and second phase of the experiment. No Region of Interest (ROI) has been set.

6.2 State of the Laboratory

For each recorded session, the RACOON-Lab axes were kept in the state listed in table 6–7 and were based on the ones used in the previous thesis [15]. Changes have been made to account for the new sensors and the presence of the Earth's albedo. The reached satellite position appears in the center of the recorded frames.

6.3 Recording Procedure

6.3.1 Data Gathering

For this experiment, given the time available and the conditions in the laboratory, two series of recordings were made under different light conditions: in the first, the Earth's albedo light was kept off, while in the second it was turned on. In this way, it was possible to test the D435 under the illumination of the projector and the flickering light of the HPS-3D160 laser scanner IR alone, and then compare the effect of the Earth's albedo as an additional light source. It is worth noting that, in the first case, it is

Tab. 6–7: Values of the axes for the RACOON-Lab

Axis	Value	Conversion factor
0	180 000 steps	85 600 steps/m
1	14 000 steps	85 600 steps/m
2	−200 steps	24 430 steps/m
3	−190 000 steps	226 250 steps/m
4	0 steps	24 430 steps/m
5	−200 steps	24 430 steps/m
6	−63 250 steps to 60 000 steps	24 890 steps/rad
7	*	190 986 steps/rad
8	−5500 steps	144 000 steps/m
9	300 steps	24 430 steps/m
10	77 000 steps	24 430 steps/m
11	N/A	N/A
12	N/A	N/A
13	37 620 steps	159 949 steps/m
14	93.1°	88.944 steps/deg

impossible to record data from the ZED, as this camera does not work within the IR spectrum.

For each series, the satellite was set at a know inclination relative to the recording devices. This angle was set by changing the 7th axis, starting from 0° angle (0 steps) up to 90° (300 000 steps), with steps of 3° (10 000 steps) between each recording.

In the second case, the light has been positioned at 15° from the axis of symmetry of the laboratory. This position, as already tested [15], prevents shadowing from the chaser and the supporting structure on the satellite, removing obstructions on the reflections. Thanks to recent works in the laboratory, it was possible to know the albedo's location as a distance from a reference point along the rail. Similarly, the same measurement system showed its inclination relative to its tangent. Since the position required was determined as an angle, a conversion to steps of the rail was necessary. As the geometry was not known with precision, some significant locations along the rail (visible in fig. 6–4) have been recorded by manual observations. These values are listed in table 6–8. The location and inclination of the light (using geometric modeling) have been consequently set to 37 625.0 steps for the axis 13 and 93.1° for the axis 14.

Like in the previous experiments, the 6th axis is the one chosen for the rotation of the satellite. The angular velocity selected for this axis was 3.0 deg/s and it swiped an angle of 270°, the broader option available that also prevents the C support structure of

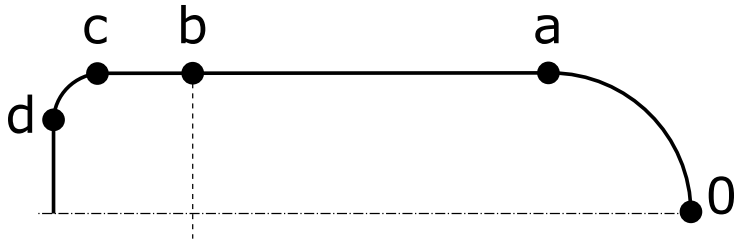


Fig. 6–4: Positions recorded along the axis 13

Tab. 6–8: Calibration of the axis 13 of the RACOON-Lab

Position	Value
0	37 625.0 mm
a	34 493.5 mm
b	30 443.5 mm
c	29 592.5 mm
d	28 360.0 mm

the satellite from standing between the chaser and the target. The selected sequence required 90 s to be completed.

Due to the lack of an available interface between the recorder and the LabView GUI, it was not possible to correctly synchronize the recording with the predicted movement of the target. Furthermore, due to the software of the RACOON-Lab, a non-deterministic time delay was present between the sending of commands and their execution. This delay also increases from the start of the GUI, adding additional uncertainty to the whole chain tool. Additionally, the time between the start of the registration and the actual data recording, even if short, was also not deterministic. The 6th axis' swipe angle has been therefore increased of 3250 steps after examining the average delays to compensate for all these phenomenons. In this way, the first section of data can be lost without the risk of losing information on the 270° angle swipe.

In addition to the already mentioned series, for each taken light condition, ten identical recordings have been taken. These data, required to obtain an indication of the measurement uncertainty of the laboratory, followed the same recording procedure but did not present variation in the condition of the laboratory. The inclination of the satellite was therefore fixed at 45° to be consistent with the previous data available.

6.3.2 Data Processing

After having collected all the recordings, the conversion phase began. Since not all data were necessary for computation or visualization purposes, only some channels have been exported, as visible in table 6–9. This choice does not prevent future conversions of more information, as the *converter.exe* program can append information to the NetCDF4 data structure.

Tab. 6–9: Exported data channels from the devices' sources

Device	Tests	Series 1	Series 2
ZED	VIEW_DEPTH	N/A	VIEW_DEPTH VIEW_CONFIDENCE
D435	DEPTH COLOR	DEPTH LEFT_INFRARED	DEPTH LEFT_INFRARED[1] COLOR
HPS-3D160	N/A	DEPTH	DEPTH

Once the NetCDF4 have been converted, the *root.nc* has been created. This file, as previously described in section 5.1, hosts the main node of the data structure, the algorithms, and the filters. For the path tracking, the information listed in tables 6–10 and 6–11 has been stored on the relative subgroups.

Since the resolutions of the various depth maps were different, the configurations settings of the algorithms had to be changed to fit the various frames. This adjustment was especially necessary for the data of the HPS-3D160 LIDAR, given its initially low spatial resolution compared with the other devices. As is visible in its relative last row on tables 6–10 and 6–11, with the selected parameters, the DIFODO algorithm did not experience further decreases in resolution.

The parameters of box filters have also been set to fit the satellite position on the depth frame. The filters have been adapted to fit the satellite in every recorded condition, ensuring that the enclosing box did not exclude parts of it. In the record where the target occupies the broader number of pixels, the satellite was inclined by $45°$ with its main surfaces parallel or perpendicular to the camera frame.

Each filter group hosts information about the enclosed frame by recording the upper-top and the lower-bottom corners of the selected area (corner_1 and corner_2). The reference system of the coordinate starts in the upper-left corner of the image, as it is possible to visualize in fig. 6–5. As the distance between the target and the origins of the depths map were within a few centimeters apart, the distance range was kept identical for all devices.

Before determining the paths, a final preparatory step had to be performed. Because the starting and stopping of the recording were done sequentially (one device after another), the results did not overlap perfectly. Furthermore, as these commands were given manually, no timestamp information was available to determine in an absolute way the investigated time range. The D435 video results have been consequentially manually scanned, as its SDK tool allows one to see the recorded frames and their relative timestamps comfortably. After the ending instants of the swipe have been

[1] Only for some sessions
[2] Different colors groups the same settings

Tab. 6–10: Algorithms stored in *root.nc*[2]

ID	Field	Value
0	alias	ZedSDK
	confidence_threshold	85
	depth_max_range_value	5
	alias	DIFODO
	skip_frames	0,1,2
	downsample	1
	cols	320
1, 2, 3	rows	180
	ctf_levels	4
	max_depth	5.0
	target_resolution_width	1280
	target_resolution_height	720
	alias	DIFODO
	skip_frames	0,1,2
	downsample	1
	cols	212
4, 5, 6	rows	120
	ctf_levels	4
	max_depth	5.0
	target_resolution_width	848
	target_resolution_height	480
	alias	DIFODO
	skip_frames	0,1,2
	downsample	1
	cols	160
7, 8, 9	rows	60
	ctf_levels	1
	max_depth	5.0
	target_resolution_width	160
	target_resolution_height	60

Tab. 6–11: Filters stored in *root.nc*

ID	Field	Value
0	alias	None
	alias	BoxFilter
	min_distance	0.8
	max_distance	2.0
1	corner_1	[350, 35]
	corner_2	[960, 540]
	target_resolution_width	1280
	target_resolution_height	720
	alias	BoxFilter
	min_distance	0.8
	max_distance	2.0
2	corner_1	[231, 90]
	corner_2	[595, 420]
	target_resolution_width	848
	target_resolution_height	480
	alias	BoxFilter
	min_distance	0.8
	max_distance	2.0
3	corner_1	[40, 0]
	corner_2	[110, 59]
	target_resolution_width	160
	target_resolution_height	60

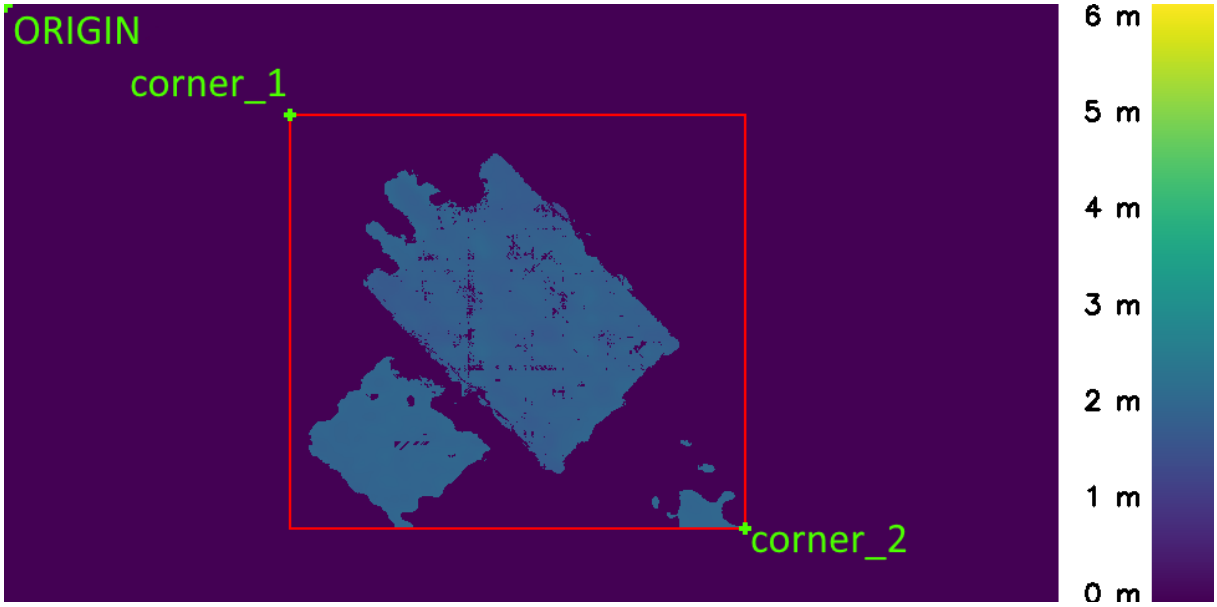


Fig. 6–5: Representation of the BoxFilter's information stored in the root file

individuated, it was possible to calculate the origin timestamp of the recording, to be given to the *path_tracking.exe*'s parameters from_timestamp and to_timestamp. The measured timestamps have been listed to table B.4.

At the end of the processing phase, a compression procedure has been enabled for decreasing the memory usage and data bandwidth required for transmission [45]. As it uses the NetCDF4's native implementation, the data handling operations do not require any knowledge of it and only experience a slowdown in the reading performances. Following the official recommendation, the datasets have been compressed using the zlib library with a deflation level of 4 and shuffling enabled. The results determine a decrease in memory footprint of approximately $56\,\%$ from the original size. The specific files show different levels of compression, as these depend on the format of the channels saved on each file.

7 Results

7.1 Reference Cases

As said before, ten reference recordings with identical conditions for each case have been recorded to be able to measure the uncertainty of the whole setup. The data gathered have then been supplied to the path tracking program for all the expected combinations, and, from the resulting paths, the metric parameters have been calculated. Their standard deviations, discriminated by skipped frames, sensors, filter algorithm, and albedo power/integration time, are collected in table 7–1, while a visual representation is visible in fig. 7–1. The table presents the results of the DIFODO algorithm, but it also hosts the results of the references for the Zed SDK algorithm (enclosed by parentheses), which have been inserted for completeness.

Tab. 7–1: Standard deviation of the metric parameters for the reference cases. The results of the ZedSDK algorithm are also visible.

Skipped	Sensor	Filter	Albedo [%]	C_F	r_F	ε	R	δ_max
0	D435	BoxFilter	0.0	0.0775 m	0.0718 m	1.3318°	0.0083 m	13.6137°
			100.0	0.1187 m	0.1055 m	4.9250°	0.0181 m	13.9979°
		None	0.0	0.1489 m	0.1959 m	2.9527°	0.0282 m	19.7020°
			100.0	0.4613 m	0.5440 m	7.2850°	0.0505 m	37.1486°
	HPS-160	BoxFilter	0.0	0.1100 m	0.1178 m	25.1101°	0.0095 m	9.3899°
			100.0	0.0412 m	0.0650 m	8.2424°	0.0241 m	15.4607°
		None	0.0	0.0886 m	0.1066 m	28.0068°	0.0041 m	10.6440°
			100.0	0.0199 m	0.0245 m	8.1694°	0.0073 m	40.2888°
	ZED	BoxFilter	100.0	0.0144 m	0.0144 m	77.3624°	0.0190 m	1.9536°
		None	100.0	0.0228 m	0.0257 m	1.6280°	0.0031 m	1.9326°
		(ZedSDK)	100.0	0.0057 m	0.0028 m	0.1432°	0.0011 m	1.2038°
1	D435	BoxFilter	0.0	0.0865 m	0.0700 m	0.4961°	0.0128 m	17.6754°
			100.0	0.0430 m	0.0451 m	0.7158°	0.0023 m	8.1359°
		None	0.0	0.1251 m	0.1610 m	0.8975°	0.0348 m	37.8012°
			100.0	0.2290 m	0.2686 m	1.7184°	0.0274 m	46.3288°
	HPS-160	BoxFilter	0.0	0.1043 m	0.1136 m	24.7693°	0.0046 m	7.2266°
			100.0	0.0654 m	0.0674 m	6.4845°	0.0166 m	9.6745°

Continued on next page

Tab. 7–1: Standard deviation of the metric parameters for the reference cases. The results of the ZedSDK algorithm are also visible.

Skipped	Sensor	Filter	Albedo [%]	C_{F}	r_{F}	ε	R	$\delta_{\max}$
		None	0.0	0.0350 m	0.0396 m	17.4561°	0.0011 m	4.0611°
			100.0	0.0145 m	0.0062 m	3.3891°	0.0036 m	36.4136°
	ZED	BoxFilter	100.0	0.0100 m	0.0088 m	81.0600°	0.0175 m	2.0535°
		None	100.0	0.0402 m	0.0494 m	66.5013°	0.0070 m	1.5127°
2	D435	BoxFilter	0.0	0.0238 m	0.0243 m	0.1185°	0.0054 m	9.9783°
			100.0	0.0380 m	0.0453 m	0.2409°	0.0103 m	11.6051°
		None	0.0	0.1190 m	0.1278 m	0.5639°	0.0137 m	16.2792°
			100.0	0.1370 m	0.1671 m	0.8130°	0.0204 m	42.2325°
	HPS-160	BoxFilter	0.0	0.0894 m	0.0900 m	18.8588°	0.0059 m	6.4491°
			100.0	0.1275 m	0.1236 m	11.1164°	0.0195 m	23.5877°
		None	0.0	0.0297 m	0.0378 m	4.1665°	0.0014 m	4.7491°
			100.0	0.0204 m	0.0228 m	7.2843°	0.0065 m	19.4313°
	ZED	BoxFilter	100.0	0.0088 m	0.0077 m	83.4556°	0.0156 m	2.4305°
		None	100.0	0.0435 m	0.0459 m	65.4879°	0.0042 m	1.9071°

The rationale for preferring to discriminate the standard deviations also based on the skipped frames lives in the presence of the dropped frames. As already described while analyzing the framerate of the various devices, unlike previous works with a stable framerate, here, a statistically noticeable delay in the duration of frames is present, as well as a significant appearance of dropped frames. As their effect on the final path is unknown, it has been regarded necessary to differentiate these results, and further analysis will be done in the following sections while investigating the dark and albedo cases.

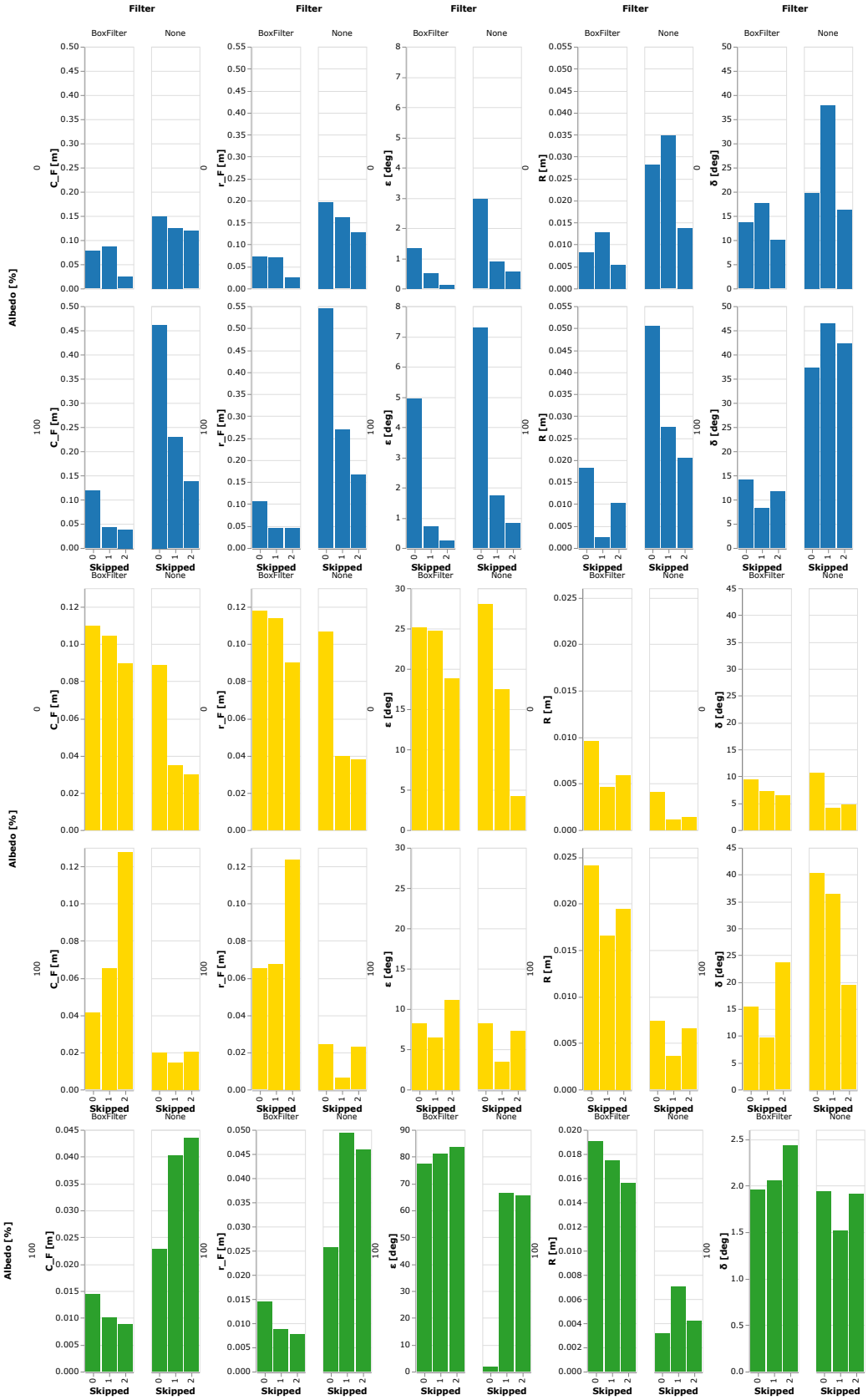


Fig. 7–1: Standard deviation of the DIFODO metric parameters for the reference cases. The D435 is represented in blue, in gold the HPS-3D160 and in green the ZED

7.2 Dark Case

Will now be presented the results for the tests conducted without the presence of the albedo, where only D435 (thanks to its IR projector) and the HPS-3D160 were able to detect depth information. The summary of all the results is presented in table 7–2, while a more particular examination, where it is possible to see the change of the metric parameters as the satellite changes its inclination, is visible in figs. 7–2 to 7–6. These plots also show the calculated fitting lines related to the inclination angles, and will later be analyzed. As the ZED did not partake in this phase, all the following results are relative to the DIFODO algorithm.

