FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Advanced visualization for deep space telemetry applications

**Beatriz Gomes da Silva**


Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Paulo Garcia

July 25, 2024

# Resumo

Os dados tornaram-se um bem central e a sua visualização desempenha um papel crucial na interação humana. Uma visualização eficaz dos dados não só melhora a interpretabilidade dos mesmos, como também facilita a implementação dos princípios de dados FAIR (localizáveis, acessiveis, interoperáveis, reutilizáveis), que são essenciais para garantir que os dados estão disponíveis e úteis para uma vasta gama de utilizadores. Esta dissertação de mestrado centra-se em dados provenientes de ótica adaptativa, uma classe especializada de instrumentos eletro-óticos. A equipa onde decorre esta dissertação publicou recentemente o formato Adaptive Optics Telemetry (AOT), que visa padronizar os dados de telemetria gerados por sistemas de ótica adaptativa em observatórios terrestres. No entanto, a sua usabilidade continua a ser limitada pelos conhecimentos de programação do utilizador e pela sua familiaridade com o pacote Python que o acompanha (aotpy).
Existe uma oportunidade para melhorar substancialmente a acessibilidade dos dados, oferecendo aos utilizadores uma ferramenta alternativa para realizar análises exploratórias de dados de forma visual e intuitiva. Vários estudos destacam a eficácia das dashboards para este fim, uma vez que permitem aos utilizadores identificar rapidamente padrões ou tendências, e a sua conceção interativa facilita a compreensão e a exploração dos conjuntos de dados.
Foi desenhada e desenvolvida uma dashboard em Python de código aberto para facilitar a exploração dos dados AOT. A ferramenta apresenta um vasto leque de funcionalidades interactivas para a análise e exploração de dados, incluindo métodos estatísticos tradicionais, criação de gráficos, manipulação e visualização de imagens, apresentação de tabelas e visualização de dados dependentes do tempo. Esta permite os utilizadores navegar sem esforço pelos conjuntos de dados e correlacionar pontos de dados, enquanto adere aos princípios de dados FAIR. O método envolveu uma análise exaustiva das directrizes de conceção de design de dashboards e das ferramentas de visualização de dados existentes, tanto na comunidade de dados espaciais como nas ferramentas de uso geral. Isto garante que a ferramenta cumpre todos os requisitos e, ao mesmo tempo, cumpre as directrizes de dados abertos, assegurando a acessibilidade do código-fonte aos utilizadores.
Por fim, está disponível uma dashboard de fácil utilização para descarregar, analisar e explorar dados de telemetria AO. A dashboard serve como uma alternativa visual às abordagens programáticas atuais, expandindo a acessibilidade e a utilização dos dados AOT entre um público mais vasto.

# Abstract

Data has become a central commodity, and its visualization plays a crucial role in human interaction. Effective visualization not only enhances the interpretability of data but also facilitates the implementation of the FAIR (Findable, Accessible, Interoperable, Reusable) data principles, which are essential for ensuring that data is available and useful to a wide range of users. This master's dissertation focuses on data from adaptive optics, a specialized class of electro-optic instruments. The team where this dissertation takes place has recently published the Adaptive Optics Telemetry (AOT) format, which aims to standardize the telemetry data generated by adaptive optics systems in ground-based observatories. However its usability remains limited by the user's programming expertise and familiarity with the accompanying Python package (aotpy).

There is an opportunity for substantial improvement in data accessibility by offering users an alternative tool for conducting exploratory data analysis visually and intuitively. Various studies highlight the effectiveness of dashboards for this purpose, as they allow users to quickly identify patterns or trends, and their interactive design facilitates the understanding and exploration of the datasets.

An open-source Python dashboard was designed and developed to facilitate the exploration of AOT data. The tool features a wide array of interactive functionalities for data analysis and exploration, including traditional statistical methods, data plotting, image manipulation and visualization, table display, and time-dependent data visualization. This allows users to navigate datasets effortlessly and correlate data points, while adhering to the FAIR data principles. The method involved an extensive review of dashboard design guidelines and existing data visualization tools, both within the space-based data community and general-purpose tools. This ensures that the tool meets all the requirements while complying with open data guidelines, ensuring accessibility of the source code to users.

Ultimately, it is available a user-friendly dashboard for downloading, analyzing, and exploring AO telemetry data. The dashboard serves as a visual alternative to current programmatic approaches, expanding the accessibility and utilization of AOT data among a broader audience.

# United Nations Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) framework addresses global challenges and promotes sustainable development in various areas, including poverty, inequality, climate change, environmental degradation, peace, and justice. The SDGs consist of seventeen interlinked global goals designed to achieve a better and more sustainable future for all by 2030 [1]. In my dissertation on adaptive optics telemetry (AOT), I explore innovations in data accessibility and visualization, which are essential for advancing scientific knowledge and achieving SDGs.

The interactive features of the dashboard developed in this study facilitate effective data exploration, enabling the identification of trends and patterns within adaptive optics telemetry data. This capability contributes to scientific advancements and aligns with quality education. By enhancing data transparency and supporting evidence-based decision-making, particularly in scientific and policy domains, this research aligns to foster sustainable development globally, reduced inequalities, and global partnerships. The dashboard supports capacity building and knowledge dissemination by promoting learning and skill development in data analysis techniques among researchers and students.

Promoting innovation and enhancing data-driven insights, it aims to contribute positively to achieving sustainable development outcomes worldwide. Table 1 shows how this dissertation can help achieve some SDGs' targets.

[1] It is possible to see the goals for the United Nations at https://sdgs.un.org/goals

Table 1: SDGs' targets covered by this dissertation.

| SDG | Target | Contributions | Performance Indicators and Metrics |
|---|---|---|---|
| 4. Quality Education | 4.1 | Enhances learning by providing interactive data visualization tools that make complex data more understandable for students. | Indicator 4.1.1: Percentage of students achieving a minimum proficiency level in reading and mathematics. Metric: Number of schools adopting data visualization tools. |
| | 4.4 | Improves data literacy and technical skills among students and educators, preparing them for a data-driven world. | Indicator 4.4.1: Proportion of youth and adults with information and communications technology (ICT) skills, by type of skill. Metric: Number of training sessions conducted. |
| 9. Industry, Innovation, and Infrastructure | 9.5 | Promotes innovation by offering a tool that aids in the research and development of new engineering solutions. | Indicator 9.5.1: Research and development expenditure as a proportion of GDP. Metric: Increase in the number of research projects utilizing the tool; amount of funding allocated to projects using the tool. |
| 10. Reduced Inequalities | 10.2 | Reduces inequalities in access to adaptive optics data by making it easier to manipulate and use, allowing countries with less experience to benefit from this technology. | Indicator 10.2.1: Proportion of people living below 50% of median income by age, sex, and persons with disabilities. Metric: Number of countries and institutions with access to the tool. |
| 17. Partnerships for the Goals | 17.6 | Facilitates international collaboration by providing a standardized dashboard for sharing adaptive optics telemetry data, enhancing global partnerships. | Indicator 17.6.1: Number of science and technology cooperation agreements and programs. Metric: Number of international collaborations and partnerships formed using the dashboard. |

# Acknowledgements

Firstly, I would like to thank my supervisor, Paulo Garcia, for his guidance and helpfulness. I'm also grateful to Doctor Carlos Correia for his expertise. A big thanks to the entire team where this project took place for being so welcoming, especially Tiago Gomes for all the advice, patience, and teaching along the way.