Tab. 7–2: Average metric parameters for all analyzed combinations without albedo

Skipped	Sensor	Filter	C_{F}	r_{F}	ε	R	$\delta_{\max}$
0	D435	BoxFilter	0.7096 m	3.3914 m	99.7636°	0.0670 m	230.8019°
		None	1.4716 m	2.7792 m	86.6293°	0.1299 m	165.8778°
	HPS-160	BoxFilter	1.0148 m	3.0234 m	98.8381°	0.0515 m	182.0101°
		None	0.5443 m	3.5020 m	104.0684°	0.0272 m	165.3665°
1	D435	BoxFilter	0.4590 m	3.6004 m	91.4012°	0.0346 m	175.8577°
		None	0.7409 m	3.3690 m	89.3439°	0.0581 m	169.4992°
	HPS-160	BoxFilter	1.0208 m	3.0015 m	91.5741°	0.0480 m	181.1260°
		None	0.5369 m	3.4898 m	105.0408°	0.0206 m	174.8283°
2	D435	BoxFilter	0.4245 m	3.6187 m	90.6920°	0.0253 m	173.9592°
		None	0.5157 m	3.5520 m	90.0648°	0.0348 m	171.2504°
	HPS-160	BoxFilter	1.0903 m	2.9303 m	82.0506°	0.0438 m	184.1009°
		None	0.5112 m	3.5091 m	108.9755°	0.0185 m	175.9724°

It is possible to observe, in the scatter plots visible in fig. 7–2, the relationship between the C_{F} parameter as the satellite changes its inclination for the various devices. These plots also show, along the first column, how the results changed when the BoxFilter was applied, while the second column illustrates what happened when no filter was employed. Similarly, the facet matrix rows show, for the case with and without filter, the effects of skipping frames of the recording, simulating an increase in the angular velocity of the satellite.

To each device's dataset, a least-squares linear regression has been calculated (as visible in the plot) to identify a general trend in the results. The lines coefficients (as well as the results of the fitting process) are visible in table B.5.

Similarly, in fig. 7–7 are visible the average of the progressions of the drift of the paths under the various computed combinations. Since no position was in precise temporal

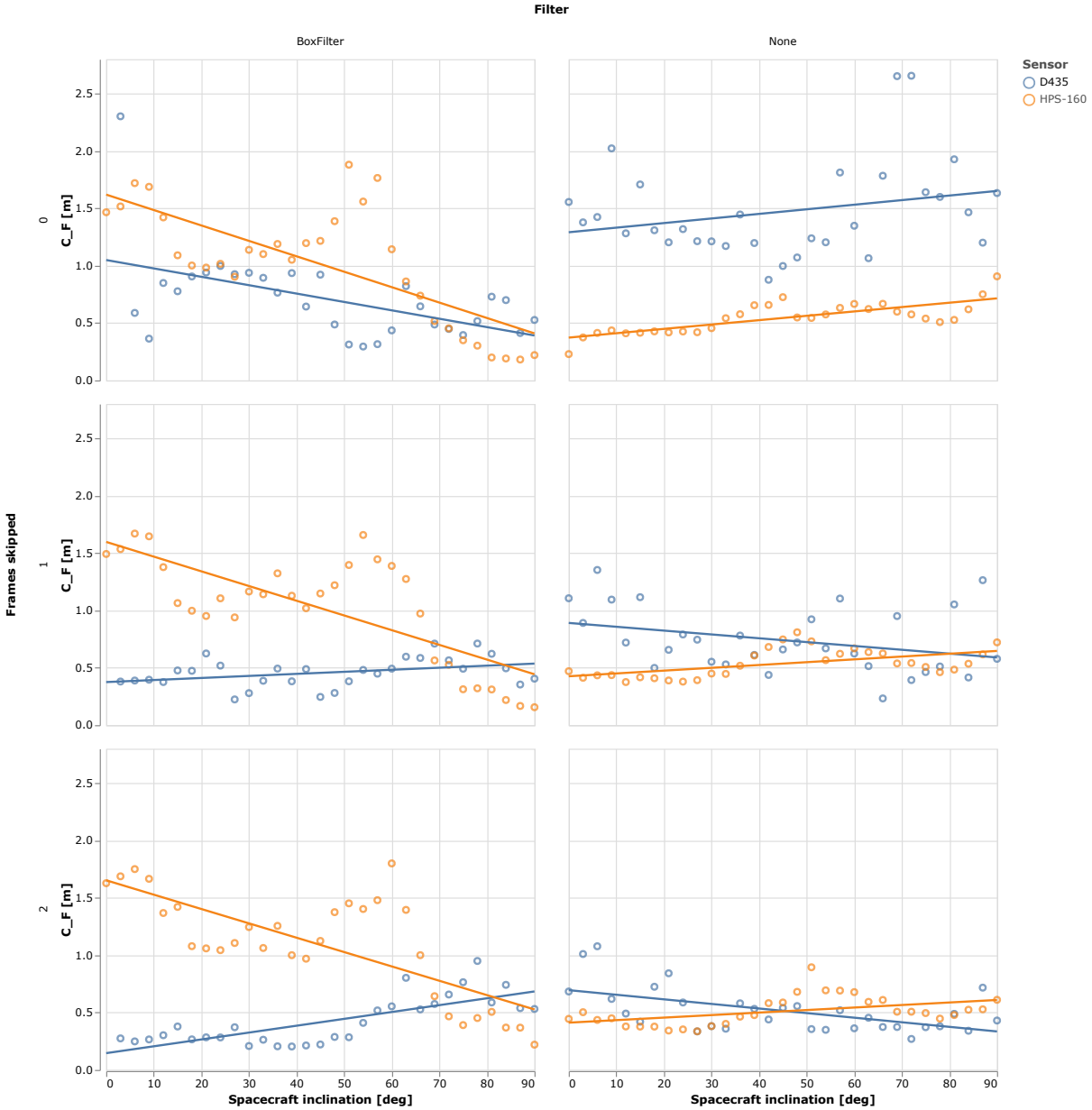


Fig. 7–2: All computed values of C_F in the dark case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.5

alignment with the others, linearly interpolated positions for every experiment have been found, calculated with steps of one second.

In these plots, it is possible to see the presence of discontinuities. As the averages give no insight about their meaning or their origin, it is convenient to analyze the single drifts. Given the high quantity of lines to be shown, to gather more efficiently the information and have a better overall insight, the interpolated points are displayed in the heatmaps in fig. 7–8, where it is also possible to discriminate how the inclination of the spacecraft affects the drift of the calculated paths. As visible in the swift changes of color, it is possible to identify a change of sign in some areas. These rapid changes are a known

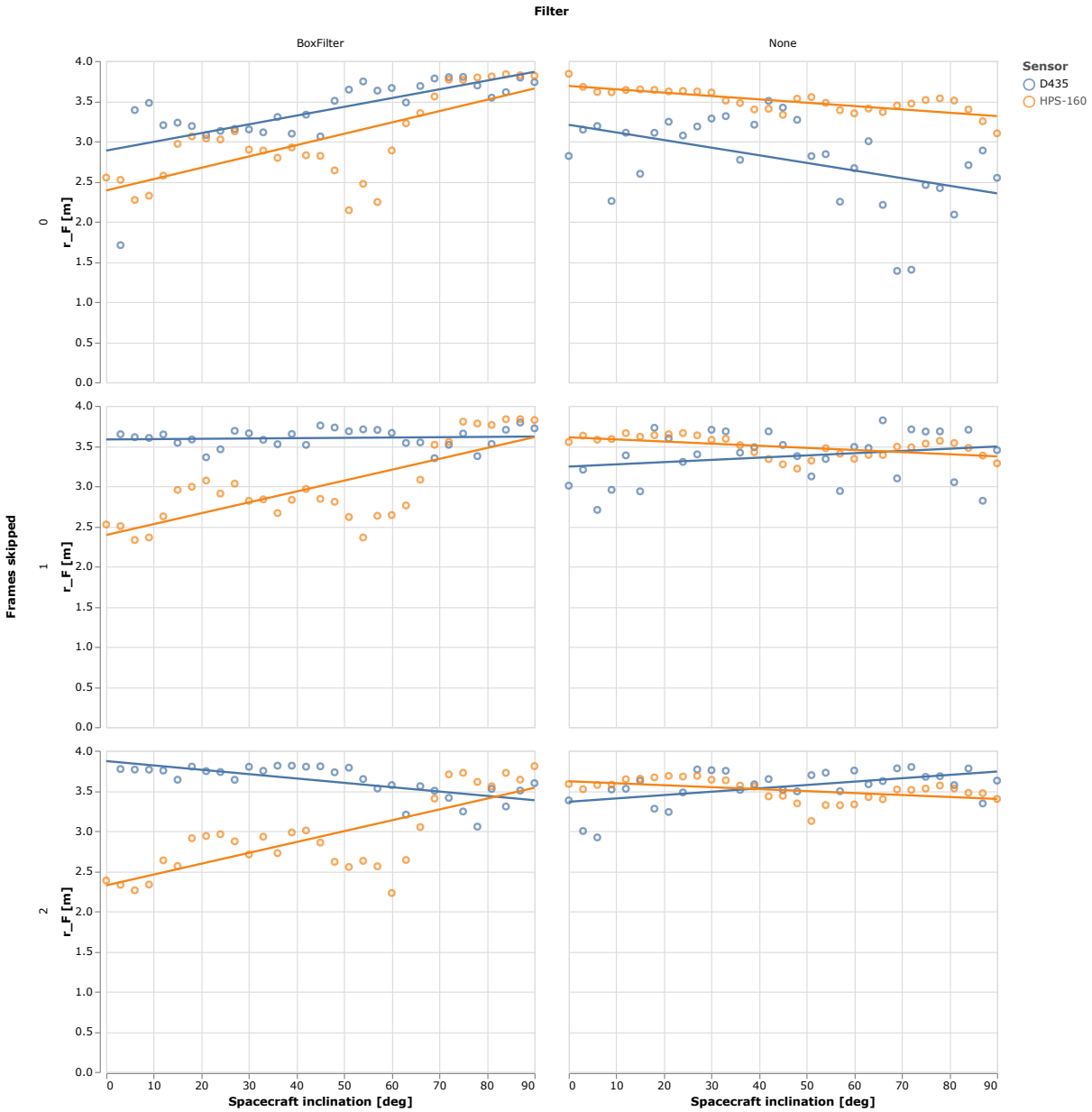


Fig. 7–3: All computed values of r_F in the dark case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.5

problem of the current implementation of the metric and are induced by trigonometric reasons.

From the thermal maps, it is possible to see that some blank lines are present instead of color, due to problems in the metric's implementation. For these paths, the called function throws an exception and does not return the metric parameters.

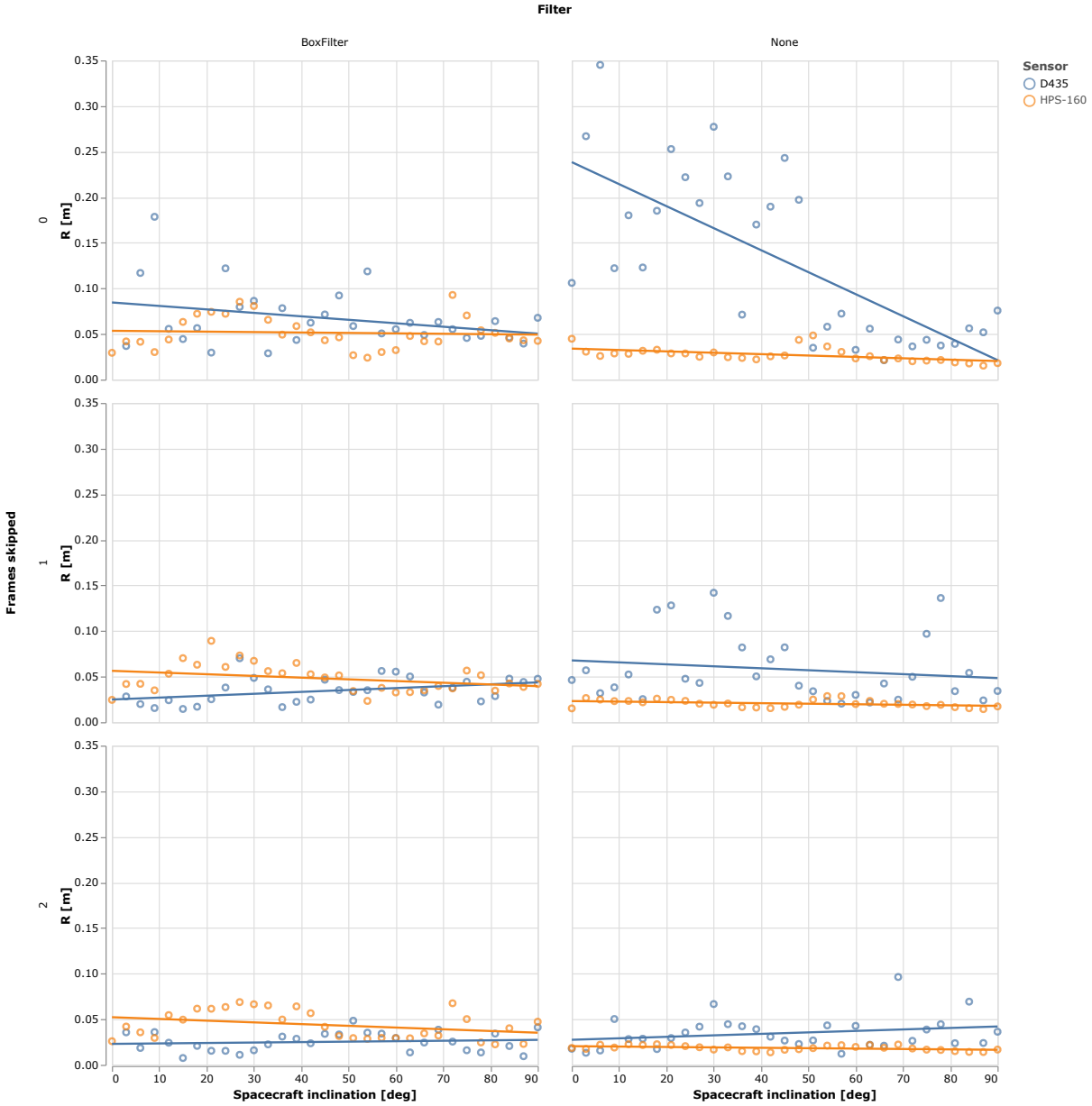


Fig. 7–4: All computed values of R in the dark case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.5

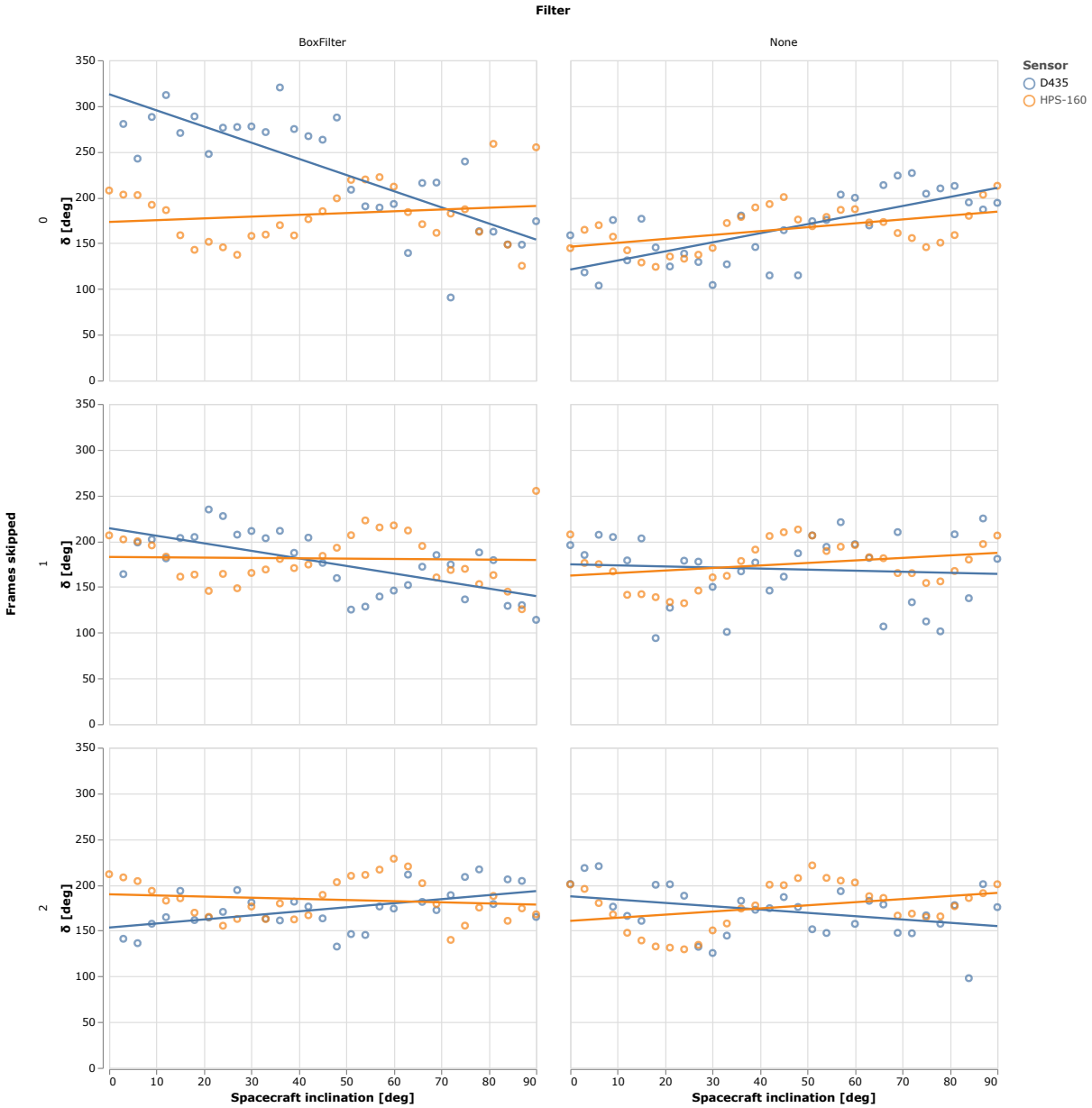


Fig. 7–5: All computed values of $\delta_{\max}$ in the dark case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.5

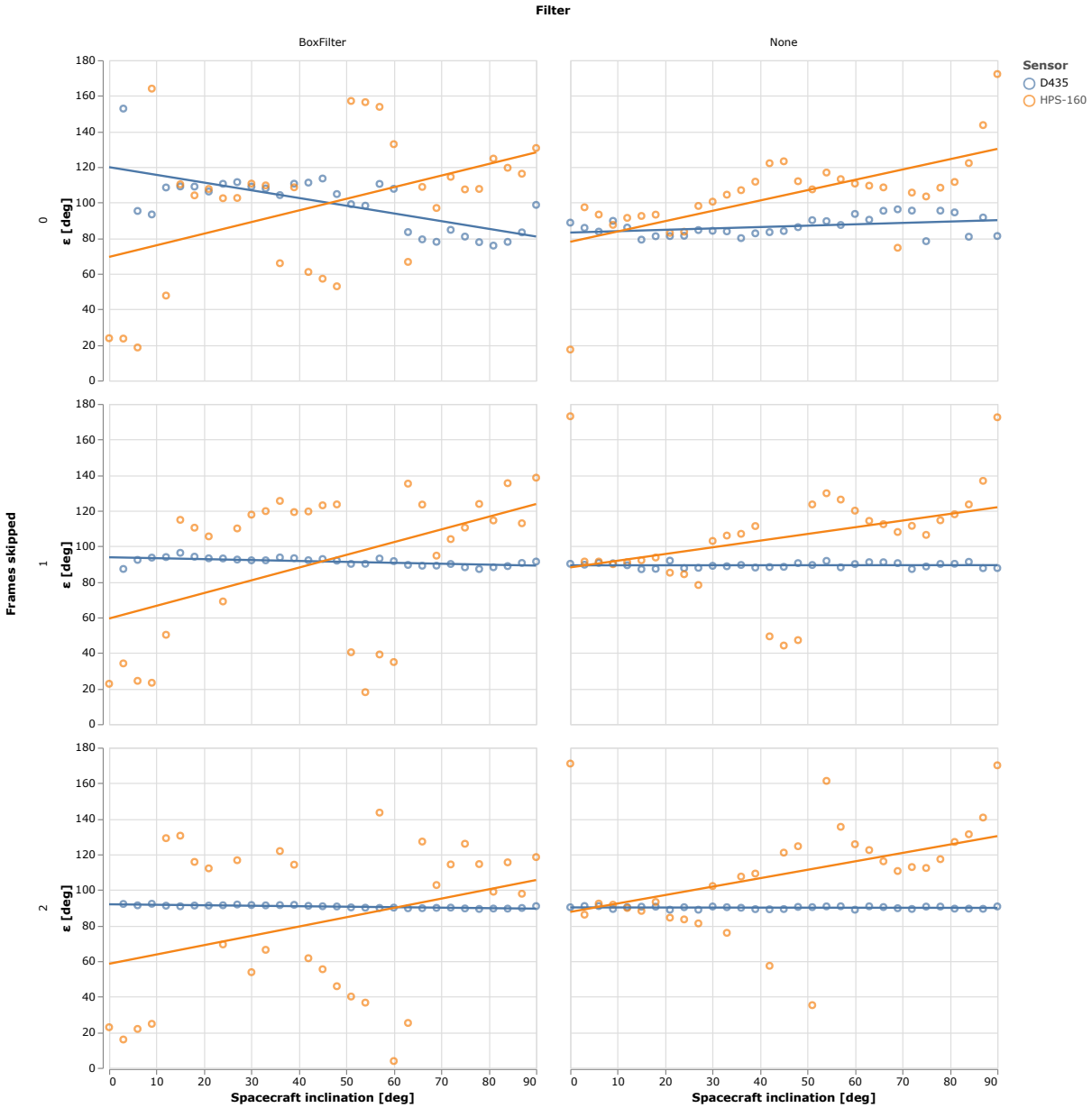


Fig. 7–6: All computed values of ε in the dark case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression line for each sensor, and their parameters are listed in table B.5