Special words of gratitude and happiness to my family for always being my biggest support from a distance. Especially my sister, Carlota, who was always by my side during these five years. It wouldn't have been the same without you.

Lastly, to all my friends, the friends from home who were very understanding of my absence and still gave me confidence even though we were far away. To the ones I met in college, I'm happy to have had you all by my side during these years, with numerous study sessions and hangouts, especially to the ones who always made my coffee breaks so full. And to my dear João, for all the kindness.

Beatriz Gomes da Silva

*"To win a race, the swiftness of a dart*
*Availeth not without a timely start*
*The hare and tortoise are my witnesses."*

Jean de La Fontaine

# Contents

# List of Figures

# List of Tables

# Abbreviations and symbols

| | |
|---|---|
| AO | Adaptive Optics |
| AOT | Adaptive Optics Telemetry |
| ESA | European Space Agency |
| ESO | European Southern Observatory |
| MCT | Mission Control Technologies |
| CDF | Compute Cumulative Distribution Function |
| GUI | Graphical User Interface |
| SDGs | Sustainable Development Goals |
| UML | Unified Modeling Language |

# Chapter 1

# Introduction

In today's data-driven world, the effective visualization and interpretation of data play a critical role in numerous scientific and technological domains. Although there have been significant advancements in data collection and processing from ground-based observatories, an important area is still poorly explored in effectively visualizing complex astronomical datasets from the world's largest telescopes, particularly those operated by the European Southern Observatory (ESO). This dissertation fills this void by developing an intuitive visualization tool designed to handle telemetry data from some of the world's largest telescopes. This tool enhances the accessibility and usability of such data, enabling better insights and more efficient research.

## 1.1 Context

In astronomy, the performance of ground-based telescopes is hindered by atmospheric turbulence, which emerges from complex interactions between different layers of the Earth's atmosphere. This turbulence causes refractive phenomena that distort the light originating from astronomical objects, resulting in a "blurring" of the images produced by these telescopes. Modern ground-based telescopes employ Adaptive Optics (AO) to mitigate this issue [47]. This technique aims to correct the light distortions by measuring them with wavefront sensors and then precisely deforming mirrors to compensate for them. A simple diagram illustrating the adaptive optics system is presented in Figure 1.1. It demonstrates how the components work together to correct the distorted wavefronts of light from a telescope. The system includes an adaptive mirror, a wavefront sensor, a control system, and a beamsplitter. The light from the telescope, which has a distorted wavefront due to atmospheric turbulence, first hits the adaptive mirror. This mirror undergoes continuous deformation to rectify these distortions. The corrected wavefront is subsequently divided by the beamsplitter; one fraction is directed to the wavefront sensor for analysis, while the corrected wavefront advances towards the high-resolution camera. The control system, a real-time computer, uses the measurements from the wavefront sensor to dynamically adjust the mirror, thereby ensuring the camera captures images clearly and sharply. An example of the effects of AO can be seen in Figure 1.2.

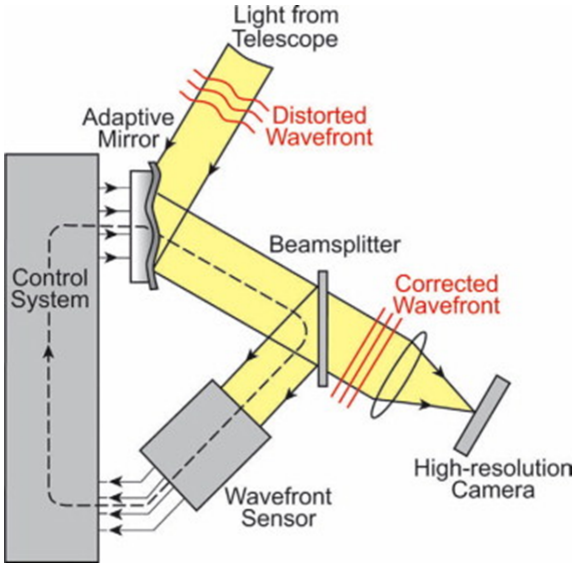



Figure 1.1: Diagram of the adaptive optics system. Light from the telescope is distorted due to atmospheric turbulence. The adaptive mirror corrects the wavefront, which is then split by the beamsplitter, to both the wavefront sensor and the high-resolution camera. The wavefront sensor sends measurements to the control system, enabling real-time adjustments of the mirror. In parallel, the high-resolution camera captures the corrected wavefront [47].

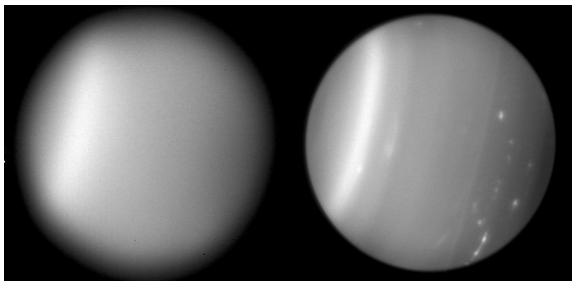


Figure 1.2: Illustration of the enhancing capacity of adaptive optics. On the left, an uncorrected image obtained by a ground-based telescope is presented. The same object is imaged on the right, but this time, corrected with adaptive optics. The second image has a higher angular resolution (or acuity) enabled by the adaptive optics correction of the "blur" (optical aberrations) caused by the turbulence of the Earth's atmosphere. Image credit: Bruce Murray Space Image Library.

The use of adaptive optics has led to an abundance of telemetry information and datasets [20]. In this context, the term "telemetry" represents adaptive optics internal signals such as wavefront sensor measurements, deformable mirror commands, and several other reconstruction, control matrices and parameters. The telemetry data resulting from these electro-optic systems can take various forms, including integers, floats, data cubes, higher order matrices, lists, and other diverse dimensions.

Adaptive optics datasets are scientifically interesting given their several applications [21], such as:

- System Performance: they can be leveraged to optimize the performance of adaptive optics systems on ground-based telescopes, ensuring they operate at their full potential, resulting in clearer views of celestial objects;
- Atmospheric Studies: the datasets can aid in monitoring atmospheric conditions and turbulence;
- Research: the datasets can be used to recover information either for instrumentation research or to apply in research using the instruments (for example, for impulse response reconstruction);
- Education: the datasets can be used by researchers and students who want to learn more about adaptive optics but cannot access the instruments.

One example of a type of adaptive optics telemetry data are the measurements produced by a Shack-Hartmann wavefront sensor. These are optical sensors commonly used for adaptive optics systems due to their ability to measure wavefront distortions with high accuracy and sensitivity. The adaptability of these sensors allows for real-time corrections of optical aberrations caused by atmospheric turbulence or other factors. Their working principle can be seen in Figure 1.3. Additionally, in Figure 1.4 we have a real-world example of one frame of data produced by the detector of a Shack-Hartmann wavefront sensor. In this example, we have a 4x4 grid of lenses (subapertures), resulting in the 16 spots.

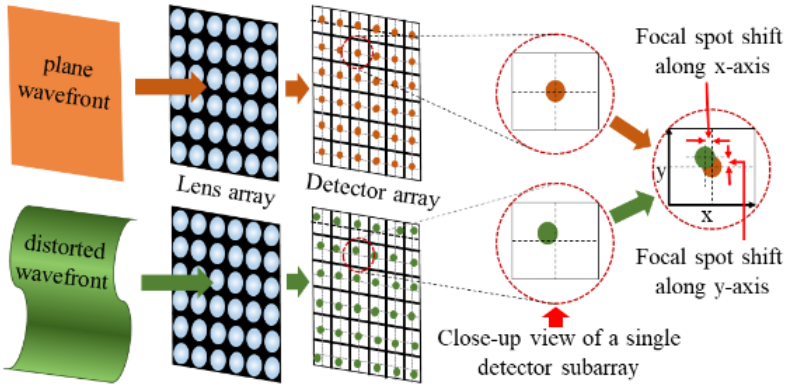



Figure 1.3: Working principle of a Shack–Hartmann wavefront sensor. Light from an astronomical object is shone through a set of lenses (subapertures) placed before a detector array. The positions of the spots of light on the detector result from the shape of the wavefront. The shifts from the central spot positions can be used to calculate the wavefront distortion, allowing it to be corrected with a mirror of an inverse shape. Image credit: [12].

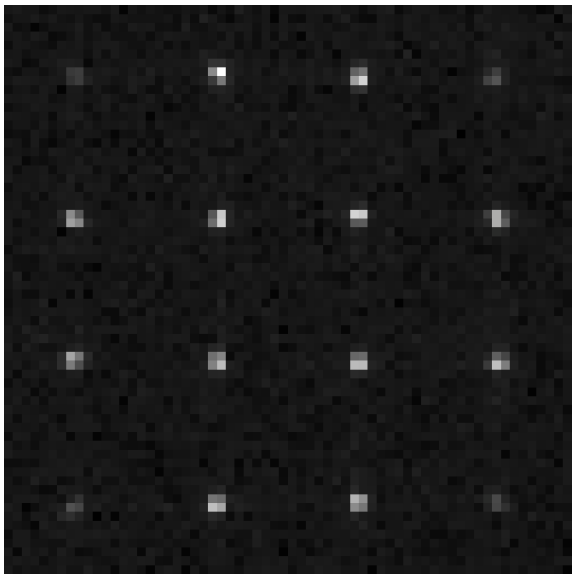


Figure 1.4: Shack–Hartmann spots image. (Source: ESO Archive Program ID 60.A-9278(D)). This is one frame (2D image) out of thousands of frames collected over time (resulting in a set of 3D data, a datacube). These frames vary over time due to atmospheric turbulence and the corrections done by mirrors in the system.

Furthermore, recent findings underscore the critical need for advanced web-based visualization tools in the field of astronomy. According to a survey conducted by the European Space Agency (ESA) Astronomy Archives User Group [8], the highest priority for future developments includes advanced web-based visualization and cut-out services. This demand reflects the broader astronomy community's need for improved tools to facilitate efficient data analysis and research activities.

## 1.2 Statement of the problem

Gomes et al. have recently proposed the Adaptive Optics Telemetry (AOT) standard [21] as a way of providing uniform access to telemetry data produced by multiple instruments. While an accompanying Python package is also provided (*aotpy*), the authors do not propose any methods or techniques for visualizing the data in such files. Due to this, users who are inexperienced with Python or not particularly familiar with the format may struggle with interpreting such data in an efficient manner. The AO community benefits from a ready-made tool that allows for the visualization of key telemetry data without requiring any prior Python or AOT experience.

Figure 1.5 shows a UML (Unified Modeling Language) class diagram that provides an overview of the data model used in *aotpy*. In general terms, when an AOT file is opened with *aotpy*, it returns a single AOSystem object. This object provides some high-level metadata about the data contained in the file (such as the system's name and the data acquisition period), and it also provides references to other objects in the system, namely the Main Telescope, Sources, Wavefront Sensors, Wavefront Correctors, Scoring Cameras, Loops and Atmospheric Parameters collected at

the time of observation. Each of these objects provides detailed information about its corresponding part of the system, granting access to the relevant telemetry data that has been recorded, as well as any other relevant configurations. Types of data are most often used by AO experts to gain insight into a system's high-level behavior, and therefore, their visualization must be prioritized.

One particularly challenging aspect of this task is that, as seen in the diagram, an AO system's configuration is highly variable. For example, the amount of wavefront sensors, light sources, wavefront correctors and loops that exist can vary from system to system. On top of that, the type of the components themselves can differ (e.g., a Wavefront Sensor may be of the Shack-Hartmann or Pyramid type), which will, in turn, alter the type of data that they store. This means that any solution for visualizing data would require a careful implementation to ensure that it is versatile and robust enough to deal with the diversity of real-world AO data.