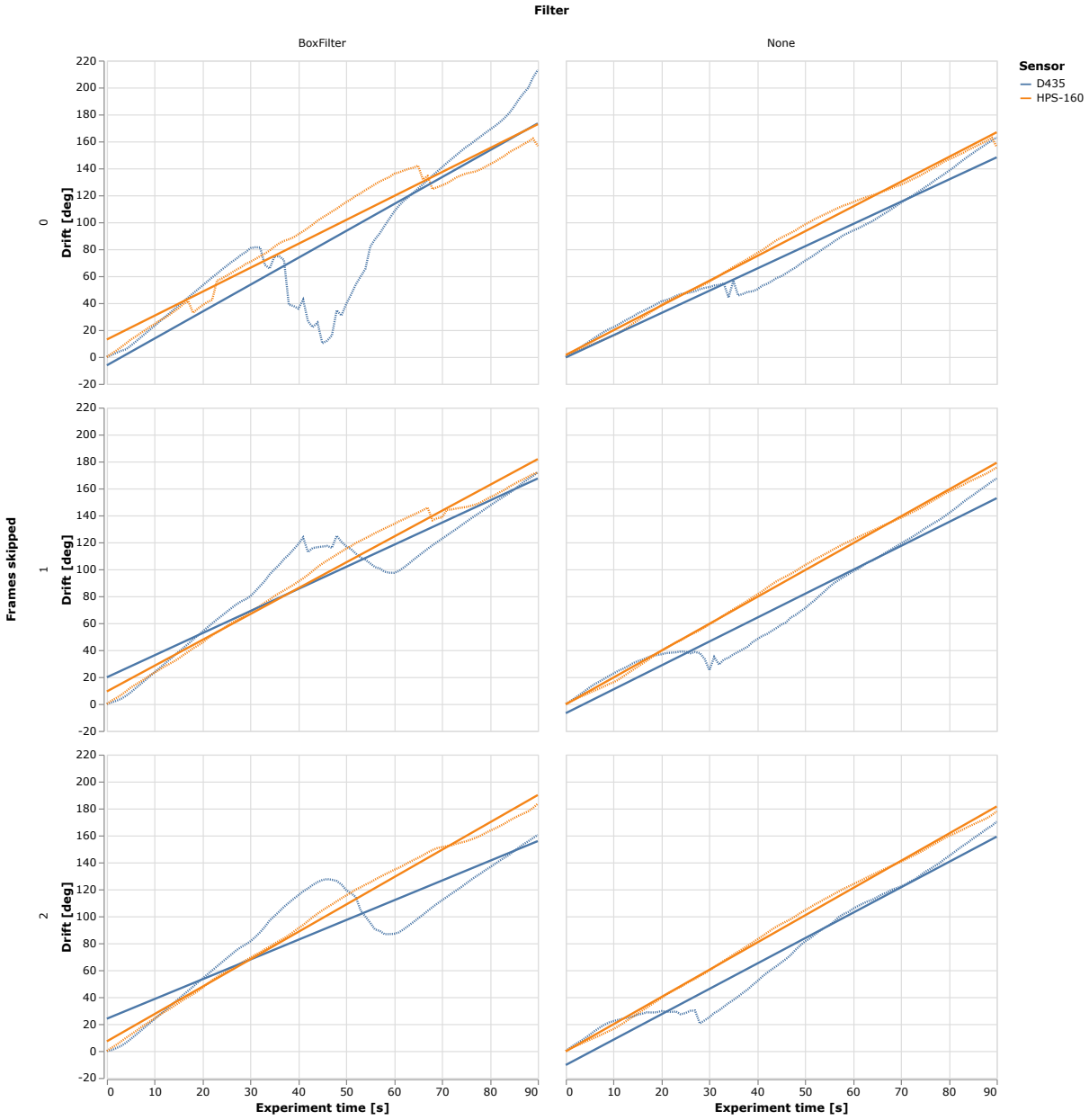
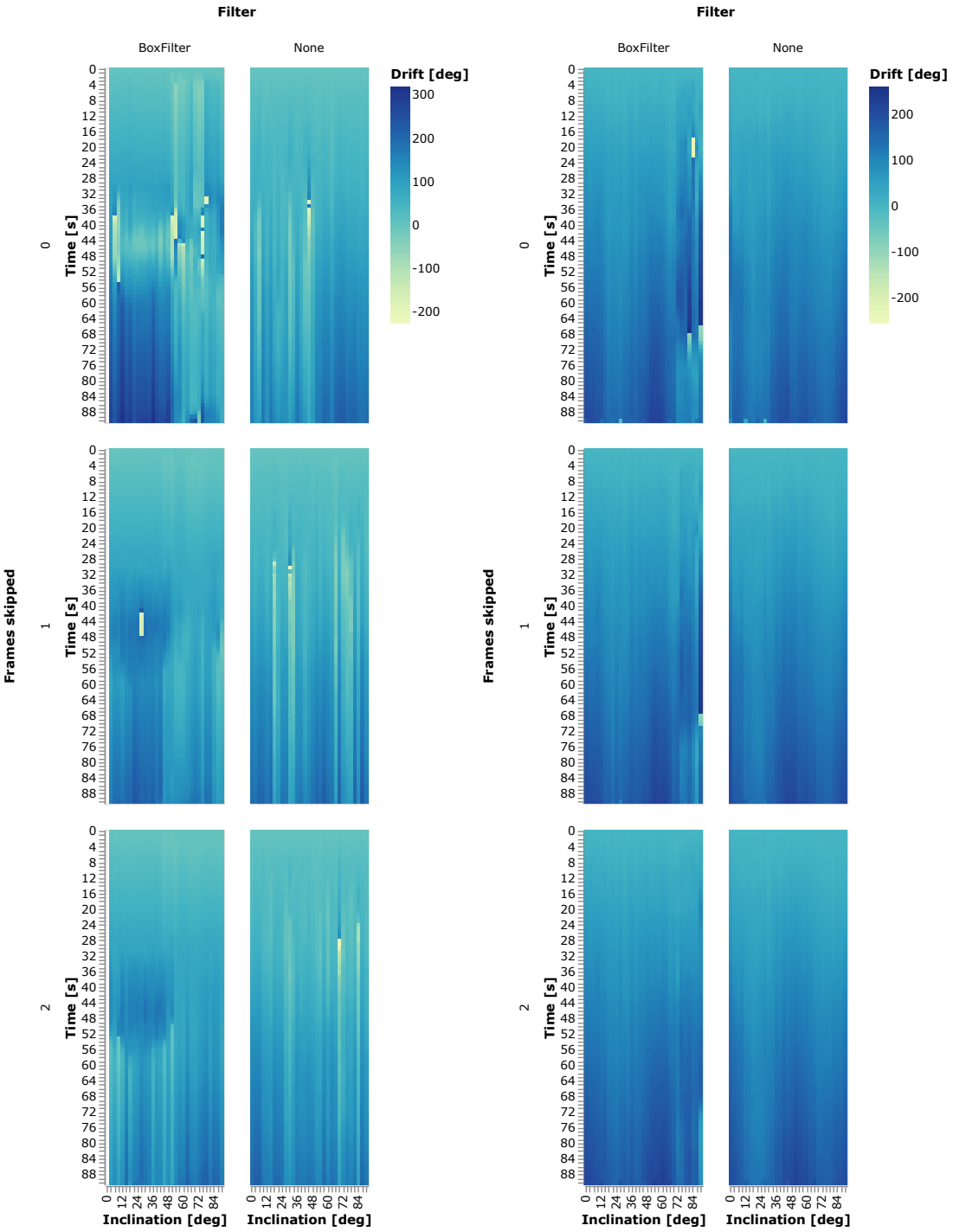


Fig. 7–7: Average of the progressions of the drift of the paths under the various computed combinations in the dark case, diplayed with the dotted line, while the solid lines show their relative regression line. The lines' parameters are listed in table B.7



(a) D435 and DIFODO

(b) HPS-3D160 and DIFODO

Fig. 7–8: Evolution of all the drifts for the dark case. In each subplot, int the x axis is visible the inclination of the satellite, while on the y axis is visible the time since the beginning of the recording session

7.3 Albedo Case

Will now be presented the results for the tests conducted with the presence of the albedo, where, together with the D435 and the HPS-3D160, the ZED camera was also enabled, as the presence of external light allows its depth algorithm to extract depth maps. Like before, a summary of all the results of the paths calculated with the DIFODO algorithm is presented in table 7–3, and, as the ZED SDK also provides the ability to use its internal algorithm to compute the path for the combination of no filter and no skipped frames, a table to compare its result with the DIFODO one under the same conditions is visible in table 7–4.

Tab. 7–3: Average metric parameters for all analyzed combinations with albedo for the DIFODO algorithm

Skipped	Sensor	Filter	C_F	r_F	ε	R	δ_max
0	D435	**BoxFilter**	0.5780 m	3.4468 m	93.9973°	0.0538 m	213.1698°
		None	1.0600 m	3.1112 m	89.2893°	0.0905 m	163.9630°
	HPS-160	**BoxFilter**	0.4202 m	3.6037 m	94.5954°	0.0505 m	155.4373°
		None	0.2552 m	3.8111 m	97.1567°	0.0282 m	134.1684°
	ZED	**BoxFilter**	2.0826 m	1.9735 m	96.8710°	0.1693 m	51.7446°
		None	1.8181 m	2.1910 m	47.9515°	0.0522 m	102.5641°
1	D435	**BoxFilter**	0.4751 m	3.5481 m	90.9584°	0.0229 m	177.9529°
		None	0.6570 m	3.4204 m	90.1248°	0.0406 m	176.8056°
	HPS-160	**BoxFilter**	0.4553 m	3.5632 m	90.4380°	0.0436 m	166.1996°
		None	0.2499 m	3.8089 m	93.4956°	0.0249 m	138.7340°
	ZED	**BoxFilter**	2.1163 m	1.9345 m	88.1081°	0.1247 m	36.2304°
		None	1.8361 m	2.1699 m	25.9397°	0.0556 m	115.4152°
2	D435	**BoxFilter**	0.6097 m	3.4071 m	90.5002°	0.0227 m	189.0922°
		None	0.5226 m	3.5419 m	90.2580°	0.0349 m	171.3274°
	HPS-160	**BoxFilter**	0.5321 m	3.4905 m	95.5115°	0.0374 m	167.2513°
		None	0.2528 m	3.8042 m	101.2463°	0.0229 m	144.2811°
	ZED	**BoxFilter**	2.1399 m	1.9009 m	98.2725°	0.0913 m	33.9560°
		None	1.8170 m	2.1969 m	16.1136°	0.0551 m	121.8735°

A more specialized examination of the change of the metric parameters as the satellite changes its inclination it is visible, in a similar manner employed for the dark case, in figs. 7–9 to 7–13, with the addition of the data points for the ZED. As two path-tracking algorithms were employed to identify the trajectory, both have been drawn in the scatter results, with the employed algorithm listed in the legend. The calculated least-squares

Tab. 7–4: Metric parameters for ZED's algorithms

Algorithm	C_{F}	r_{F}	ε	R	$\delta_{\max}$
DIFODO	1.9504	2.1823	1.2638	0.1108	77.1544°
ZedSDK	2.0615	2.0259	0.1088	0.0195	20.5339°

linear regression fitting lines for each device are also present, with their parameters listed in table B.6.

In fig. 7–14 are visible the average of the progressions of the drift of the paths under the various computed combinations. The points of plots have been extracted by linear interpolation of the single drifts, which are visible in the heatmaps of figs. 7–15 and 7–16. It is worth pointing out that the span of color bars is scaled to comprise the value of the specific combination of sensors and algorithms, and the same color for different combinations does not imply that the drifts have an equivalent value.

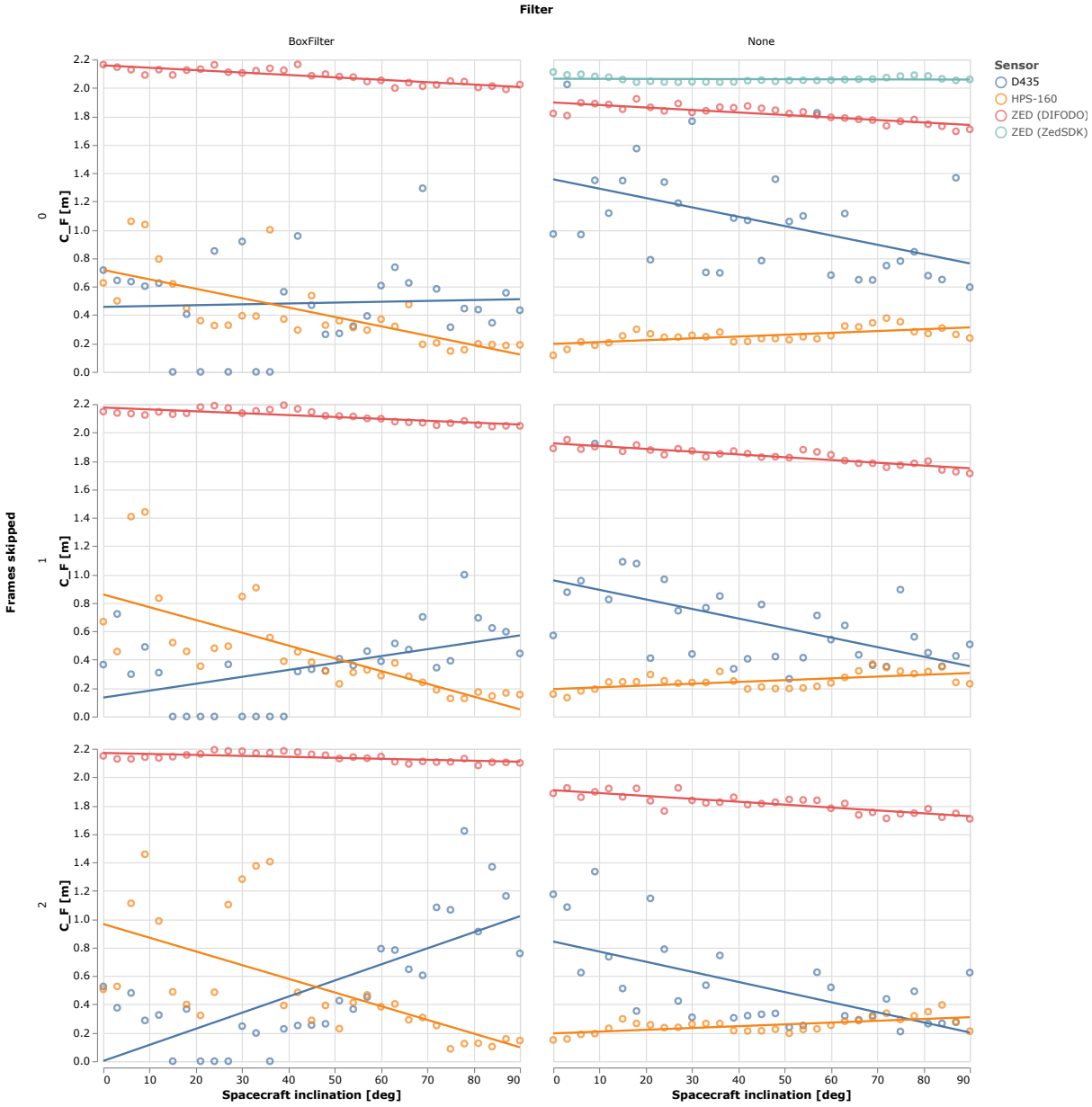


Fig. 7–9: All computed values of C_F in the albedo case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression linse for each sensor, and their parameters are listed in table B.6

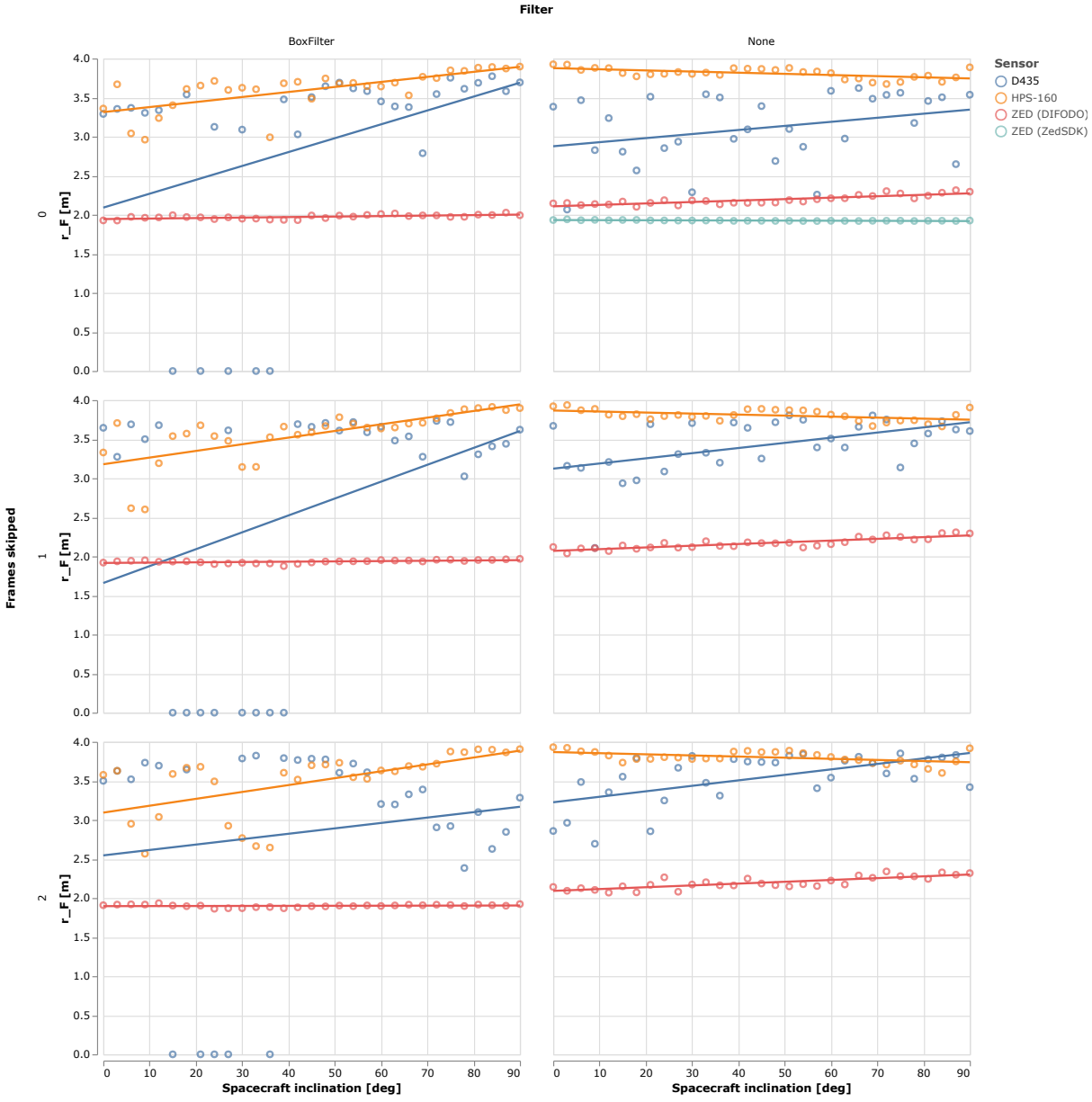


Fig. 7–10: All computed values of r_F in the albedo case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.6

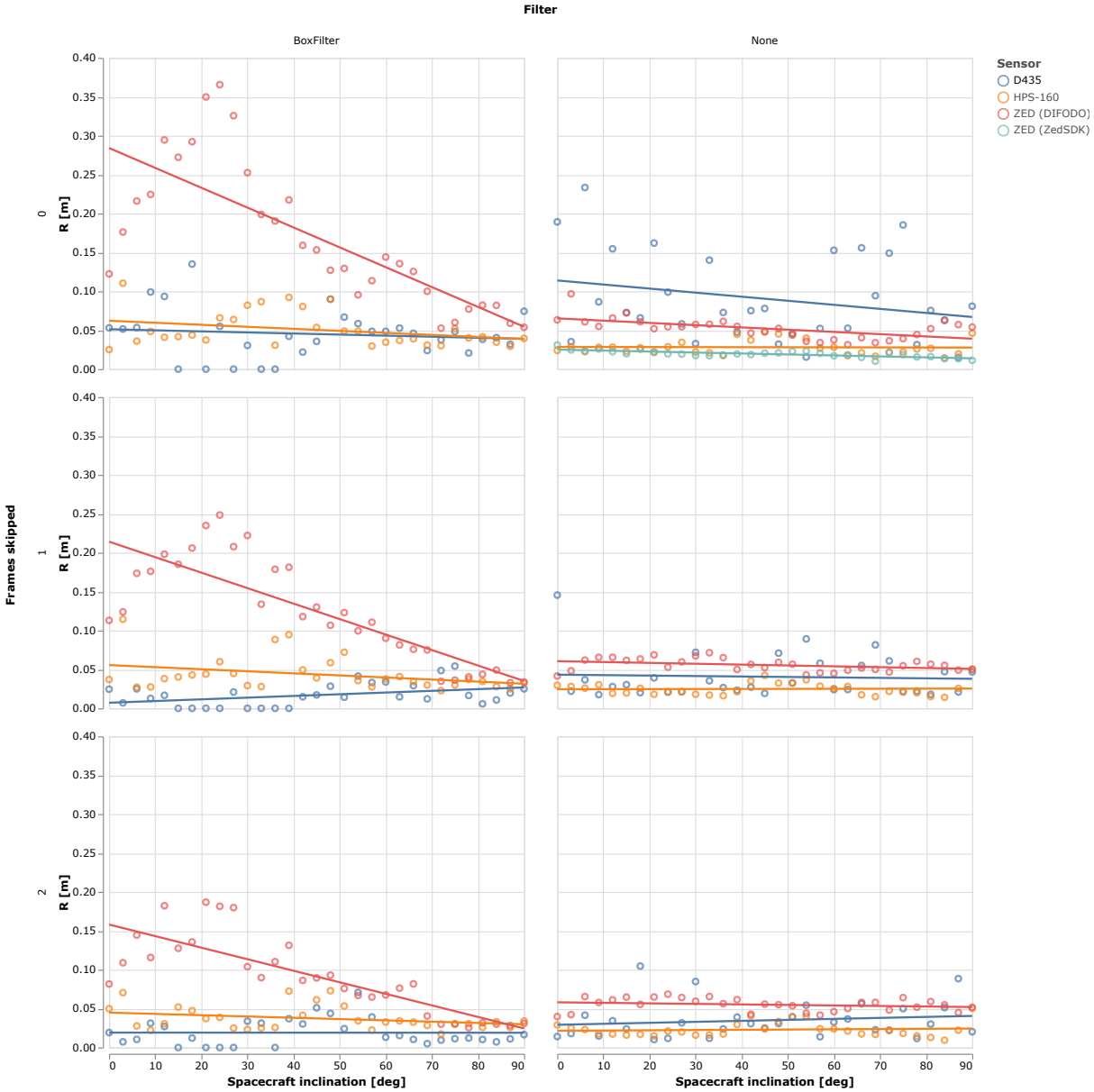


Fig. 7–11: All computed values of R in the albedo case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.6

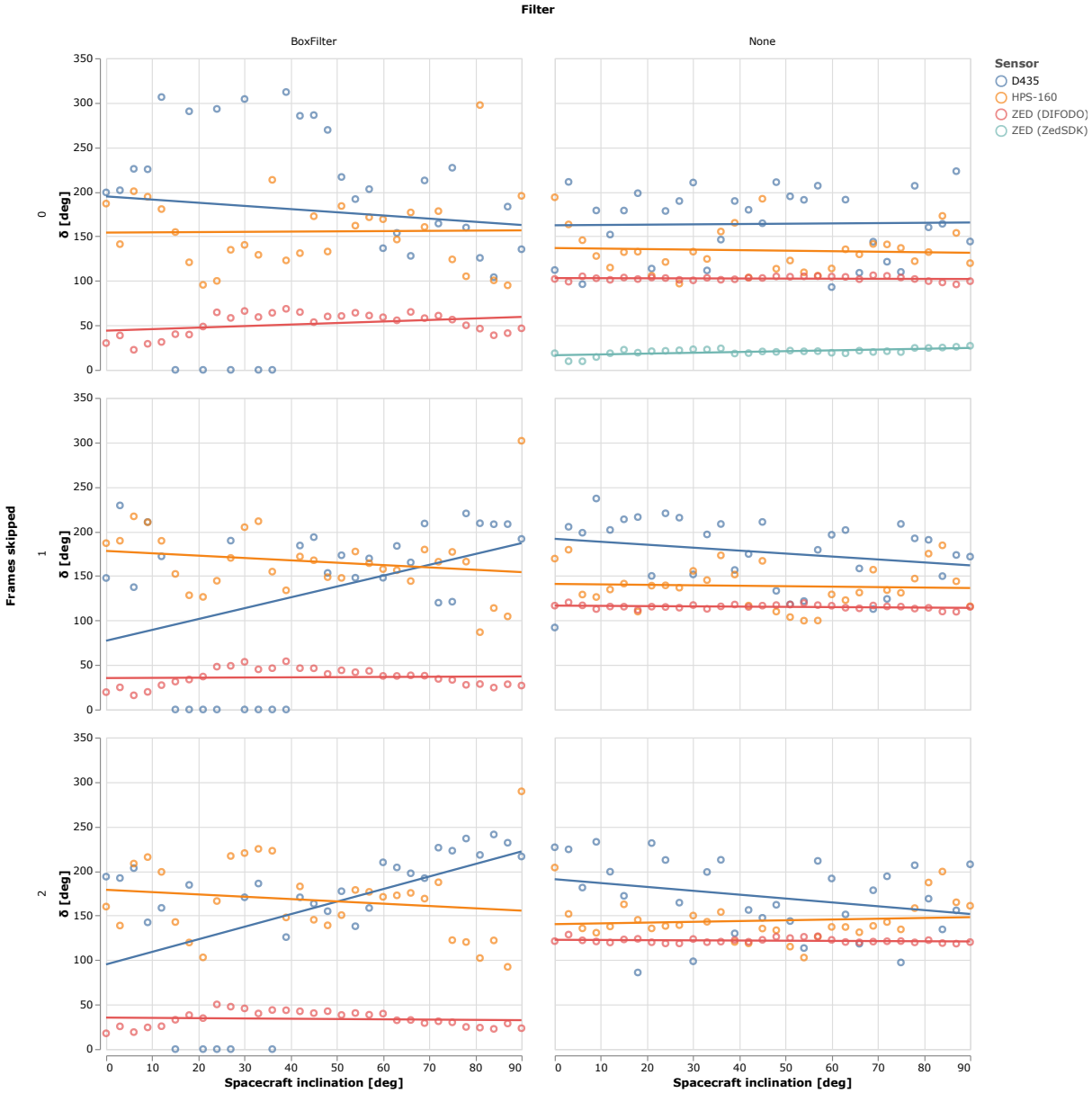


Fig. 7–12: All computed values of $\delta_{\max}$ in the albedo case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.6

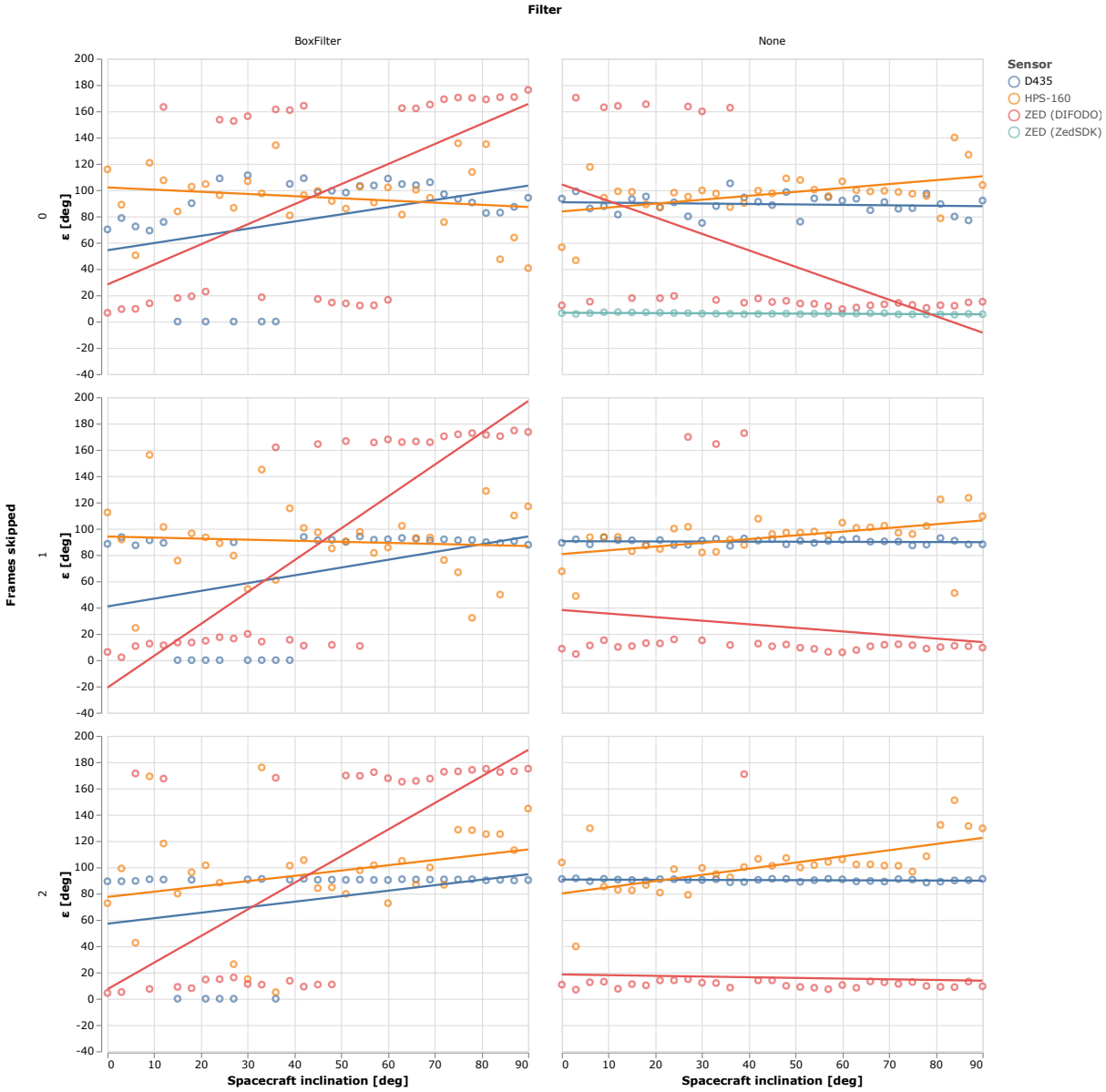


Fig. 7–13: All computed values of ε in the albedo case for all analyzed combinations of inclinations, filters and algorithms. It is also possible to see the regression lines for each sensor, and their parameters are listed in table B.6

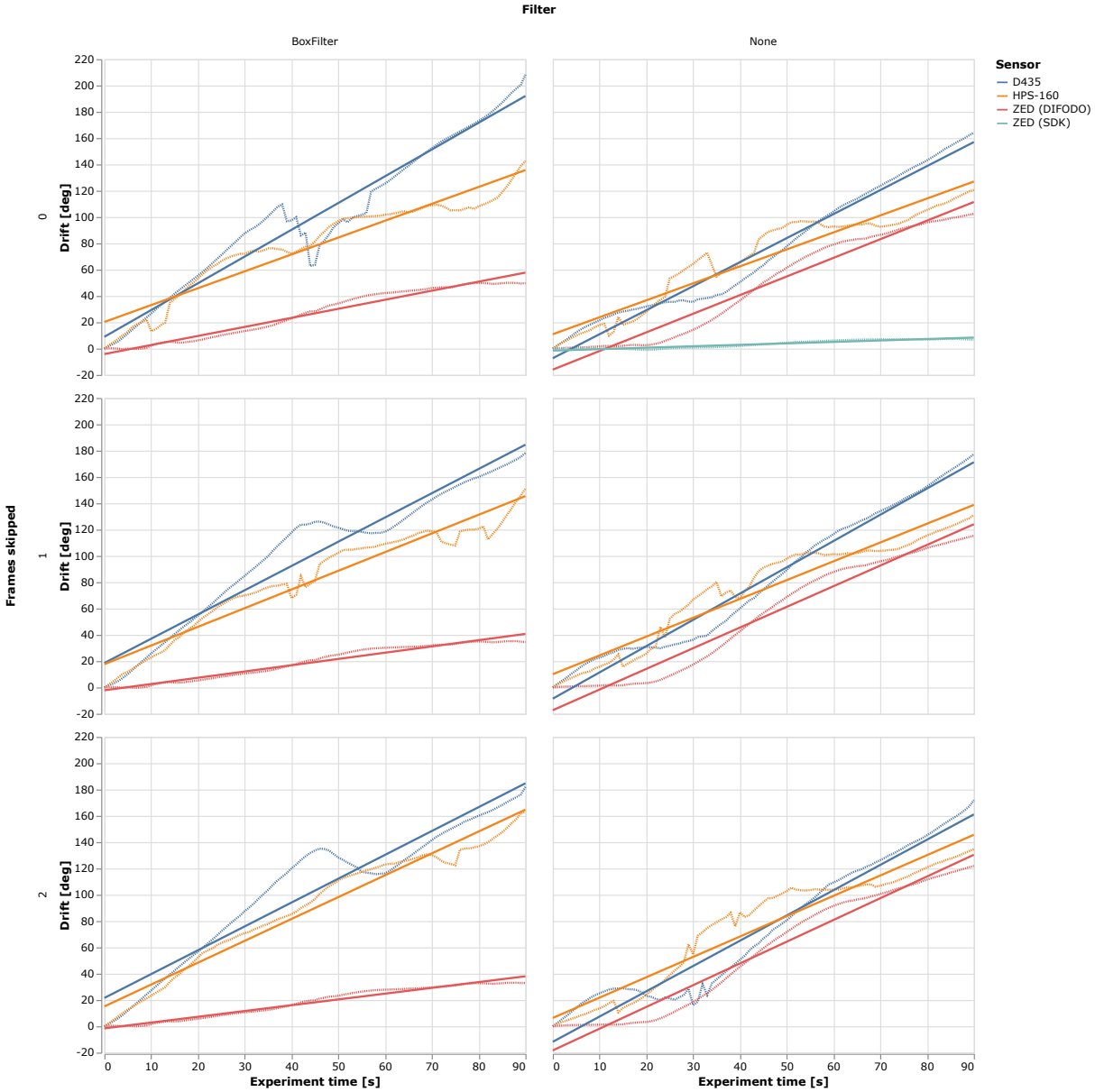
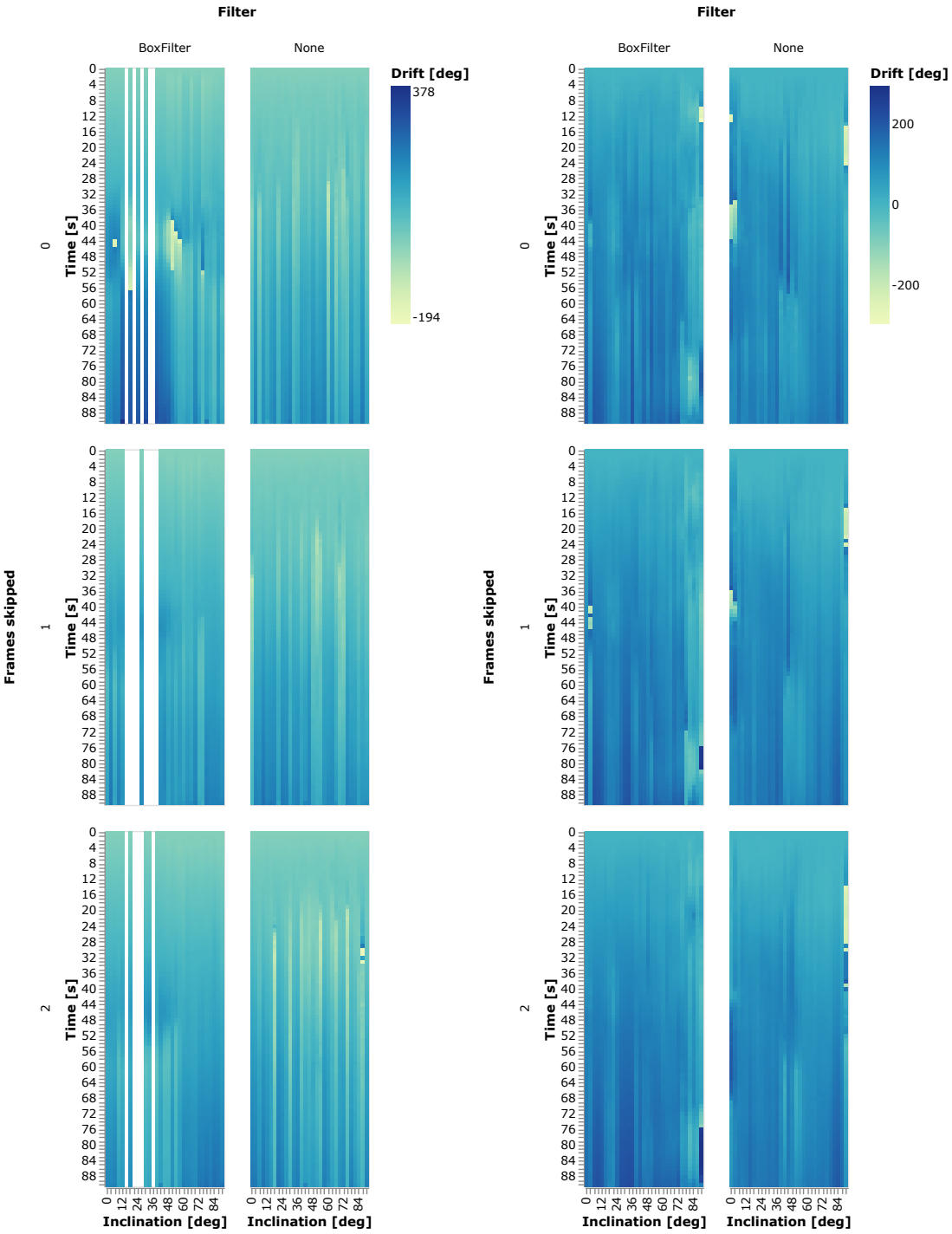