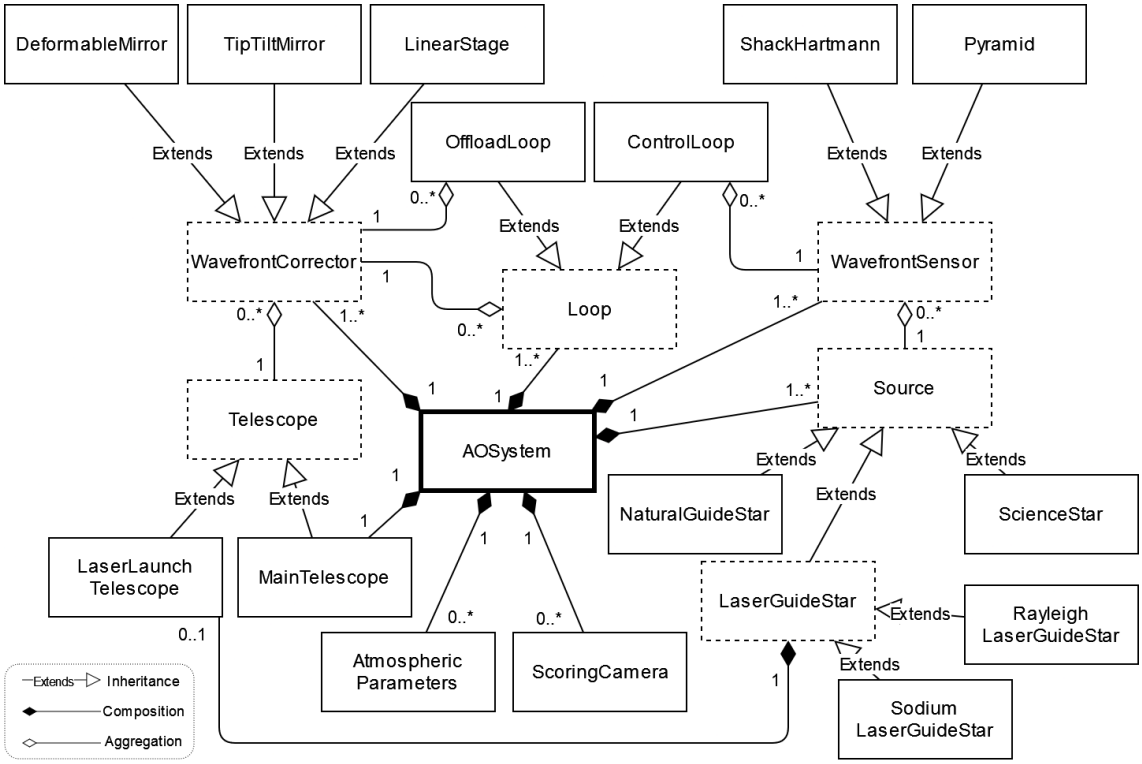



Figure 1.5: Simplified UML Class Diagram representing the AOT data model from [21].

## 1.3 Solution

To address these problems, we developed an open-source Python visualization tool, specifically a dashboard. Such a tool would make it easier for both researchers and users to work with telemetry data across multiple data production systems. Using dashboards is a good way to visualize and analyze data because they provide a well-organized graphical user interface (GUI) and consistent visual elements display. They are also a valuable tool for data exploration and analysis, especially

in the context of the AOT format, as they can enable the exploration of different instruments with different data sets that are inherently very complex and can have a hierarchical structure.

The data telemetry visualization displays recorded data and allows for a comparison between simulated and actual telemetry data. Finally, the data tool should adhere to the FAIR data management principles [55].

## 1.4 Structure of this dissertation

This document is divided into four chapters:

Chapter 1 introduces the context of this research and gives basic background into the problem to be solved, its statement, and the research objectives.

Chapter 2 addresses the state of the art. Reviewing current aspects that are relevant to the dissertation, including definitions of dashboards, guidelines, software, and related work.

Chapter 3 details the methodology employed in developing the AOT data visualization tool. It covers the requirements analysis and visual design considerations. Additionally, it discusses the implementation details of the dashboard.

Chapter 4 presents the results of the dashboard, including an evaluation of the tool's effectiveness in facilitating data exploration and analysis.

Finally, Chapter 5 summarizes the findings from the research, discusses the conclusions drawn based on the results and offers suggestions for future work.

# Chapter 2

# State of the Art

This chapter provides a comprehensive overview of the sources used to thoroughly understand the various aspects that this dissertation will address. It aims to highlight the significance of the dashboard and its importance in the context of this study. Additionally, we will present design guidelines, evaluation practices, examples of existing tools, FAIR data principles, and software review.

## 2.1 Dashboard fundamentals

Dashboards provide visual displays of data for monitoring and comprehension [54], and they deliver well-organized, graphical user interfaces and a consistent display of visual elements [6]. They often serve as a primary interaction point, allowing users to grasp information visually before delving into deeper analysis [58, 4]. With these characteristics and the fact that they are such ubiquitous tools used across various industries and organizations, it is only natural that they are often an object of research [42].

Researchers have proposed multiple definitions for what is the purpose of an ideal dashboard. The authors in [16] claim that "The aim of a dashboard is to tell a story, that's why we need a way to link all the dashboard's charts that bold a connection and give the story we need.". Additionally, authors in [29] refer that "The basic idea of visual analytics is to visually represent information, allowing the human to directly interact with it, to gain insight, to draw conclusions, and to ultimately make better decisions.". These definitions help us keep in mind what is the core purpose of a dashboard and must be taken into account when developing one.

There are several ways in which we can categorize a dashboard. One such way is when we consider the symbiosis between dashboards and analytics, which is the field that delves into the systematic use of data analysis and interpretation to gain valuable insights, make informed decisions, and optimize outcomes. According to [35], analytical dashboards give users a higher situational awareness, i.e., a better understanding of the matter at hand, and lead to a higher task performance.

Analytics can also be performed in different temporal contexts depending on the analyzed data, whether it is past, present, or future. These are descriptive, predictive, and prescriptive, as seen in Figure 2.1. Descriptive analytics involves explaining past events by analyzing historical data, predictive analytics uses this historical data to forecast future events, and prescriptive analytics goes a step further by recommending actions based on predictive analytics to achieve the best possible outcomes. Each type serves a distinct purpose, but when combined together can help in extracting actionable intelligence to guide decision-making, improve efficiency, and drive strategic initiatives [7]. Furthermore, a study indicates that dashboards incorporating descriptive and prescriptive analytics promote development more effectively than those without any analytics [51]. Our dashboard must incorporate descriptive analytics, as the data will be from past files, but not prescriptive as the user does not require recommending.

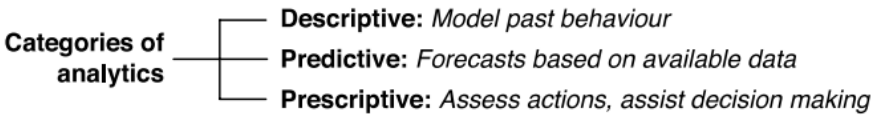



Figure 2.1: Categories of analytics from [7].