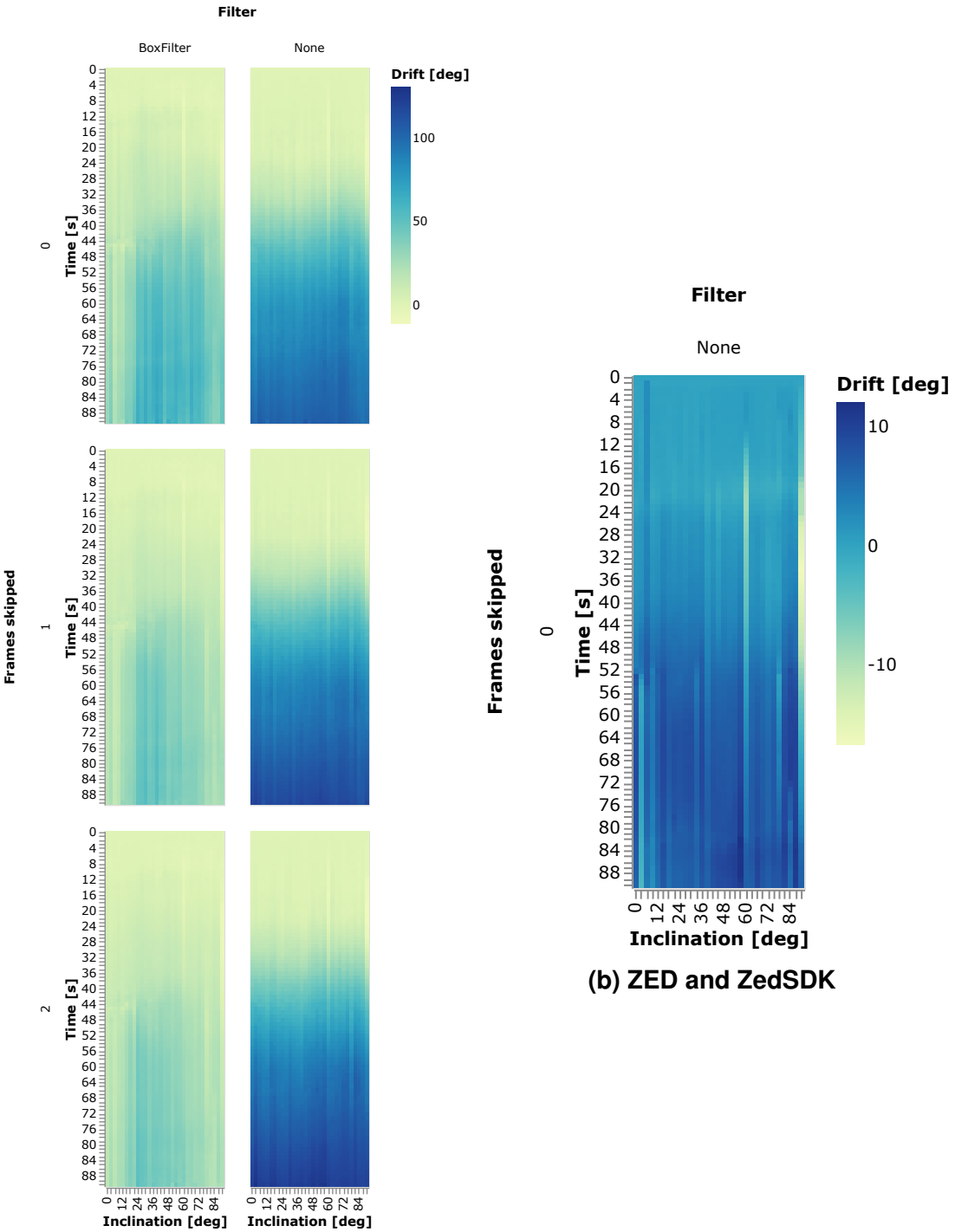
Fig. 7–14: Average of the progressions of the drift of the paths under the various computed combinations in the albedo case, diplayed with the dotted line, while the solid lines show their relative regression line. The lines' parameters are listed in table B.8



(a) D435 and DIFODO

(b) HPS-3D160 and DIFODO

Fig. 7–15: Evolution of all the drifts for the D435 and the HPS-3D160 for the albedo case. In each subplot, int the x axis is visible the inclination of the satellite, while on the y axis is visible the time since the beginning of the recording session



(a) ZED and DIFODO

(b) ZED and ZedSDK

Fig. 7–16: Evolution of all the drifts for the ZED for the albedo case. In each subplot, int the x axis is visible the inclination of the satellite, while on the y axis is visible the time since the beginning of the recording session

7.4 Framerate of the devices

In addition to the previously presented data regarding the performance of the path tracking capabilities of the analyzed devices (as is one of the main objectives of this work), here information regarding their framerate is shown. The reason for also displaying this data (and later analyzing it) is given by the fact that the path tracking algorithm, as it is currently implemented, assumes a constant framerate and does not handle in a sophisticated way dropped frames. As visible in the summary of the duration of the frames and the drop count in table 7–5, these considerations do not hold up properly. An analysis of these time aspects can, therefore, be helpful to understand the result of this study better, as well as to integrate these aspects in future works. As previously stated, it is essential to remember that a frame is considered dropped if the device provides it a non-consecutive number, not if its duration is larger than the period requested for its expected framerate.

Tab. 7–5: Time information about the framerate of the devices

			Frame Period	Dropped Frames
Sensor	**Channel**	**Albedo [%]**		
D435	**COLOR**	**100.0**	0.0602	29.5161
	DEPTH	**0.0**	0.0334	0.0000
		100.0	0.0339	922.3226
HPS-160	**DEPTH**	**0.0**	0.1579	0.0000
		100.0	0.1130	0.0000
ZED	**VIEW_CONFIDENCE**	**100.0**	0.0388	0.0000
	VIEW_DEPTH	**100.0**	0.0388	0.0000

To further display the unsteadiness of the results, in fig. 7–17, it is possible to see where, in the period of the analyzed data, the D435 dropped frames. It is worth pointing out that the frame number axis is split in 60-frames bins.

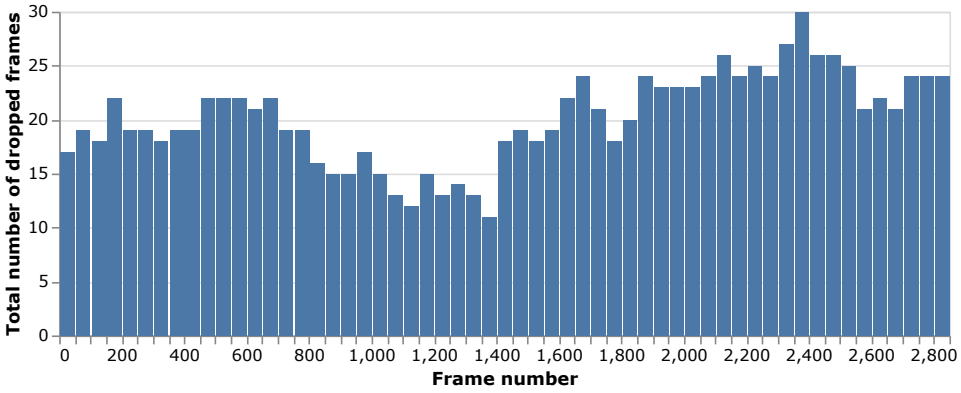


Fig. 7–17: Dropped frames distribution over time

Also, it is presented in fig. 7–18, an example of the devices' framerate evolution, showed to illustrate the unsteadiness of the frame rate during the recording process. The data here displayed have been gathered from a casually selected session, namely session 20200221T132658.259450500+0000.

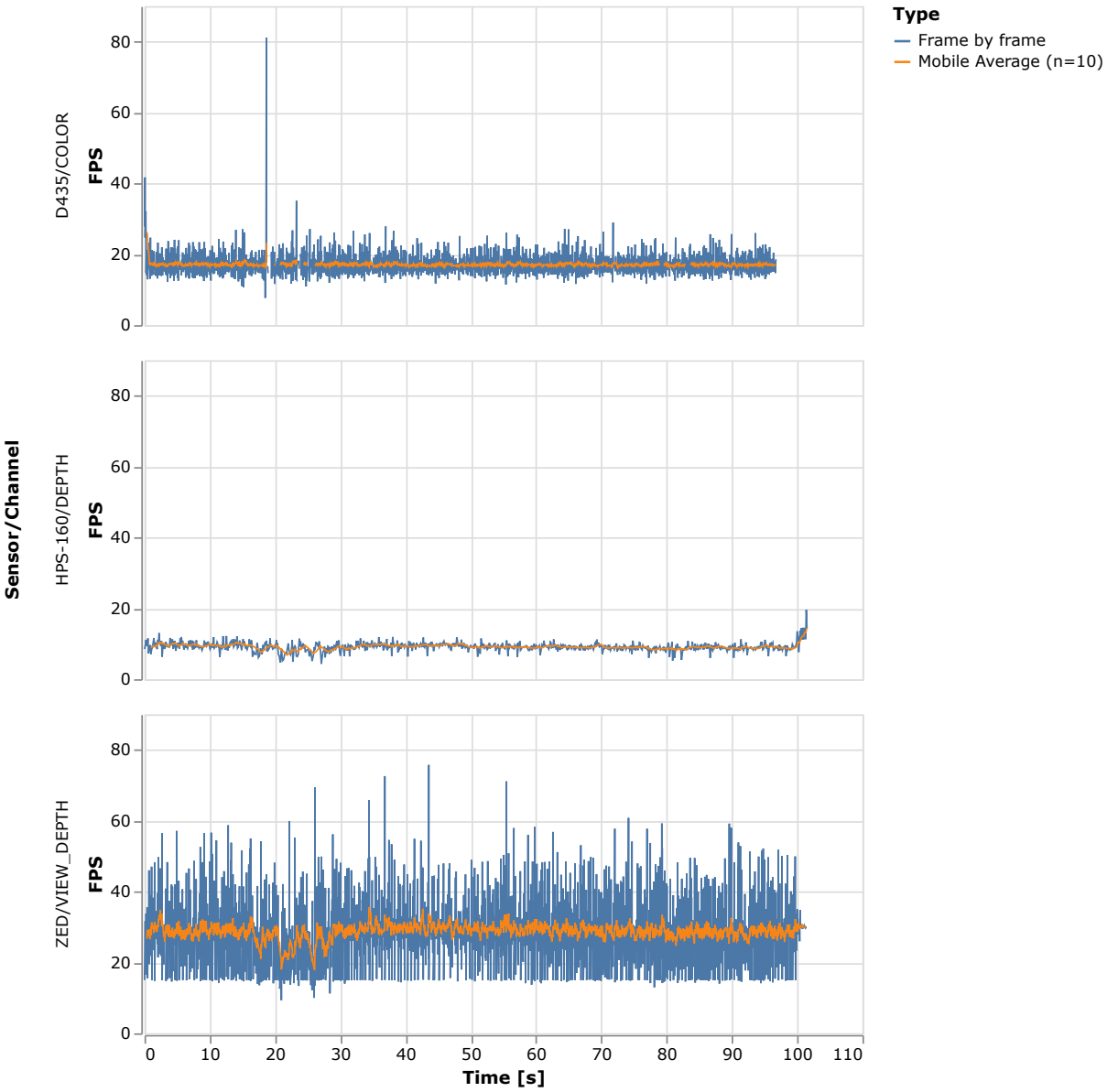
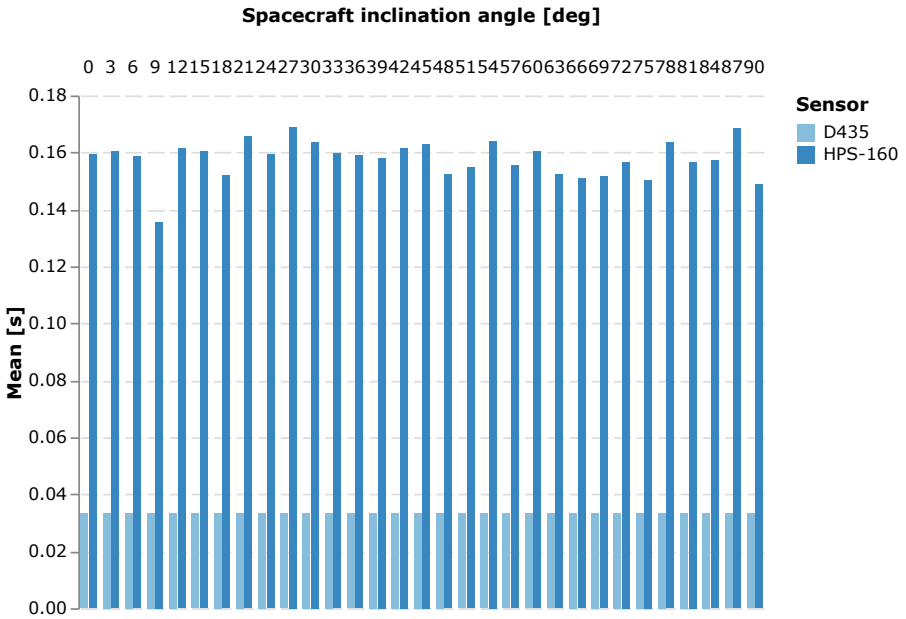
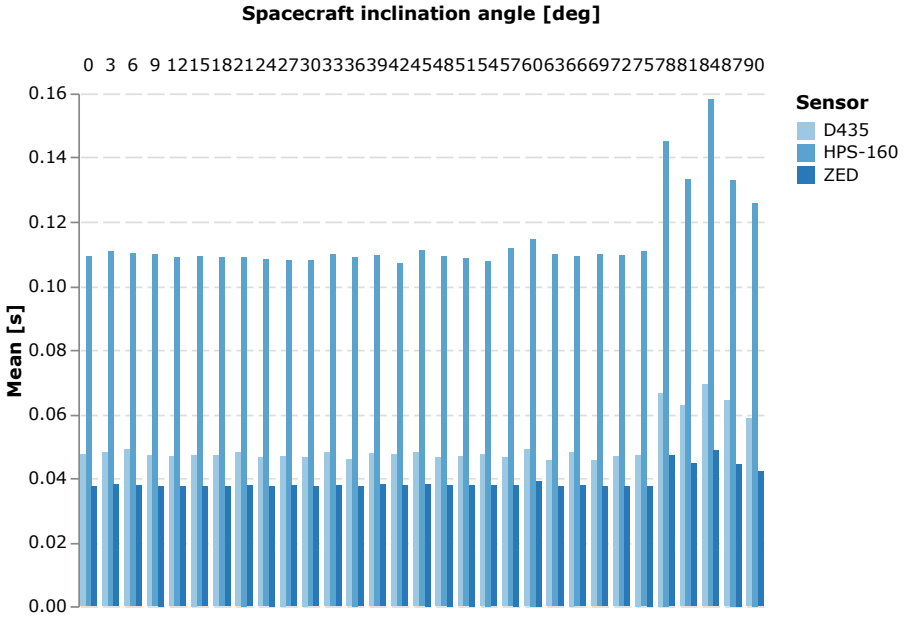


Fig. 7–18: Example of framerate of various devices

Finally, in fig. 7–19, it is possible to see an association between the average frames duration and the orientation of the target. In particular, in fig. 7–19a, it is shown the dark case for the D435 and the HPS-3D160, while fig. 7–19b presents the albedo case, where the ZED was also operated.



(a) Without albedo



(b) With albedo

Fig. 7–19: Time difference between keyframes in paths

8 Discussion

Before comparing the results, it is essential to notice the effect of the framerate of the various devices. As the original study was supposed to have the same light conditions and the same or comparable framerates for the ZED and D435, the comparison between the various case was meant to be done by skipping frames and, thus, virtually speed up the rotations of the satellite. The planned situation has not come to be for the following reasons:

- The failure of the original sun simulator changed the design of the experiment in using, in one recording, the Earth's albedo and, in the other, in keeping the light off. In this last situation, the ZED could not be used, as it is purely optical and requires external illumination;
- The HPS-3D160, added to the experiment in a second moment, do not offer the possibility to set a specific framerate for the HPS-3D160 LIDAR, as it depends on unspecified parameters;
- The only situation that was expected to present easily comparable results between the ZED and the D435 exhibited, despite previous tests, different framerates, and dropped frames in a substantial quantity.

These listed facts bring, on some occasions, missing data, and, in others, an extreme difference between the framerate (table 7–5), causing unsteadiness even inside single recordings (fig. 7–19). Considering the just listed facts, it is, therefore, necessary to have prudence while relating the results of the various parameters based only on the frame numbers, as a more complex time-based approach in calculating the paths and analyzing the results solution would be preferable. While the necessary information to follow this approach is readily available on the data structure (thanks to the widespread presence of the *timestamp* fields), the required implementation was not developed for this experiment for time reasons.

Nevertheless, as one of the main objectives of this work is to tests the implemented devices, it is, consequently, worthwhile to analyze their framerate under the tested conditions.

8.1 Framerate of the devices

The D435, after the preliminary tests showed in section 6.1.1, was set to record video at $30\,\mathrm{FPS}$ for both the color and infrared channels. Despite keeping both dark and albedo scenarios with identical settings, the final framerate for the color channel in the albedo tests, as visible in table 7–5, presents a value of approximately $16.61\,\mathrm{FPS}$.

The associated depth channel, on the other hand, exhibiting the predicted framerate, but it also shows a large quantity of dropped frames, with an average of $30.78\,\%$. As the path calculator skip missing frames, this reduction decreases the framerate to an

equivalent rate of 20.76 FPS. Furthermore, this decreased framerate is not even stable, as the average count of frames dropped presents a standard deviation of 185.61 frames.

This drop is visible only with the albedo on, as in the case with no external illumination, the depth channel presents no dropped frames and a very stable framerate of 29.98 FPS, with a frame period of $0.033\,352\,\mathrm{s} \pm 0.000\,004\,\mathrm{s}$. Further investigations suggest that this slowdown is connected to the autoexposure setting parameter of the camera, as the light provided by the albedo requires the internal camera algorithm to increase the exposition time on such a duration that it exceeds the $0.033\,33\,\mathrm{s}$ mark for a 30 FPS framerate, thus requiring to decrease it not to drop frames. This delay, together with the enabled *Auto-Exposure Priority* parameter [46], may prompt the camera to drop depth frames not in sync with the color information, leading to the observed outcome.

It is interesting to notice that this parameter alone could not be the whole reason for this lack of data, as frames drop in the framerate of the color was also present in the test data, but not the presence of dropped depth frame, as visible in table B.2. If only the final framerate of 30 FPS is considered, it is expected to count, for the 15 s of recording, approximately 450 frames. While this is not the case for the color channel (as it is presenting the same framerate of the experiment data of 16.61 FPS), this value matches the one counted for the depth channel, even if the *Auto-Exposure Priority* was enabled. The possible missing factor between the tests and experiments that may provoke this drop is the static condition of the satellite because, on the first situation, as visible in fig. 7–17, the number of dropped frames is not constant during the whole recording but is more frequent in certain positions.

The auto setting for the exposure is likely the same reason for the decreased framerate in the ZED, as the duration of each frame is $0.038\,49\,\mathrm{s} \pm 0.002\,64\,\mathrm{s}$ (25.98 FPS). While it is not possible, as it was arranged with the D435, to compare this value with the dark case, as the camera was disabled for those recordings since no depth data is available in the darkness, there is nonetheless the possibility to analyze old data gathered from previous works. After rapidly feeding those sources into the newly developed toolchain produced for this experiment, it has been measured, for a ZED camera video recorded under the light of the sun simulator, a frame period of $0.033\,33\,\mathrm{s} \pm 0.000\,10\,\mathrm{s}$ (30.00 FPS).

The framerate of the HPS-3D160 will be now briefly analyzed. It is interesting to see that the integration time does not affect only the framerate, but also its stability. For the integration time of 15 000 µs, each frame has a period of 0.1586 s (6.304 FPS) with a standard deviation of 0.006 394 s. When the integration time is decreased to 5000 µs, the duration goes down only to 0.1117 s (8.954 FPS), only a 29.59 % decrease despite the three times difference between the integration times. Furthermore, its standard deviation goes up to 0.010 54 s, 1.647 times bigger than the previous case. It is also worth remarking that these uncertainties are the biggest among the sensors, by a minimum of three times.

8.2 Reference Cases

Before delving into the results of the experiment, it is worth analyzing the results of the reference cases to understand the quality of the experiments' data adequately. Starting with the case with no skipped frames, thus removing the effect of the way the path tracking program handles the missing frames, the ZED camera presents, on average, the better consistency of the results in almost all conditions (with or without filter and with the DIFODO or the ZedSDK algorithm) for all but ε parameters. As visible in table 8–1, the ZED is for C_{F}, r_{F} and $\delta_{\max}$ least three times more consistent than the HPS-3D160, while compared with the D435 the discrepancy is of one order of magnitude. This striking difference also indicates the sensibility of the camera at slight variations in the laboratory conditions and numerical approximations, in particular for the newly tested devices. It is also possible to theorize a mutual interference among these two, as both operate in the IR spectrum and both project light in that range, but because no data are available with independent operations, no further inquiry can be made on this aspect.