In past years and complementing analytical dashboards, interactive analytical dashboards [35] emerged as a way to engage users in a more immersive experience in response to the increasing need to display varied and large quantities of datasets coherently. The authors in [42] extend on this definition, albeit calling this functional dashboards, stating they allow for real-time monitoring of dynamically updated data, in contrast with visual dashboards, which are static layouts of simple charts. These interactive dashboards go beyond simply displaying data and enabling users to interact with the data in real-time: they allow users to interact with the data and perform analytical tasks directly within the dashboard interface, as they can filter, sort, and drill down into specific data points to better understand the information presented; they offer a wide range of functional features, including point-and-click interactivity, zooming on interesting points, applying filters drag and drop, checkbox, range sliders, details-on-demand, relationship exploration, history tracking, actions to support undo, replay, progress, extraction of sub-collections of query parameters [44], and interactive analytical tools that facilitate operational decision-makers to simulate trends automatically; they allow for displaying multidimensional data in 2D, which will be extremely useful for representing the datasets considered in this dissertation. The authors in [35] state that interactive analytical dashboards outperform static ones, and even though they can be more confusing due to the new capabilities given to the user, they have fewer issues in handling complex and multidimensional data.

### 2.1.1 Design Guidelines

Although there are no specific rules on how a dashboard should be designed, it is imperative to follow a set of guidelines to design an effective and coherent dashboard, which go far beyond aesthetics.

The authors in [37] suggested a template for a dashboard development roadmap:

1. Selecting the Key Metrics;
2. Populating the Dashboard With Data;
3. Establishing Relationships Between the Dashboard Items;
4. Forecasting and Scenarios.

Following a predefined process for the dashboard creation will ensure that all steps are covered and made in a logical sequence. In addition to having a well-established roadmap to follow, the designer should also take into account the following aspects [32]:

- **Importance of context**: knowing who the users are, their level of knowledge, their information needs, and how they can be satisfied;
- **Appropriate visual display**: choosing an appropriate display according to the context. In distinct situations, different data representations may be more fit, such as using a simple text, a scatter plot, a table, a line graph, a heatmap, or a slope graph, among others.
- **Eliminate clutter**: erasing what does not add informative value to avoid information overload;
- **Focusing the attention**: utilizing preattentive attributes, which are visual features that the subconscious finds most salient and deserving of attention, like orientation, shape, line length, line width, size, added marks, color, and position;
- **Design:** knowing the importance of highlighting important information, eliminating distractions, taking care of accessibility, not complicating, and having a pleasant design;
- **Telling a story**: introducing the context, developing the situation with relevant background, discussing potential, and conclusions.

As technology continues to evolve, the size of datasets also increases, which makes it difficult to simplify and ensure scalability for visual design [29], as it is hard to make relations and find patterns among the vast amount of data. It is important to be careful with the amount of data displayed to avoid overwhelming the user with information. This is also beneficial as many users may lack the time or motivation to learn new software applications [33]. Incorporating functionalities for linking views from multiple interactive panels with different visualizations derived from various dimensions of the same dataset or even different datasets can significantly enhance the usability and effectiveness of dashboards [34]. This allows for the simultaneous identification of objects or groups across different parameter projections, empowering multi-dimensional data exploration. Alternatively, using tabbed layouts or pagination instead of the traditional all-in-one view presents users with limited amounts of information to digest progressively. However, while

multi-page dashboards with tabbed layouts simplify data visualization through information segmentation, users cannot grasp the complete dataset at once. Therefore, tabbed layouts should be used with some caution. Linking views and tabbed layouts are complementary tools in dashboard design: linking views facilitate interactive exploration across different dimensions, while tabbed layouts organize and present information in a structured manner, enhancing usability and clarity in complex datasets. Integrating these features effectively offers users a balanced approach to navigate and comprehend data-rich environments.

It is important to reiterate the importance of using adequate data representations to not confuse the user and provide an accurate notion of the results. This problem is evident in [13]. Here, when dealing with very large datasets, we get significantly more information if the data is visualized with a density plot, resulting in a clear visualization. They show the difference between the Scatter and Density plots in Figure 2.2.

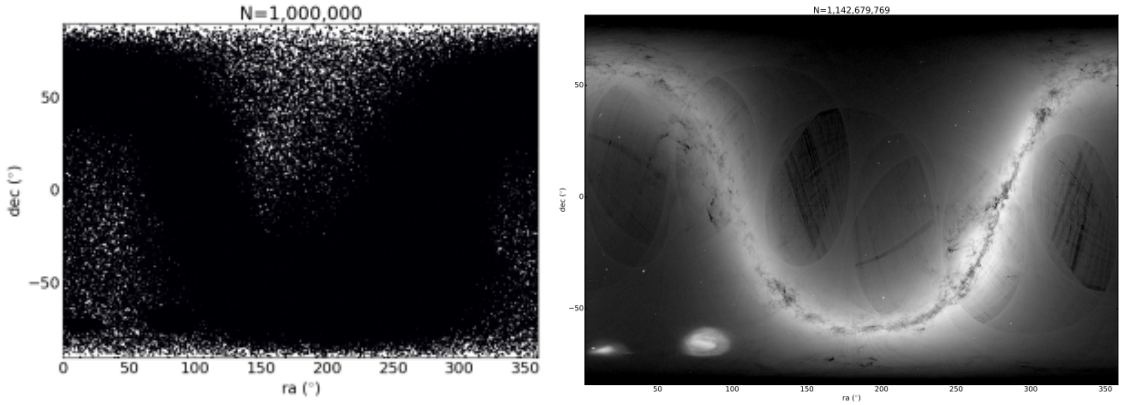



Figure 2.2: Comparison between Scatter and Density plots. Left: Example of a scatter plot, with 1,000,000 sources, hides almost any structure in the data. Right: Example of a density plot with 1,142,679,769 sources, where the logarithm of the density is color-mapped (black is low density and white is high density), preserving data structure from [13].

Additionally, experts [16] emphasize the importance of visual indicators, such as icons or flags, attached to data points on plots. Furthermore, they advocate for expandable and collapsible annotations that provide more comprehensive information about specific data targets. Key considerations include multiple data and chart targets, attachment to data points rather than entire charts, visual snapshots of annotated data changes, and transparent, reusable annotations. These are crucial for analysis, record-keeping, and data discovery.

The transparent and reusable nature of annotations can significantly reduce the user effort involved in visualization systems that offer multiple data views, such as multidimensional visualization systems. Annotations that are transparent in granularity and highlight critical data across data levels can benefit any system with a layered or hierarchical data representation, such as those that support semantic zooming. Moreover, it is imperative to consider issues related to the validity and lifetime of annotations in any dynamic data visualization system.