Tab. 8–1: Standard deviation of the references relative to the ZED camera and with no frames skipped

Sensor	C_{F}	r_{F}	ε	R	$\delta_{\max}$
D435	10.8415	11.4443	0.1044	2.3748	10.8668
HPS-160	3.4913	3.9161	0.4311	1.4216	9.7502

The results are also particularly critical for R and $\delta_{\max}$, as these values are firmly tight with the proximity of the points to the best-fit circle. It is possible to observe that the filters mitigate this variation (in particular for the drift), hinting a significant impact of their estimated background in the final results.

Tab. 8–2: Standard deviation of the references relative to case with no frames skipped

Skipped	Sensor	C_{F}	r_{F}	ε	R	$\delta_{\max}$
1	**D435**	0.5997	0.5938	0.2321	0.7349	1.3017
	HPS-160	0.8439	0.7229	0.8978	0.6403	0.7571
	ZED	1.3519	1.4519	1.8681	1.1050	0.9176
2	**D435**	0.3942	0.3974	0.1053	0.4729	0.9483
	HPS-160	1.0281	0.8734	1.0463	0.8260	0.7154
	ZED	1.4078	1.3356	1.8856	0.8917	1.1162

As previously stated, the presence of dropped frames and unsteady framerates showed in table 7–5 may create more instability when skipping frames, with diverging results.

In comparing the result to the case with no skipped frames, as visible in table 8–2, it possible to see that, for the D435 (that suffers from dropped frames), the paths present even more concordance, with results similar also for the HPS-3D160. The uncertainties are, strangely, more prominent for the ZED, which has no frame drops. It is interesting to notice that this sensor presents a framerate with a standard deviation of $5.3648\,\mathrm{FPS}$, more significant than the $0.4194\,\mathrm{FPS}$ one of the D435 and the $3.080\,\mathrm{FPS}$ one of the HPS-3D160, but, at the moment, no way to identify if this the reason is known.

Finally, it is useful to examine the relative standard deviation, visible in table 8–3.

Tab. 8–3: Relative Standard deviation of the references relative to case with no frames skipped

		C_F	r_F	ε	R	δ_max
Algorithm	**Sensor**					
DIFODO	**D435**	$40.8745\,\%$	$9.6316\,\%$	$12.5234\,\%$	$63.7636\,\%$	$31.5212\,\%$
	HPS-160	$62.6577\,\%$	$10.5194\,\%$	$22.0613\,\%$	$46.2686\,\%$	$15.7991\,\%$
	ZED	$6.8688\,\%$	$5.0659\,\%$	$118.0085\,\%$	$54.1196\,\%$	$26.7339\,\%$
ZedSDK	**ZED**	$0.2790\,\%$	$0.1362\,\%$	$2.2791\,\%$	$5.8343\,\%$	$5.9481\,\%$

Analyzing the results for the ZED camera, it is interesting to see that its internal algorithm shows that all values are at least an order of magnitude smaller than the DIFODO. Despite the much lower intensity of light available, these values are also at least half of the one that this configuration presented in previous studies [15] with the same camera, but only for the ZedSDK algorithm, while the DIFODO shows similar results. It is essential to notice that, since those measurements were conducted, the SDK received significant upgrades, and in these experiments, the ULTRA setting for the depth quality was chosen. Apart from this detail, no general trend is visible for the various devices.

8.3 Dark Case

It is possible, looking at table 7–2, to see substantial values of the parameter δ_max in all the examined conditions. More interesting, averaging these results, it is visible a substantial drift of $\delta_\mathrm{max} = (179.2209 \pm 17.3013)°$. As also visible in the fig. 7–7, as expected, the angle grows linearly with time, starting from $0°$. This behavior is consistent with the one found in the previous studies, but with higher values. An example of these paths is visible in fig. 8–1.

From the left image in fig. 8–1, it is also interesting to see that, in some situations, a very sharp turn of the path can occur, while remaining on the best-fit plane. The maximum drift of this specific path is $194.324°$, not very far from the one on the path in the right image in fig. 8–1 (relatively to the average of this parameter for all results), with an angle of $160.700°$. At the moment, no reliable way has been found to identify these situations on a quantitative side.

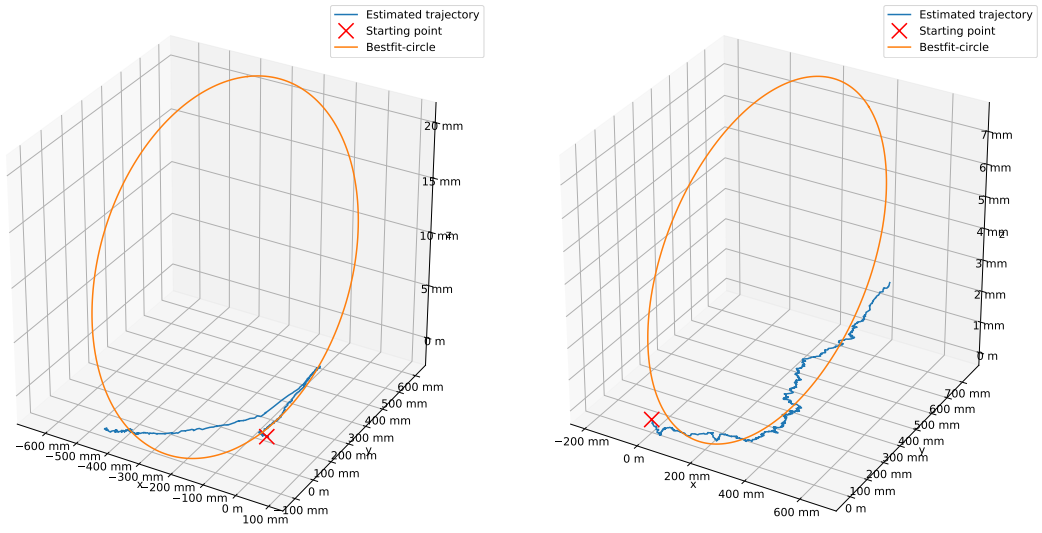


Fig. 8–1: Example of paths with a significant drift

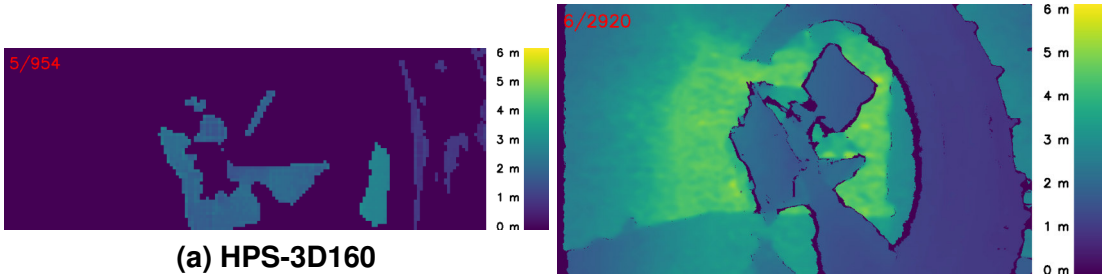
From the plots represented in fig. 7–7, it is also possible to see that, while the fitting line nicely describes the drift of the HPS-3D160, the same cannot always be stated for the D435. As noticeable, with the BoxFilter, a dip is located between $30\,\mathrm{s}$ and $60\,\mathrm{s}$. It is not clear the reason for this behavior, because this phenomenon is not present with the LIDAR, even if the HPS-3D160 does not record as much data about the background, while it also shows reflections. An example of the density of the depth data is visible in fig. 8–2. It is also noticeable in fig. 7–8 that this dip in the drift is only present for inclination between $0°$ and $45°$.

For the ε values visible in fig. 7–6, no consistent trend can be found for the HPS-3D160, while the results for the D435 show an almost constant line at $90°$. This value, contrary to what it may seem, does not indicate that the reference system is wrong, as the same algorithm, with the ZED (fig. 7–13), presents valid results. No valid reason is known for this trend.

This lack of depth data also explains the stable behavior of the drift when the filter is not applied, also visible in the fitting lines in fig. 7–7. These lines, apart from a weak negative correlation of the drift values as the angle increase for the filter case of the D435, do not show other notable trends as the target changes its inclination, and remain close to the mean value of the analyzed condition.

It is rather interesting to see, moving the analysis to the R parameter, that its values are much closer to zero and much lower than the average results in the previous experiments, where the worst situation is $0.1299\,\mathrm{m}$. As the residuum measures how close the path is to the fitting plane, this suggests that the calculated path can at least be approximate in a plane, correctly describing the real condition of the experiment. An example of this aspect can be seen in fig. 8–3.

Even looking at the plots in fig. 7–4 and their relative fitting lines, both sensors maintain a low residuum as the target increases its inclination, with the only exception of the



(a) HPS-3D160

(b) D435

Fig. 8–2: Depth difference between the D435 and the HPS-3D160

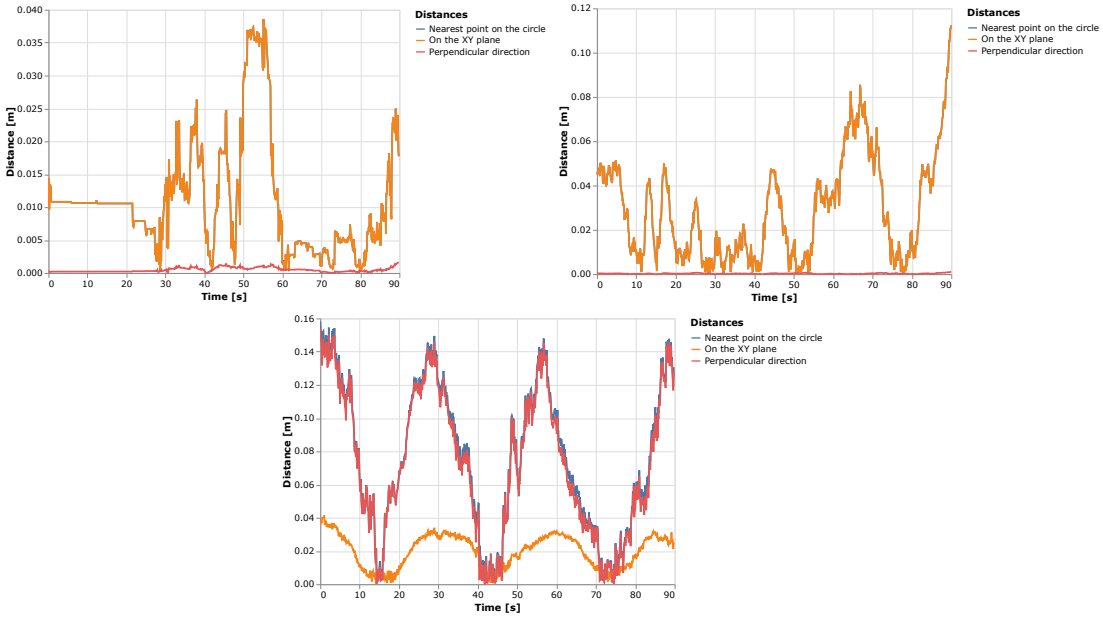


Fig. 8–3: Example of the distance between the path and the best-fit planes and circles

D435 without filter and without skipping frames, where no clear trend can be seen between $0°$ and $45°$.

Moving the discussion to the C_{F} and r_{F} parameters, it is, first of all, essential to keep in mind, as already stated, that their reference values were not directly available and had to be estimated by the data recorded in the previous thesis and updated with the knowledge of the state of the laboratory. By looking at the scatter plots in figs. 7–2 and 7–3, a more distinct picture (compared with the previous parameters), can be seen. While for the cases without filter the parameters are relatively constants or with small slopes (with an increased spread for the D435, as visible in the standard errors in table B.5), the BoxFilter presents, for the LIDAR at various angular velocities, more pronounced trends, but with peaks at the origin and from $45°$ to $60°$. As this is an active sensor, no justifiable explanation can be found.

Gathering the results for the various speeds, as already analyzed, no significant difference could be found. It is possible to see the relative values of the parameters compared with the case with no skipped frames in table 8–4.

Tab. 8–4: Values for the dark case parameters relative to the case with no skipped frames

Sensor	Filter	Skipped frames	C_{F}	r_{F}	ε	R	$\delta_{\max}$
D435	BoxFilter	1.0	0.6469	1.0599	0.1244	0.5174	0.7619
		2.0	0.5982	1.0651	0.0459	0.3779	0.7537
	None	1.0	0.5035	1.2048	0.6958	0.4474	1.0218
		2.0	0.3504	1.2684	0.0395	0.2679	1.0324
HPS-160	BoxFilter	1.0	1.0059	0.9930	1.0751	0.9329	0.9951
		2.0	1.0745	0.9702	0.9178	0.8505	1.0115
	None	1.0	0.9864	0.9966	0.9521	0.7582	1.0572
		2.0	0.9392	1.0020	0.9073	0.6809	1.0641

8.4 Albedo Case

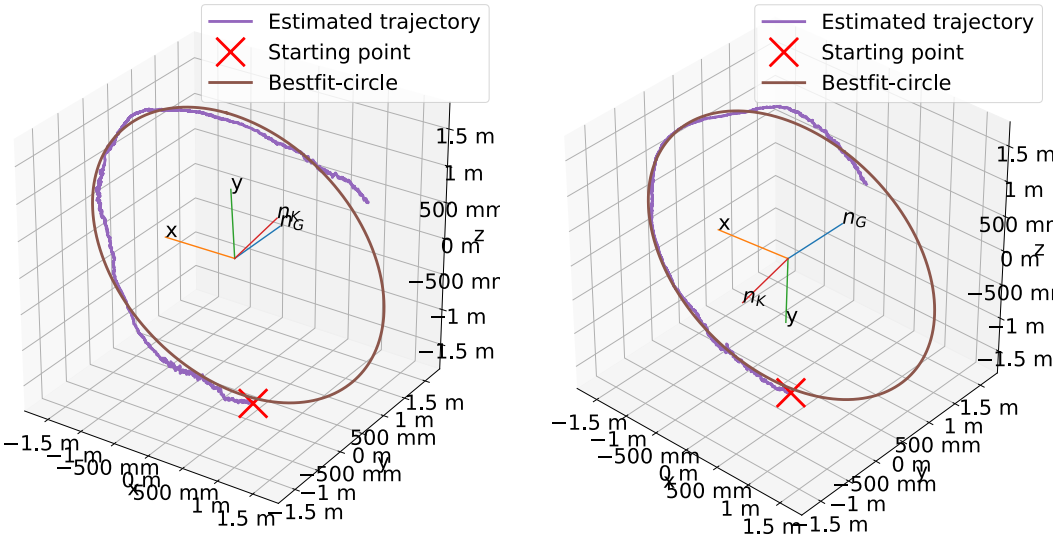
For the albedo case, looking at table 7–3, for the case at the base speed it is visible that, while the D435 and the HPS-3D160 present results similar to the dark case (all within the references' 2σ) for almost all parameters (with some exception only when two frames are skipped), it is interesting to notice a much smaller drift for the ZED camera. This fact is, in particular, valid for the BoxFilter case, and even more so for the ZedSDK algorithm, as visible in table 7–4. The average results are comparable with the $113.9850°$ found in the previous works, under the Sun simulator, for the DIFODO, while the drift is even lower for the ZedSDK, where was previously found a maximum angle of $76.9088°$. Even with the speed-up setup, the ZED still presents the lower value among the tested devices. Its evolution over time visible in fig. 7–14 shows an identical pattern of the dark case for the D435 and the LIDAR, including the dip halfway with the filter. Looking at the ZED, the DIFODO algorithm finds a similar response to the others for the case without a filter but with an initial almost null increase, while the BoxFilter presents much lower drift evolution for the entire duration of the experiment. In every case, it is worth noticing that the ZedSDK algorithm presents the lowest slope for the fitting line. If the evolution of the drift as the target changes its inclination is observed (fig. 7–12), it is again possible to observe a similar response of the D435 and HPS-3D160 of the dark case, while all the combinations of the ZED has a less spread out response around its constant, with no significant changes as the inclination increase.

As even for the parameters observable in figs. 7–9 to 7–11, for the D435 and the DIFODO, no significant changes are observed compared with the dark case, no further discussions are therefore here required.

Examining now the ZED, in fig. 7–11, it is visible that, while it does not differentiate itself from the other devices for the case without filter, it presents a decreasing trend with the BoxFilter, even if a peak is noticeable between $30°$ and $40°$ at all speeds. In

both figs. 7–9 and 7–10, it keeps a linear profile in all situations, with much less spread than the other devices.

The ε values visible in fig. 7–13, shows, similarly to the dark case, an almost constant line at $90°$ for the D435 and, on average, the same outcome for the HPS-3D160. Curiously enough, this is not the same thing that happens to both ZED algorithms, where the values are all around $0°$ or $180°$. As visible in fig. 8–4, this happens because, sometimes, the fitting algorithm fits the plane normal in the opposite direction but without having repercussions on the drift.



(a) Correct direction **(b) Wrong direction**

Fig. 8–4: Directions of the n_{K} versor find by the metric

On a final note, since this case presents a similar situation to the one analyzed by the previous thesis, it is worth directly comparing those results with the one here obtained by the ZED. Under the Sun simulator, the previous results, with the filtered DIFODO and without skipping frames, showed $C_{\mathrm{F}} = 0.7580\,\mathrm{m}$, $r_{\mathrm{F}} = 0.4060\,\mathrm{m}$ and $R = 0.1879\,\mathrm{m}$, while the ZedSDK presented $C_{\mathrm{F}} = 0.268\,813\,\mathrm{m}$, $r_{\mathrm{F}} = 0.012\,200\,\mathrm{m}$ and $R = 0.092\,025\,\mathrm{m}$. As visible in tables 7–3 and 7–4, these values are, for both the DIFODO and ZedSDK ZED's plane parameters, worst, while the residuum presents a lower value. It is also noticeable a much weaker link to the inclination of the target, as almost all parameters, as analyzed, describe a straight line. Their standard deviations are visible in table 8–5.

Tab. 8–5: Standard deviation of the BoxFilter case of the albedo without skipped frames

Algorithm	C_{F}	r_{F}	ε	R	$\delta_{\max}$
DIFODO	0.0538 m	0.0268 m	13.4312°	0.0914 m	12.7876°
ZedSDK	0.0189 m	0.0051 m	0.3556°	0.0045 m	3.9041°

Gathering the results for the various speeds, as already analyzed, even for the albedo case, no significant difference could be found. As was done for the dark case, in table 8–6, it is possible to see the values of the metric parameters relative to the case with no skipped frames.

Tab. 8–6: Values for the albedo case parameters relative to the case with no skipped frames

Sensor	Filter	Skipped frames	C_{F}	r_{F}	ε	R	$\delta_{\max}$
D435	BoxFilter	1.0	0.8220	1.0286	0.1232	0.4259	0.8348
		2.0	1.0548	0.9888	0.0416	0.4228	0.8870
	None	1.0	0.6199	1.0963	0.3886	0.4487	1.0783
		2.0	0.4930	1.1341	0.0265	0.3855	1.0449
HPS-160	BoxFilter	1.0	1.0835	0.9891	1.0291	0.8637	1.0692
		2.0	1.2664	0.9694	0.8841	0.7401	1.0760
	None	1.0	0.9791	0.9994	0.7192	0.8859	1.0340
		2.0	0.9905	0.9982	0.7759	0.8150	1.0754
ZED	BoxFilter	1.0	1.0162	0.9812	0.9728	0.7363	0.7002
		2.0	1.0275	0.9650	0.9888	0.5389	0.6562
	None	1.0	1.0099	0.9908	0.9898	1.0662	1.1253
		2.0	0.9994	1.0026	0.9694	1.0552	1.1883

8.5 Effect of the integration time on the HPS-3D160

As visible in fig. 8–2, the HPS-3D160, for the explored scenario, is not able to fill a wide area of its field of view with depth information, leaving many areas with invalid values. This aspect seems to contradict the $12\,\mathrm{m}$ range claimed by the producer [7]. Since no precise information has been found for the parameters of the LIDAR, an analysis of its depth capabilities can improve future usage of this sensor.

As, for time reasons, only two configurations have been tried, it is only possible to search for a linear trend correlated with the integration time. Other than the already mentioned, in section 7.4, time differences between the collection of two consecutive point clouds, it is also possible to search additional information in the cloud itself. As it is expected, given the nature of the HDR itself, increasing the integration time enables the sensor to fill the cloud with more data points. On a quantitive aspect, the relative filled area goes from an average of $56.1165\,\% \pm 13.9713\,\%$ for the $5000\,\mu\mathrm{s}$ integration time, to the value of $74.9028\,\% \pm 22.6900\,\%$ if the integrated period goes to $15\,000\,\mu\mathrm{s}$.