Data visualizations are associated with truthfulness and objectivity [31]. To maintain these characteristics, practitioners must include data sources and evidence links in their visualizations to provide proof that the data has been represented accurately [6]. Dashboards requiring manual data input may be susceptible to manipulation, leading to inaccurate or misleading information. Researchers emphasize the importance of providing information and explanations about underlying data and the techniques applied to build trust and accountability [56]. The first concept is correlation [32], when analyzing relationships in data, it is important to remember that the graph alone does not tell the whole story, and it is necessary to interpret the results accurately. This is especially true because data-checking and verification are crucial aspects of graph design and can be considered the most important steps in the process [57].

### 2.1.2 Evaluation

The effectiveness of a dashboard goes beyond aesthetics and functionality. It depends on the ability to meet user needs, enhance decision-making, and optimize operational efficiency. Evaluation is a critical process in ensuring that dashboards not only fulfill their intended purposes but also excel in usability and user satisfaction. This section examines methodologies and criteria used to evaluate the effectiveness of dashboards. Additional studies are necessary to comprehend how real-life situations impact the use of dashboards [24].

#### 2.1.2.1 Criteria for evaluation

According to a study on dashboard design [17], a well-designed dashboard should follow certain characteristics that can be used to evaluate its effectiveness. The information on the dashboard should be well-organized and easily recognizable, allowing users to identify what requires their attention immediately. To avoid overwhelming viewers with excessive data, the dashboard should be condensed, primarily using summaries and exceptions to represent large sets of numbers or unusual occurrences. The dashboard should also be customized to suit the specific audience and objectives it is intended for. Finally, the data should be presented using concise and small media formats that communicate the data and its message as clearly and directly as possible.

The process of evaluating dashboards can also be defined as the systematic measurement, collection, analysis, and reporting of data concerning their respective contexts. However, the assessment of dashboards presents numerous challenges, including inadequate data quality, limited data comprehension, wrong analysis, misconstrued interpretation, and ambiguity regarding outcomes. It is imperative to address these challenges to ensure the efficacy of dashboard evaluation practices [6].

Other suggestions for key criteria for dashboard evaluation are classified in [28]:

- **User customization**: prioritizing empowering users for agile responses. Collaborative tools, despite lower priority, enhance user collaboration;

- **Knowledge discovery**: high-priority criteria aid root cause analysis, crucial for incident investigation;
- **Version control**: emphasizing version control for source files, ensuring tracked changes with identity attribution. Any changes made to the source files are tracked, along with the identity of who made the changes, the reasons behind the changes, and references to any problems fixed or enhancements introduced;
- **Information delivery**: aligning the dashboard design with predefined objectives. The contents displayed on the dashboard should be clear and easy to comprehend.
- **Visual design**: advocating for visually appealing and engaging dashboards without overwhelming users, promoting concise and minimalist design to avoid information overload and unnecessary navigation;
- **Metadata and Help**: informing the user about the dashboard and providing instructions for its use. It emphasizes the importance of visual intelligence, which refers to the software's ability to provide better insight by intelligently highlighting relevant areas and values on the dashboard in response to a user's cursor movement;
- **Alerting**: it is a mechanism to give attention to the exceptions, outliers, and data highlights. Alerts by color coding are advised to show levels of threats, such as a target zone in green, a warning zone in yellow, and a trouble zone in red.

As illustrated above, there is a large diversity of possible evaluation criteria proposed by the literature. With this in mind, we compiled a summary of key points, features, and descriptions based on this literature that aims to provide a simple and actionable guide for evaluating a dashboard, on Table 2.1.

Table 2.1: Summary of features and guidelines for evaluating dashboards.

| Key Points | Description |
|---|---|
| Interactivity | • Implementing interactive **point-and-click** features for users to interact with specific data points easily.<br>• Providing options for users to **filter** and sort data based on their preferences.<br>• Integrating **drag-and-drop** functionality for seamless organization and manipulation of data elements.<br>• Offering **highlighting** features to emphasize important insights or trends. Allow users to perform **multi-dimensional analysis** through interactive tools. |
| Visual Design | • Using **shapes and forms** in the visual design to enhance the overall aesthetics and user experience.<br>• Using visual elements such as **colours or annotations** to highlight critical or important stages.<br>• Incorporating **meaningful icons** to quickly recognize different data categories.<br>• Ensuring **consistency** in typography and iconography for a unified visual theme. |
| General Guidelines | • Limiting the amount of displayed data to **prevent overwhelming** users.<br>• Ensuring a **user-friendly design** that facilitates easy navigation and comprehension of the displayed information.<br>• Maintaining a **consistent visual theme** throughout the dashboard for a cohesive and professional look.<br>• Designing the dashboard to be **responsive**, adapting to different screen sizes and devices for a **versatile** user experience.<br>• Providing **user customization** options to tailor the dashboard to individual preferences.<br>• Implementing **clear metadata and help** sections to guide users in understanding the dashboard content. |

#### 2.1.2.2 Tests for evaluation

The method of heuristic evaluation [36] holds a significant position in the field of usability evaluation. It is an informal usability analysis approach in which multiple evaluators assess a design

and provide feedback based on their individual judgment and established usability principles. This method is valued for its practicality and efficiency, particularly in early-stage evaluations where formal techniques may be impractical or unavailable. The nine usability heuristics proposed in [36] are:

- **Simple and natural dialogue**: users should be able to easily understand the current status of data and ongoing processes;
- **Speak the user's language**: using familiar language and concepts to enhance user comprehension;
- **Minimize user memory load**: avoiding requiring users to remember information across different parts of the system;
- **Be consistent**: maintaining consistency in layout, design elements, terminology, and interactions;
- **Provide feedback**: offering immediate feedback to users about their actions;
- **Provide clearly marked exits**: users should be able to easily exit or navigate away from different sections or tasks;
- **Provide shortcuts**: offering shortcuts can enhance efficiency for users who frequently perform specific tasks or actions;
- **Good error messages**: error messages should clearly explain what went wrong, why it happened, and provide guidance on how to correct the issue;
- **Prevent errors**: minimize the occurrence of errors. This includes using constraints and confirmations for critical actions.

When designing a dashboard, it is essential to ensure the presence of clear and comprehensible labels, descriptions, and instructions. Consistency in navigation patterns fosters familiarity and ease of use. Providing immediate feedback through visual cues and notifications on the dashboard keeps users informed about their actions and system status. By integrating these principles into dashboard development, designers can optimize usability, streamline user interactions, and ultimately improve overall user satisfaction and productivity.

In addition to traditional usability evaluation methods such as heuristic evaluation, A/B testing has emerged as a pivotal technique in software systems for assessing and optimizing user interfaces. As the authors [38] describe, A/B testing involves the comparison of two variants (A and B) of a design to determine which performs better with end-users, typically through live experiments. Stakeholders in A/B testing, including experiment contributors, play crucial roles in managing and interpreting test outcomes, thereby facilitating informed decision-making in software development. Integrating A/B testing into the development of a dashboard would enable iterative improvements.

### 2.1.3 Examples

When developing a dashboard, it is useful to review existing tools, as it enables us to learn from both their strengths and their pitfalls. As such, in this section, we will discuss a set of relevant examples.

It is beneficial to have a single page to avoid being overwhelmed, so a good option is to showcase multiple panels on the same page, presenting different perspectives on the data. One can, for instance, show a bigger and more general view and multiple smaller ones giving more detailed perspectives, as presented by [41] and depicted in Figure 2.3.

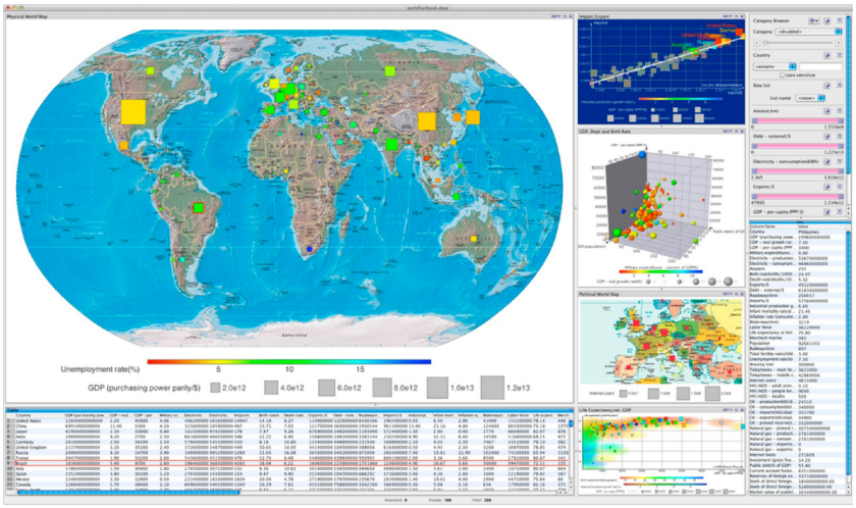


Figure 2.3: A window of the DataWarrior software with various views shows different aspects of the same underlying data table from [41].

Another example is NASA's Eyes on the Solar System[1], a web-based application that visualizes the solar system and various space missions in 3D. The visualization uses real-time data to show the positions of planets, moons, spacecraft, and other celestial bodies. Users can explore space missions, planetary alignments, and historical events using color-coded trajectories, labels, and interactive features [2].

One more example, also from NASA's Ames Research Center, is the open source Open MCT (Mission Control Technologies) platform, in Figure 2.4.

[1] The NASA's Eyes on the Solar System can be accessed at https://eyes.nasa.gov/apps/solar-system/#home

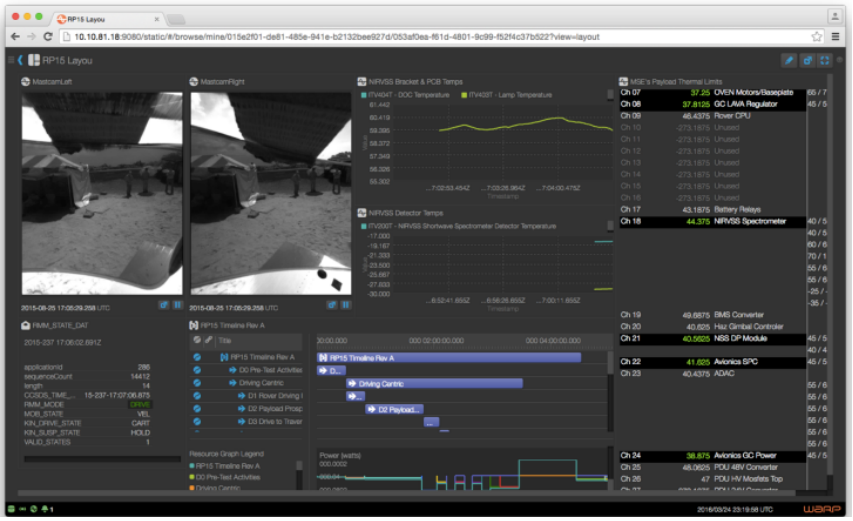


Figure 2.4: NASA's OpenMCT display combining timeline, activity, telemetry, and image object types from [48].

This example is very effective because it provides all the necessary information within a single-page view. The information is presented clearly, and the colors are not overwhelming. Additionally, objects from various domains can be assembled into layouts, and multiple data sources can be integrated and displayed.

There are a few additional astronomy examples of tools that are well-suited for specific tasks and visualizations. These include Firefly [23], ESASky [9], FitsMap [26] Tera-scale [25], and ALADIN [11]. The FitsMaps [26] platform is designed to open FITS files using modern web technologies, allowing users to interactively explore FITS files directly in their web browser. This approach eradicates the need for specialized software installations and ensures a high level of accessibility. An example provided in the article is depicted in Figure 2.5 and is comprised of HTML, CSS, and JavaScript. Despite offering numerous image interaction features, it lacks comprehensive functionality and description for the overall file.

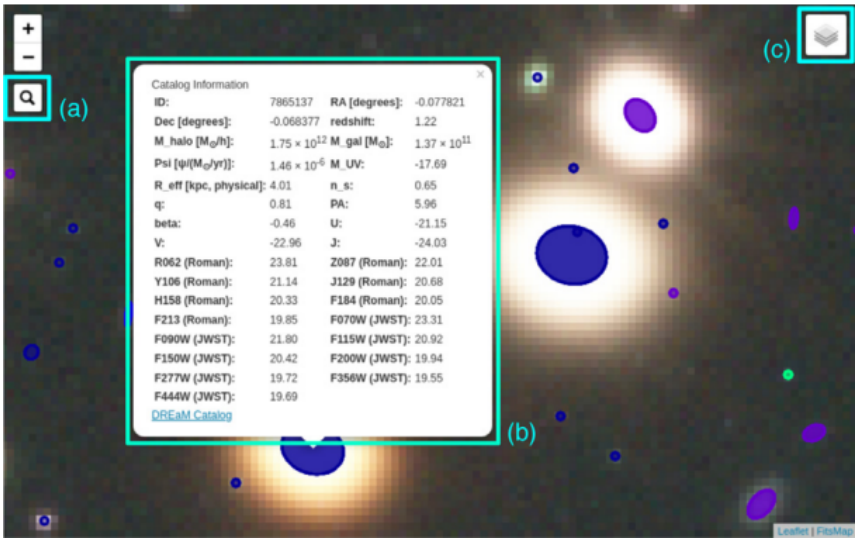



Figure 2.5: An example of the FitsMap interface. (a) The search function button searches catalogs by id. (b) A marker pop-up displaying catalog data associated with the indicated source. (c) The Leaflet layer control allows users to switch between display images and catalog overlays from [26].