The new points detected are further located relative to the device, as the average moves from $1.981\,60\,\mathrm{m} \pm 0.062\,16\,\mathrm{m}$ to $2.2469\,\mathrm{m} \pm 0.0643\,\mathrm{m}$. This increase is a range

extension and not merely a shift in the detected values, as the maximum detected distance moves from $2.6069\,\text{m} \pm 0.1743\,\text{m}$ to $2.2700\,\text{m} \pm 0.1832\,\text{m}$, while the close range keeps relatively close values, from $1.6733\,\text{m} \pm 0.1325\,\text{m}$ to $1.6949\,\text{m} \pm 0.1037\,\text{m}$

9 Conclusion

The scope of this thesis required the integration of the D435 camera, the HPS-3D160 LIDAR in the RACOON-Lab, their testing under orbital light condition, the calculation of the traversed paths with the DIFODO algorithm and the ZedSDK algorithm and the analysis of the computed trajectory with the metric.

To summarize the results, it is possible to see that the primary and secondary objectives of this work have achieved. In particular:

1. The D435 camera and the HPS-3D160 LIDAR have been successfully integrated into the RACOON-Lab, with an extensible supports support for themselves and the chaser's computer. Softwares have been written to record, convert, and analyze the results;
2. Two measurements campaign has been executed, one with the presence of the Earth's albedo and another with no external illumination.
3. Paths have been calculated from the depth maps via the DIFODO and ZedSDK algorithm and analyzed via the metric.

The secondary objectives have also been fulfilled:

- Hardware and software grant a versatile and rapid extension of the laboratory with new devices, as the addition of the HPS-3D160 has proven. The LIDAR received its 3D-printed support, was added in the recorded procedure without any significant change to the interface, the conversion required no change in the data structure, and the path tracking algorithm required only a function to decode the stored data in a comprehensible way for the DIFODO;
- The data structure allowed many computers to read the data without the complex toolchain needed by the converter, necessitating only a few minutes to set up a working device.
- The metric was successfully converted in Python, refactoring its structure to being able to understand the source code better.

The metric's parameters of the calculated paths have been analyzed under various points of view to see the effects of the various algorithms, filters, and velocities, while also investigating the framerate of the various devices. The ZedSDK, when available, showed for almost all parameters the best and more consistent performances, followed by the DIFODO's ZED result, while the other sensors showed comparable results. The filter also showed the best performances with the ZED data (as its parameters were initially tailored to it), and its effects are especially visible on the drift of the path. With the other sensors, however, its effect is much less visible in the evolution of the drift, and often its ending values do not vary significantly. Finally, in particular, compared with previous results, no significant changes among the parameters have been found as the satellite changes its inclination.

Overall, no significant differences have been determined between the dark and albedo cases, meaning that the albedo light does not add a significant contribution to the one already available from the projector and the LIDAR laser.

The framerate has also being analyzed for all three devices. The ZED and D435 frames' durations are significantly impacted by the low light intensity offered by the albedo, as each frame requires longer exposition time, while the D435 exhibits the expected framerate when it is in the dark case. The HPS-3D160 is also impacted by its HDR integration time, with a drop of $33\,\%$ as the integration time increases.

9.1 Outlook

On the technical side, the development of this study has found problematic the lack of a precise interface between the recording software and the laboratory interface. In particular, its LabVIEW interface makes it impossible to have a time recording of the state of the laboratory, and further development in this aspect would allow having an accurate alignment of the recording with the RACOON-Lab state. The implementation of the metric (both the old one and the translated one) offers stringent requirements in its application, as it expects a constant framerate and a full $360°$ to be made, both of which were not accurate in the conducted experiments. The development of a new time-based implementation with more configuration settings would make the measurement more correct.

On the research side, better filters have to be developed. The simple BoxFilter can only work if the camera is fixed, and the C structure cannot be removed in this way, as it moves with the satellite. Furthermore, all sensors are unable to detect the presence of the reflecting surfaces, and, as acknowledged from the previous studies [3], the ghost components confuse the tracking algorithm.

Bibliography

[1] ESA Space Debris Office, “ESA’s Annual Space Environment Report 2019,” European Space Agency, Report GEN-DB-LOG-00271-OPS-SD, July 2019. [Online]. Available: https://www.sdo.esoc.esa.int/environment_report/Space_Environment_Report_latest.pdf

[2] R. Opromolla, G. Fasano, G. Rufino, and M. Grassi, “A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations,” *Progress in Aerospace Sciences*, vol. 93, pp. 53 – 72, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0376042117300428

[3] M. Dziura, T. Wiese, and J. Harder, “3D reconstruction in orbital proximity operations,” in *2017 IEEE Aerospace Conference*. IEEE, 2017, pp. 1–10.

[4] Lehrstuhl für Raumfahrttechnik. (2016) RACOON Lab. Technische Universität München. [Online]. Available: https://www.lrg.tum.de/lrt/forschung/robotic-operations/racoon-laboratory/

[5] Stereolabs. ZED Stereo Camera — Stereolabs. [Online]. Available: https://www.stereolabs.com/zed/

[6] Intel Corporation. Depth Camera D435 – Intel® RealSense™ Depth and Tracking Cameras. [Online]. Available: https://www.intelrealsense.com/depth-camera-d435/

[7] Hypersen Technologies Co., Ltd. 3D Solid-state LiDAR HPS-3D160 - Hypersen Technologies Co., Ltd. [Online]. Available: https://en.hypersen.com/product/detail/10.html

[8] The HDF Group. (1997-2020) Introduction to HDF5. [Online]. Available: https://portal.hdfgroup.org/display/HDF5/Introduction+to+HDF5

[9] *Intel® RealSense™ D400 Series Product Family*, Intel Corporation.

[10] *HPS-3D160*, Hypersen Technologies Co., Ltd.

[11] Intel Corporation. D400 Series Visual Presets. [Online]. Available: https://github.com/IntelRealSense/librealsense/wiki/D400-Series-Visual-Presets

[12] D. J. Kessler and B. G. Cour-Palais, “Collision frequency of artificial satellites: The creation of a debris belt,” *Journal of Geophysical Research: Space Physics*, vol. 83, no. A6, pp. 2637–2646, 1978. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/JA083iA06p02637

[13] B. B. Virgili, “The future of the environment,” ESA/ESOC Space Debris Office, May 2019, PowerPoint presentation.

[14] M. Jaimez and J. Gonzalez-Jimenez, “Fast visual odometry for 3-d range sensors,” *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 809–822, Aug 2015.

[15] N. Hab, “Execution and Evaluation of an Experimental Study for the Quantification of Influencing Factors on the Quality of Optical Pose Estimation in Earth’s Orbit,” Bachelorarbeit, Technical University of Munich, 2018, RT-BA 2018/12.

[16] F. Rehn, “Experimentelle studie mit RACOON und EPOS zur quantitativen Bewertung von Einflussfaktoren auf die Qualität einer 3D-Objektrekonstruktion im Erdorbit,” Bachelorarbeit, Technische Universität München, 2017, RT-BA-2017/01.

[17] M. Wieser, H. Richard, G. Hausmann, J.-C. Meyer, S. Jaekel, M. Lavagna, and R. Biesbroek, “e.deorbit mission: Ohb debris removal concepts,” in *ASTRA 2015 - 13th Symposium on Advanced Space Technologies in Robotics and Automation*, May 2015. [Online]. Available: https://elib.dlr.de/100732/

[18] V. Capuano, K. Kim, J. Hu, A. Harvard, and S.-J. Chung, “Monocular-based pose determination of uncooperative known and unknown space objects,” 2018.

[19] R. Biesbroek, “Active debris removal,” ESA, May 2019, PowerPoint presentation.

[20] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, “A survey on deep learning techniques for image and video semantic segmentation,” *Applied Soft Computing*, vol. 70, pp. 41 – 65, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494618302813

[21] H. Xu, J. Xu, and W. Xu, “Survey of 3d modeling using depth cameras,” *Virtual Reality & Intelligent Hardware*, vol. 1, no. 5, pp. 483 – 499, 2019, 3D Vision. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2096579619300646

[22] C. V. Poulton, A. Yaacobi, D. B. Cole, M. J. Byrd, M. Raval, D. Vermeulen, and M. R. Watts, “Coherent solid-state lidar with silicon photonic optical phased arrays,” *Opt. Lett.*, vol. 42, no. 20, pp. 4091–4094, Oct 2017. [Online]. Available: http://ol.osa.org/abstract.cfm?URI=ol-42-20-4091

[23] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, “Intel RealSense Stereoscopic Depth Cameras,” *CoRR*, vol. abs/1705.05548, 2017. [Online]. Available: http://arxiv.org/abs/1705.05548

[24] B. E. Tweddle, E. Muggler, A. Saenz-Otero, and D. Miller, “The SPHERES VERTIGO Goggles: Vision based mapping and localization onboard the International Space Station,” in *Proc. iSAIRAS*, 2012.

[25] F. Schnitzer, K. Janschek, and G. Willich, “Experimental results for image-based geometrical reconstruction for spacecraft rendezvous navigation with unknown and uncooperative target spacecraft,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 5040–5045.

[26] S. Sharma and S. D’Amico, “Pose estimation for non-cooperative rendezvous using neural networks,” 2019.

[27] F. Amzajerdian, V. E. Roback, A. Bulyshev, P. F. Brewster, and G. D. Hines, *Imaging Flash Lidar for Autonomous Safe Landing and Spacecraft Proximity Operation*. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2016-5591

[28] M. Dziura, "Benchmark of Computer-Vision-Based Tracking Performance in Proximity Operations," 2019.

[29] M. Jaimez and J. Gonzalez-Jimenez, "Fast Visual Odometry for 3-D Range Sensors," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 809–822, Aug 2015.

[30] Microsoft Corporation. Kinect - Windows app development. [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect/

[31] H. Sarbolandi, D. Lefloch, and A. Kolb, "Kinect range sensing: Structured-light versus Time-of-Flight Kinect," *Computer Vision and Image Understanding*, vol. 139, p. 1–20, Oct 2015. [Online]. Available: http://dx.doi.org/10.1016/j.cviu.2015.05.006

[32] The HDF Group. (1997-2020) Hierarchical Data Format, version 5. [Online]. Available: http://www.hdfgroup.org/HDF5/

[33] The University Corporation for Atmospheric Research. (1997-2020) Network Common Data Form, version 4. [Online]. Available: https://www.unidata.ucar.edu/software/netcdf/

[34] "DIN ISO 8601-1:2017-02, datenelemente und austauschformate - informationsaustausch - darstellung von datum und uhrzeit - teil 1: Grundlegende regeln (ISO/DIS 8601-1:2016); text deutsch und englisch." [Online]. Available: https://doi.org/10.31030/2604381

[35] S. v. d. Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011. [Online]. Available: https://aip.scitation.org/doi/abs/10.1109/MCSE.2011.37

[36] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[37] A. Collette, *Python and HDF5*. O'Reilly, 2013.

[38] W. McKinney, "Pandas: A Foundational Python Library for Data Analysis and Statistics," *Python for High Performance and Scientific Computing*, vol. 14, 2011.

[39] S. Hoyer and J. Hamman, “xarray: N-D labeled arrays and datasets in Python,” *Journal of Open Research Software*, vol. 5, no. 1, 2017. [Online]. Available: http://doi.org/10.5334/jors.148

[40] Dask Development Team, *Dask: Library for dynamic task scheduling*, 2016. [Online]. Available: https://dask.org

[41] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: https://aip.scitation.org/doi/abs/10.1109/MCSE.2007.55

[42] J. VanderPlas, B. Granger, J. Heer, D. Moritz, K. Wongsuphasawat, A. Satyanarayan, E. Lees, I. Timofeev, B. Welsh, and S. Sievert, “Altair: Interactive statistical visualizations for python,” *Journal of Open Source Software*, vol. 3, no. 32, p. 1057, 2018. [Online]. Available: https://doi.org/10.21105/joss.01057

[43] J. W. Anders Grunnet-Jepsen, John N. Sweetser, *Best-Known-Methods for Tuning Intel® RealSense™ D400 Depth Cameras for Best Performance*, Intel Corporation.

[44] *Intel® RealSense™ Camera - Depth Testing Methodology*, Intel Corporation.

[45] Intel Corporation. NetCDF Compression - Unidata Developer’s Blog. [Online]. Available: https://www.unidata.ucar.edu/blogs/developer/en/entry/netcdf_compression

[46] Strange Framerate behaviour when changing Exposure. [Online]. Available: https://github.com/IntelRealSense/librealsense/issues/1957

A Appendix: Examples of configuration files

Listing A.1: Example of settings.yaml

```
metadata:
  axis_0_steps: 37625
  axis_1_steps: -400
  # Other metadata
devices:
  ZED:
    fps: 30
    resolution: HD720
    depth_mode: PERFORMANCE
    depth_minimum_distance: -1
    sensing_mode: STANDARD
    reference_frame: CAMERA
    camera_disable_self_calib: True
    camera_image_flip: False
    enable_right_side_measure: False
    depth_stabilization: True
    sdk_verbose: True
    compression_mode: LOSSLESS
    enable_depth: False
    enable_point_cloud: False
  D435:
    Depth:
      width: 848
      height: 480
      fps: 30
      infrared_1_enabled: True
      infrared_2_enabled: True
      preset: "High Accuracy"
      laser_power: 180.0
      emitter_enabled: True
      depth_units: 0.0002
      emitter_on_off: False
    Color:
      width: 848
      height: 480
      fps: 30
      format: RGB8
      backlight_compensation_enabled: True
      brightness: 0.0
      contrast: 50.0
```

```
      gamma: 300.0
      hue: 0.0
      saturation: 64.0
      sharpness: 50.0
      auto_exposure_priority: True
  HPS-160:
    super_hdr:
      frame_number: 4
      max_integration_milliseconds: 2500
    enable_debug: false
```

Listing A.2: Example of settings.yaml

```
filters:
- alias: None
- alias: BoxFilter
  min_distance: 0.8
  max_distance: 2.0
  corner_1: [340, 60]
  corner_2: [940, 560]
  target_resolution_width: 1280
  target_resolution_height: 720
algorithms:
- alias: ZedSDK
  confidence_threshold: 85
  depth_max_range_value: 5
- alias: DIFODO
  skip_frames: 0
  downsample: 1
  cols: 320
  rows: 180
  ctf_levels: 4
  max_depth: 5.0
  target_resolution_width: 1280
  target_resolution_height: 720
```

B Appendix: Data

Tab. B.1: Results for the D435 parameters

ID	Fill-Rate	Plane Fit RMS Error	Subpixel RMS Error	Resolution	FPS	Preset
0	93.6344 %	3.6804 %	0.1501 px	1280×720	30	H. A.
1	93.0084 %	3.7126 %	0.1517 px	1280×720	15	H. A.
2	99.9199 %	3.2037 %	0.0858 px	848×480	15	H. A.
3	99.9084 %	3.2483 %	0.0870 px	848×480	30	H. A.
4	99.7985 %	3.2793 %	0.0879 px	848×480	90	H. A.
5	96.3213 %	5.3534 %	0.1381 px	640×480	90	H. A.
6	96.2205 %	5.3607 %	0.1386 px	640×480	30	H. A.
7	96.0526 %	5.4092 %	0.1402 px	640×480	15	H. A.
8	98.1062 %	4.9488 %	0.1038 px	640×360	15	H. A.
9	98.1019 %	4.9203 %	0.1032 px	640×360	30	H. A.
10	98.1608 %	4.9188 %	0.1029 px	640×360	90	H. A.
11	78.5405 %	5.1787 %	0.0773 px	480×270	90	H. A.
12	79.6462 %	5.2625 %	0.0787 px	480×270	30	H. A.
13	83.9996 %	5.2754 %	0.0790 px	480×270	15	H. A.
14	61.7405 %	5.8671 %	0.0817 px	424×240	30	H. A.
15	57.5936 %	5.8753 %	0.0814 px	424×240	60	H. A.
16	57.6241 %	5.8691 %	0.0812 px	424×240	90	H. A.
17	99.4973 %	4.9325 %	0.2066 px	1280×720	30	H. D.
18	99.5185 %	4.9520 %	0.2073 px	1280×720	15	H. D.
19	100.0000 %	4.0188 %	0.1100 px	848×480	15	H. D.
20	99.9998 %	4.0166 %	0.1099 px	848×480	30	H. D.
21	99.9999 %	4.0052 %	0.1094 px	848×480	60	H. D.
22	99.9996 %	4.0144 %	0.1097 px	848×480	90	H. D.
23	99.9983 %	4.0680 %	0.1047 px	640×480	90	H. D.
24	99.9984 %	4.0224 %	0.1036 px	640×480	60	H. D.
25	99.9983 %	4.0014 %	0.1032 px	640×480	30	H. D.

Continued on next page

Tab. B.1: Results for the D435 parameters

ID	Fill-Rate	Plane Fit RMS Error	Subpixel RMS Error	Resolution	FPS	Preset
26	99.9975 %	4.0233 %	0.1039 px	640×480	15	H. D.
27	99.6608 %	4.6630 %	0.0701 px	480×270	15	H. D.
28	99.6333 %	4.6452 %	0.0698 px	480×270	30	H. D.
29	99.6228 %	4.5600 %	0.0684 px	480×270	60	H. D.
30	99.6260 %	4.5682 %	0.0685 px	480×270	90	H. D.

Tab. B.2: List of tests for dropped frames

	Session			Depth		Color		Results
ID	**FPS**	**Depth Resolution**	**Color Frames**	**Dropped Frames**		**Dropped Frames**	**Dropped**	**Dropped**
1	30	424×240	448	0		251	0	0.0000 %
2	30	848×480	450	0		250	0	0.0000 %
3	30	640×360	447	0		250	0	0.0000 %
4	30	960×540	448	13		250	0	1.8625 %
5	60	1920×1080	830	125		0	0	15.0602 %
6	30	1280×720	449	159		251	0	22.7143 %
7	60	424×240	898	276		251	0	24.0209 %
8	60	640×360	912	335		254	0	28.7307 %
9	60	848×480	910	432		253	0	37.1453 %
10	60	960×540	910	475		254	0	40.8076 %
11	90	1920×1080	1276	570		0	0	44.6708 %
12	90	424×240	1360	747		253	0	46.3112 %
13	90	640×360	1345	790		250	0	49.5298 %
14	60	1280×720	915	627		254	8	54.3199 %
15	90	848×480	1362	903		253	0	55.9133 %
16	90	960×540	1346	948		251	0	59.3613 %
17	90	1280×720	1359	1079		251	11	67.7019 %

Tab. B.3: Scale factor effects

	Depth FPS	Resolution	Color Fraction with holes	Scale factor
0	30	1920×1080	6.3580 %	0.0002
1	60	1920×1080	6.1055 %	0.0002
2	90	1920×1080	5.9759 %	0.0002
3	30	1280×720	6.5154 %	0.0002
4	60	1280×720	6.0845 %	0.0002
5	90	1280×720	6.2557 %	0.0002
6	30	960×540	6.4290 %	0.0002
7	60	960×540	6.2584 %	0.0002
8	90	960×540	6.0716 %	0.0002
9	30	848×480	6.3321 %	0.0002
10	60	848×480	6.4229 %	0.0002
11	90	848×480	5.7203 %	0.0002
12	30	640×360	6.5316 %	0.0002
13	60	640×360	6.3906 %	0.0002
14	90	640×360	5.9275 %	0.0002
15	30	424×240	6.3514 %	0.0002
16	60	424×240	6.3252 %	0.0002
17	90	424×240	6.0801 %	0.0002
18	30	1280×720	6.5233 %	0.0001
19	30	1280×720	6.2141 %	0.0001
20	60	1280×720	6.2774 %	0.0001
21	60	1280×720	6.0782 %	0.0001
22	90	1280×720	6.2400 %	0.0001
23	90	1280×720	5.6256 %	0.0001
24	30	1280×720	6.5501 %	0.0002
25	30	1280×720	6.2868 %	0.0002
26	60	1280×720	6.3510 %	0.0002
27	60	1280×720	5.9877 %	0.0002
28	90	1280×720	6.1701 %	0.0002
29	90	1280×720	5.7220 %	0.0002

Continued on next page

Tab. B.3: Scale factor effects

	Depth FPS	Resolution	Color Fraction with holes	Scale factor
30	30	1280×720	6.6297 %	0.0005
31	30	1280×720	6.3100 %	0.0005
32	60	1280×720	6.4690 %	0.0005
33	60	1280×720	6.0152 %	0.0005
34	90	1280×720	6.1710 %	0.0005
35	90	1280×720	5.7196 %	0.0005

Tab. B.4: Measured timestamps of the origin and completion of the swipe maneuver

	Session	Start	Stop
0	20200221T141238.484460800+0000	14:14:17.330000	14:12:47.330000
1	20200221T141659.279895200+0000	14:18:37.979000	14:17:07.979000
2	20200221T142050.233259500+0000	14:22:30.853000	14:21:00.853000
3	20200221T142457.796636200+0000	14:26:36.241000	14:25:06.241000
4	20200221T142903.949621000+0000	14:30:41.954000	14:29:11.954000
5	20200221T143334.109409500+0000	14:35:12.198000	14:33:42.198000
6	20200221T143823.593815400+0000	14:40:01.706000	14:38:31.706000
7	20200221T144246.127202700+0000	14:44:23.911000	14:42:53.911000
8	20200221T144718.507568600+0000	14:48:56.261000	14:47:26.261000
9	20200221T145212.814385400+0000	14:53:50.459000	14:52:20.459000
10	20200221T145708.933453100+0000	14:58:46.597000	14:57:16.597000
11	20200221T150247.607317800+0000	15:04:24.775000	15:02:54.775000
12	20200221T150748.034147900+0000	15:09:25.290000	15:07:55.290000
13	20200221T160126.440225500+0000	16:03:05.458000	16:01:35.458000
14	20200221T160508.816318100+0000	16:06:47.794000	16:05:17.794000
15	20200221T160927.596425500+0000	16:11:06.326000	16:09:36.326000
16	20200221T161511.426084100+0000	16:16:50.122000	16:15:20.122000
17	20200221T161943.775978400+0000	16:21:22.634000	16:19:52.634000
18	20200221T162351.771345000+0000	16:25:30.487000	16:24:00.487000

Continued on next page

Tab. B.4: Measured timestamps of the origin and completion of the swipe maneuver

	Session	Start	Stop
19	20200221T162835.450825500+0000	16:30:14.267000	16:28:44.267000
20	20200221T163218.195298000+0000	16:33:56.925000	16:32:26.925000
21	20200221T163657.058093600+0000	16:38:35.758000	16:37:05.758000
22	20200221T164213.349318100+0000	16:43:51.995000	16:42:21.995000
23	20200221T164611.522823500+0000	16:47:50.146000	16:46:20.146000
24	20200221T165041.852401600+0000	16:52:20.328000	16:50:50.328000
25	20200221T165449.757374100+0000	16:56:28.243000	16:54:58.243000
26	20200221T171052.672544400+0000	17:12:31.044000	17:11:01.044000
27	20200221T171450.308880000+0000	17:16:28.627000	17:14:58.627000
28	20200221T171911.279664300+0000	17:20:49.614000	17:19:19.614000
29	20200221T172336.671331000+0000	17:25:14.785000	17:23:44.785000
30	20200221T172819.844047400+0000	17:29:58.034000	17:28:28.034000
31	20200221T132133.478275900+0000	13:23:11.820000	13:21:41.820000
32	20200221T132658.259450500+0000	13:28:36.747000	13:27:06.747000
33	20200221T133126.068860900+0000	13:33:04.333000	13:31:34.333000
34	20200221T133554.389464200+0000	13:37:32.842000	13:36:02.842000
35	20200221T134014.731426300+0000	13:41:53.081000	13:40:23.081000
36	20200221T134417.153217400+0000	13:45:56.135000	13:44:26.135000
37	20200221T134814.064140800+0000	13:49:52.566000	13:48:22.566000
38	20200221T135214.786923200+0000	13:53:53.367000	13:52:23.367000
39	20200221T135630.824964900+0000	13:58:08.932000	13:56:38.932000
40	20200221T140028.538258700+0000	14:02:07.022000	14:00:37.022000
41	20200219T102226.408819700+0000	10:24:20.687000	10:22:50.687000
42	20200219T103238.261749100+0000	10:34:15.264000	10:32:45.264000
43	20200219T103917.124311700+0000	10:40:54.534000	10:39:24.534000
44	20200219T104555.202113800+0000	10:47:33.070000	10:46:03.070000
45	20200219T105238.612877200+0000	10:54:15.737000	10:52:45.737000
46	20200219T105755.025045000+0000	10:59:32.019000	10:58:02.019000
47	20200219T110236.726511600+0000	11:04:12.490000	11:02:42.490000
48	20200219T110740.957500100+0000	11:09:18.041000	11:07:48.041000

Continued on next page

Tab. B.4: Measured timestamps of the origin and completion of the swipe maneuver

	Session	Start	Stop
49	20200219T111211.666802000+0000	11:13:48.312000	11:12:18.312000
50	20200219T111652.051880200+0000	11:18:29.187000	11:16:59.187000
51	20200219T112208.502657600+0000	11:23:45.837000	11:22:15.837000
52	20200219T112727.095824800+0000	11:29:04.165000	11:27:34.165000
53	20200219T113209.695873900+0000	11:33:46.755000	11:32:16.755000
54	20200219T113707.153126400+0000	11:38:44.453000	11:37:14.453000
55	20200219T114203.112219300+0000	11:43:40.214000	11:42:10.214000
56	20200219T114657.199483100+0000	11:48:34.462000	11:47:04.462000
57	20200219T151018.273167000+0000	15:11:56.102000	15:10:26.102000
58	20200219T151555.614665900+0000	15:17:33.221000	15:16:03.221000
59	20200219T151938.180721300+0000	15:21:15.844000	15:19:45.844000
60	20200219T152329.916258300+0000	15:25:07.029000	15:23:37.029000
61	20200219T152720.870705300+0000	15:28:58.042000	15:27:28.042000
62	20200219T153121.562514800+0000	15:32:58.531000	15:31:28.531000
63	20200219T153608.343811100+0000	15:37:45.718000	15:36:15.718000
64	20200219T154125.664013700+0000	15:43:02.882000	15:41:32.882000
65	20200219T154758.821108500+0000	15:49:35.970000	15:48:05.970000
66	20200219T155450.356839200+0000	15:56:27.597000	15:54:57.597000
67	20200219T160452.891532700+0000	16:06:30.016000	16:05:00.016000
68	20200219T161016.123760700+0000	16:11:53.242000	16:10:23.242000
69	20200219T161514.394626700+0000	16:16:50.606000	16:15:20.606000
70	20200219T162135.779043900+0000	16:23:12.883000	16:21:42.883000
71	20200219T162615.144356800+0000	16:27:52.280000	16:26:22.280000
72	20200221T112910.816101100+0000	11:30:49.752000	11:29:19.752000
73	20200221T113414.852025200+0000	11:35:53.812000	11:34:23.812000
74	20200221T113916.747543400+0000	11:40:55.691000	11:39:25.691000
75	20200221T114353.883023100+0000	11:45:32.623000	11:44:02.623000
76	20200221T114727.299898300+0000	11:49:06.263000	11:47:36.263000
77	20200221T115048.859353000+0000	11:52:27.583000	11:50:57.583000
78	20200221T115425.395165000+0000	11:56:04.212000	11:54:34.212000

Continued on next page

Tab. B.4: Measured timestamps of the origin and completion of the swipe maneuver

	Session	Start	Stop
79	20200221T121110.346419800+0000	12:12:48.906000	12:11:18.906000
80	20200221T121643.594799800+0000	12:18:22.228000	12:16:52.228000
81	20200221T122104.438632700+0000	12:22:42.814000	12:21:12.814000

Tab. B.5: Coefficients of the fitting lines for the dark case's parameters

Skipped	Filter	Sensor	Algorithm	Parameter	Slope	Intercept	Standard error
1.0	**BoxFilter**	**D435**	**DIFODO**	C_F [m]	0.0018	0.3758	0.0008
				r_F [m]	0.0004	3.6818	0.0008
				ε [rad]	−0.0009	1.6357	0.0002
				δ_max [rad]	−0.8243	214.1888	0.1793
		HPS-160	**DIFODO**	C_F [m]	−0.0128	1.5976	0.0021
				r_F [m]	0.0135	2.4924	0.0022
				ε [rad]	0.0125	1.0367	0.0043
				R [m]	−0.0002	0.0564	0.0001
				δ_max [rad]	−0.0369	182.7879	0.1895
	None	**D435**	**DIFODO**	C_F [m]	−0.0033	0.8914	0.0018
				r_F [m]	0.0028	3.3438	0.0020
				ε [rad]	0.0000	1.5584	0.0002
				R [m]	−0.0002	0.0678	0.0002
				δ_max [rad]	−0.1161	174.7215	0.2609
		HPS-160	**DIFODO**	C_F [m]	0.0025	0.4265	0.0007
				r_F [m]	−0.0026	3.7086	0.0007
				ε [rad]	0.0066	1.5382	0.0032
				R [m]	−0.0001	0.0232	0.0000
				δ_max [rad]	0.2759	162.4134	0.1578
	BoxFilter	**D435**	**DIFODO**	C_F [m]	0.0060	0.1466	0.0010
				r_F [m]	−0.0054	3.9706	0.0010

Continued on next page

Tab. B.5: Coefficients of the fitting lines for the dark case's parameters

Skipped	Filter	Sensor	Algorithm	Parameter	Slope	Intercept	Standard error
				ε [rad]	−0.0004	1.6012	0.0001
				$\delta_{\max}$ [rad]	0.4428	153.3680	0.1364
				C_{F} [m]	−0.0125	1.6532	0.0021
				r_{F} [m]	0.0135	2.4229	0.0022
		HPS-160	**DIFODO**	ε [rad]	0.0091	1.0212	0.0048
				R [m]	−0.0002	0.0523	0.0001
				$\delta_{\max}$ [rad]	−0.1249	189.7228	0.1516
				C_{F} [m]	−0.0040	0.6956	0.0011
				r_{F} [m]	0.0042	3.4637	0.0013
		D435	**DIFODO**	ε [rad]	−0.0001	1.5743	0.0001
				R [m]	0.0002	0.0275	0.0001
				$\delta_{\max}$ [rad]	−0.3619	187.5361	0.1710
	None			C_{F} [m]	0.0022	0.4124	0.0008
				r_{F} [m]	−0.0024	3.7190	0.0008
		HPS-160	**DIFODO**	ε [rad]	0.0083	1.5302	0.0032
				R [m]	0.0000	0.0205	0.0000
				$\delta_{\max}$ [rad]	0.3400	160.6706	0.1651
				C_{F} [m]	−0.0073	1.0494	0.0023
				r_{F} [m]	0.0109	2.9850	0.0021
		D435	**DIFODO**	ε [rad]	−0.0079	2.1116	0.0014
				$\delta_{\max}$ [rad]	−1.7658	312.9097	0.2588
	BoxFilter			C_{F} [m]	−0.0134	1.6196	0.0024
				r_{F} [m]	0.0141	2.4887	0.0025
		HPS-160	**DIFODO**	ε [rad]	0.0114	1.2122	0.0042
				R [m]	0.0000	0.0535	0.0001
				$\delta_{\max}$ [rad]	0.1941	173.2744	0.2191
0.0				C_{F} [m]	0.0040	1.2916	0.0027
				r_{F} [m]	−0.0095	3.3059	0.0032
		D435	**DIFODO**	ε [rad]	0.0013	1.4512	0.0006

Continued on next page

	None						

Tab. B.5: Coefficients of the fitting lines for the dark case's parameters

Skipped Filter	Sensor	Algorithm	Parameter	Slope	Intercept	Standard error
			R [m]	−0.0024	0.2385	0.0004
			$\delta_{\max}$ [rad]	0.9916	121.2557	0.1787
			C_{F} [m]	0.0038	0.3732	0.0006
			r_{F} [m]	−0.0042	3.7897	0.0006
	HPS-160	**DIFODO**	ε [rad]	0.0101	1.3608	0.0022
			R [m]	−0.0002	0.0340	0.0000
			$\delta_{\max}$ [rad]	0.4255	146.2191	0.1357

Tab. B.6: Coefficients of the fitting lines for the albedo case's parameters

Skipped Filter	Sensor	Algorithm	Parameter	Slope	Intercept	Standard error
			C_{F} [m]	0.0097	0.1340	0.0020
			r_{F} [m]	−0.0099	3.9894	0.0021
	D435	**DIFODO**	ε [rad]	0.0001	1.5756	0.0001
			$\delta_{\max}$ [rad]	0.6212	158.7008	0.1937
			C_{F} [m]	−0.0096	0.9660	0.0022
			r_{F} [m]	0.0088	3.1939	0.0022
	HPS-160	**DIFODO**	ε [rad]	0.0070	1.3515	0.0043
BoxFilter			R [m]	−0.0002	0.0449	0.0001
			$\delta_{\max}$ [rad]	−0.2599	178.9483	0.2968
			C_{F} [m]	−0.0007	2.1708	0.0002
			r_{F} [m]	0.0001	1.9968	0.0001
	ZED	**DIFODO**	ε [rad]	0.0353	0.1265	0.0071
			R [m]	−0.0015	0.1580	0.0002
			$\delta_{\max}$ [rad]	−0.0321	35.3994	0.0600
2.0			C_{F} [m]	−0.0071	0.8429	0.0016
			r_{F} [m]	0.0070	3.3264	0.0018
	D435	**DIFODO**	ε [rad]	−0.0002	1.5827	0.0001
			R [m]	0.0001	0.0292	0.0002

Continued on next page

None

Tab. B.6: Coefficients of the fitting lines for the albedo case's parameters

Skipped Filter	Sensor	Algorithm	Parameter	Slope	Intercept	Standard error
			$\delta_{\max}$ [rad]	−0.4341	190.8602	0.2747
			C_{F} [m]	0.0013	0.1958	0.0003
			r_{F} [m]	−0.0014	3.9692	0.0005
	HPS-160	**DIFODO**	ε [rad]	0.0082	1.3985	0.0018
			R [m]	0.0000	0.0215	0.0001
			$\delta_{\max}$ [rad]	0.0870	140.3651	0.1515
			C_{F} [m]	−0.0020	1.9086	0.0002
			r_{F} [m]	0.0023	2.1922	0.0003
	ZED	**DIFODO**	ε [rad]	−0.0009	0.3226	0.0034
			R [m]	−0.0001	0.0581	0.0001
			$\delta_{\max}$ [rad]	−0.0202	122.7835	0.0170
			C_{F} [m]	−0.0020	0.6762	0.0017
	D435	**DIFODO**	r_{F} [m]	0.0038	3.3622	0.0016
			ε [rad]	0.0016	1.6147	0.0015
			$\delta_{\max}$ [rad]	−1.3921	280.7935	0.3651
			C_{F} [m]	−0.0066	0.7177	0.0012
			r_{F} [m]	0.0064	3.4139	0.0012
BoxFilter	**HPS-160**	**DIFODO**	ε [rad]	−0.0029	1.7805	0.0027
			R [m]	−0.0003	0.0622	0.0001
			$\delta_{\max}$ [rad]	0.0279	154.1798	0.2908
			C_{F} [m]	−0.0017	2.1587	0.0002
			r_{F} [m]	0.0006	2.0448	0.0001
	ZED	**DIFODO**	ε [rad]	0.0266	0.4924	0.0076
			R [m]	−0.0026	0.2842	0.0004
			$\delta_{\max}$ [rad]	0.1715	44.0259	0.0810
0.0			C_{F} [m]	−0.0066	1.3561	0.0023
			r_{F} [m]	0.0052	2.9768	0.0028
	D435	**DIFODO**	ε [rad]	−0.0006	1.5847	0.0008
			R [m]	−0.0005	0.1140	0.0004

Continued on next page

None

Tab. B.6: Coefficients of the fitting lines for the albedo case's parameters

Skipped Filter	Sensor	Algorithm	Parameter	Slope	Intercept	Standard error
			$\delta_{\max}$ [rad]	0.0347	162.4012	0.2670
			C_{F} [m]	0.0013	0.1972	0.0003
			r_{F} [m]	−0.0015	3.9772	0.0004
	HPS-160	**DIFODO**	ε [rad]	0.0052	1.4619	0.0017
			R [m]	0.0000	0.0286	0.0001
			$\delta_{\max}$ [rad]	−0.0591	136.8285	0.1658
			C_{F} [m]	−0.0018	1.8973	0.0002
			r_{F} [m]	0.0018	2.2080	0.0002
	ZED	**DIFODO**	ε [rad]	−0.0219	1.8203	0.0064
			R [m]	−0.0003	0.0652	0.0001
			$\delta_{\max}$ [rad]	−0.0115	103.0818	0.0161
			C_{F} [m]	0.0022	0.3624	0.0012
	D435	**DIFODO**	r_{F} [m]	−0.0016	3.7279	0.0014
			ε [rad]	0.0001	1.5822	0.0002
			$\delta_{\max}$ [rad]	0.0847	173.6124	0.2372
			C_{F} [m]	−0.0090	0.8597	0.0015
			r_{F} [m]	0.0085	3.2800	0.0016
	HPS-160	**DIFODO**	ε [rad]	−0.0014	1.6414	0.0034
BoxFilter			R [m]	−0.0003	0.0556	0.0001
			$\delta_{\max}$ [rad]	−0.2645	178.1006	0.2662
			C_{F} [m]	−0.0013	2.1762	0.0002
			r_{F} [m]	0.0004	2.0165	0.0001
	ZED	**DIFODO**	ε [rad]	0.0423	−0.3648	0.0052
			R [m]	−0.0020	0.2142	0.0003
			$\delta_{\max}$ [rad]	0.0205	35.3091	0.0693
1.0			C_{F} [m]	−0.0067	0.9594	0.0019
			r_{F} [m]	0.0066	3.2235	0.0021
	D435	**DIFODO**	ε [rad]	−0.0001	1.5791	0.0002
			R [m]	−0.0001	0.0433	0.0002

Continued on next page

None

Tab. B.6: Coefficients of the fitting lines for the albedo case's parameters

Skipped Filter	Sensor	Algorithm	Parameter	Slope	Intercept	Standard error
			$\delta_{\max}$ [rad]	−0.3311	191.7053	0.2473
	HPS-160	DIFODO	C_{F} [m]	0.0013	0.1934	0.0003
			r_{F} [m]	−0.0013	3.9675	0.0004
			ε [rad]	0.0050	1.4088	0.0017
			R [m]	0.0000	0.0245	0.0001
			$\delta_{\max}$ [rad]	−0.0511	141.0349	0.1597
	ZED	DIFODO	C_{F} [m]	−0.0020	1.9240	0.0002
			r_{F} [m]	0.0022	2.1709	0.0002
			ε [rad]	−0.0047	0.6657	0.0056
			R [m]	−0.0001	0.0606	0.0001
			$\delta_{\max}$ [rad]	−0.0277	116.6624	0.0154

Tab. B.7: Coefficients of the fitting lines for the dark case's average drifts. The resulting values are in degrees

Filter	Skipped	Sensor	Slope	Intercept	Std. Err.
BoxFilter	0	D435	0.0348	−0.1083	0.0019
		HPS-160	0.0310	0.2261	0.0007
	1	D435	0.0286	0.3459	0.0011
		HPS-160	0.0334	0.1624	0.0005
	2	D435	0.0256	0.4187	0.0013
		HPS-160	0.0355	0.1255	0.0003
None	0	D435	0.0288	−0.0058	0.0006
		HPS-160	0.0321	0.0261	0.0002
	1	D435	0.0309	−0.1189	0.0007
		HPS-160	0.0348	−0.0038	0.0002
	2	D435	0.0329	−0.1810	0.0007
		HPS-160	0.0353	−0.0047	0.0002

Tab. B.8: Coefficients of the fitting lines for the albedo case's average drifts. The resulting values are in degrees

Filter	Skipped	Sensor	Slope	Intercept	Std. Err.
BoxFilter	0	**D435**	0.0355	0.1567	0.0009
		HPS-160	0.0224	0.3535	0.0007
		ZED (DIFODO)	0.0120	−0.0725	0.0002
	1	**D435**	0.0322	0.3248	0.0009
		HPS-160	0.0248	0.3087	0.0007
		ZED (DIFODO)	0.0083	−0.0373	0.0002
	2	**D435**	0.0316	0.3769	0.0010
		HPS-160	0.0290	0.2643	0.0006
		ZED (DIFODO)	0.0077	−0.0267	0.0001
None	0	**D435**	0.0319	−0.1274	0.0006
		HPS-160	0.0225	0.1908	0.0007
		ZED (DIFODO)	0.0247	−0.2788	0.0006
		ZED (SDK)	0.0019	−0.0265	0.0001
	1	**D435**	0.0349	−0.1484	0.0006
		HPS-160	0.0250	0.1759	0.0008
		ZED (DIFODO)	0.0274	−0.3014	0.0006
	2	**D435**	0.0335	−0.2052	0.0008
		HPS-160	0.0270	0.1114	0.0008
		ZED (DIFODO)	0.0288	−0.3193	0.0006