SAOImageDS9 is an example of an application that can visualize simple astronomical images stored in the FITS file format. While AOT files are also FITS files, this tool is unable to deal with the complexity of that format, highlighting the need for a purpose-built application. A screenshot of this application can be seen in Figure 2.6.

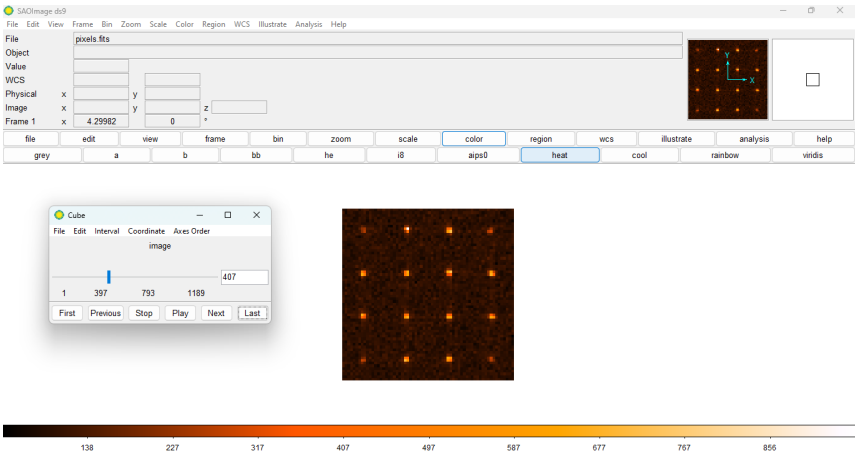



Figure 2.6: A screenshot of SAOImageDS9 framework displaying a Shack-Hartmann wavefront sensor.

It is possible to see a main image display window, a button menu panel, a display magnifier, a pan and zoom reference image, and a color bar. A color table graph window can be opened by clicking on the color bar. Although all the main features are for the researchers to use, the tool could be more visually appealing and intuitive. One major issue with SAOImageDS9 is that most functionalities are hidden under various sub-menus, making it difficult for users to quickly

locate and utilize the needed features. Menu categorization is sometimes incorrect; for example, the "scale" menu includes not only scaling options but also intervals, which can confuse users.

The SAOImageDS9 menus are primarily text-based, lacking icons or visual cues for intuitive user guidance. Furthermore, the interface does not effectively utilize shapes and forms to enhance aesthetics. The incorporation of icons can significantly enhance the user experience by providing immediate visual recognition of functions, reducing the cognitive load required to navigate through text-heavy menus. Another issue is that the same tasks can be carried out from completely different menus. This redundancy can lead to confusion, as users might not know which menu path is the most efficient or appropriate for their task. It also increases the learning curve, as users must remember multiple pathways to perform the same function.

It does not clearly distinguish between basic and advanced features, which can overwhelm users. A more effective design would segregate these features, allowing novice users to focus on essential functionalities while still providing access to advanced tools for experienced users. Also, there is no textual description of what each feature does within the tool. Users are left to either experiment with features to understand their purpose or refer to external documentation. This lack of in-app guidance can hinder productivity and frustrate users.

## 2.2 From open-source to FAIR

The ubiquity of data and the availability of visualization technologies to the public have significantly expanded the adoption of dashboards across various domains [42].

According to [10], the current era can be characterized as a "visualization of culture". Visualization practitioners argue that visualizations play a crucial role in enhancing the understanding of data by making it more accessible and transparent [18], [59]. Visual representation of data is often used to comprehend its substance, motivating users to seek advice. A case study on the successful dashboard deployment in hospital management [53] identified two key aspects for promoting shared understanding and employee acceptance: shared context (dashboards presenting specific details alongside an overview of organizational strategic objectives and targets) and transparency (ensuring that 90 percent of dashboards are accessible to every employee).

Tools for accessing data reservoirs are heterogeneous and often limited to graphical user interfaces or websites, reproducibility is a cornerstone of research, and data availability is crucial for achieving it [14]. Data sharing is not only vital for promoting new discoveries, but also for reviewing and validating published results. Observatories, including the ESO/A observatory, actively share full datasets. As it is said in [19]: “Individual teams and small groups often share their data via their custom websites. These services do not follow any particular standard and can be widely varied in the type and amount of data shared.”

Open source is desirable as it allows transparency, reuse, and exploitation at a low cost in other contexts. Indirectly, the sharing of code has the potential to amplify good practices and increase the speed and agility of code development.

The FAIR principles [55] extend the open-source paradigm to data, serving as a guideline to enhance data reusability. These principles stand for Findability, Accessibility, Interoperability, and Reusability, providing data producers and publishers with a framework for navigating obstacles and maximizing the value gained from formal scholarly digital publishing. By following the FAIR principles, researchers who want to share, get credit, and reuse each other's data and interpretations can overcome many of the obstacles that hinder the reusability of data. However, not all datasets or data types can be submitted to special-purpose repositories that adhere to these principles. Nonetheless, these datasets are equally important in terms of integrative research, reproducibility, and general reuse. The elements of the FAIR Principles are related, but independent and separable, allowing publishers to prioritize certain principles based on their needs.

In this dissertation, the dashboard to be developed (code) is intimately connected to the data to be visualized, following both open-source and FAIR principles.

## 2.3 Software Review

In this section, we present an overview of software suitable for tool visualizations, particularly for dashboards. We focus on compatibility with Python, open-source availability, and integration capabilities. This review aims to identify the best options for our specific needs, including their advantages and disadvantages.

First, we compare various software options based on key characteristics summarized in Table 2.2.

Table 2.2: Comparison of Software in Terms of Python Support and Open Source Status [46], [50].

| **Software** | **Python Support** | **Open Source** | **Integrations** | **Ease of Use** | **Community Support** |
|---|---|---|---|---|---|
| Google Looker Studio | Yes | No | Google Cloud | Very easy | Growing |
| Grafana | Yes | Yes | Multiple | Easy | Strong |
| Kibana | Limited | Yes | Elastic Stack | Easy | Strong |
| Metabase | Limited | Yes | Multiple | Very easy | Growing |
| Power BI | Limited | No | Microsoft Ecosystem | Little to easy | Strong |
| QlikView | Yes | No | Multiple | Very easy | Strong |
| Redash | Yes | Yes | Multiple | Very easy | Growing |
| Salesforce | Limited | No | Salesforce Ecosystem | Easy to very easy | Strong |
| SAP Analytics Cloud | Limited | No | SAP Ecosystem | Easy to very easy | Growing |
| Streamlit | Yes | Yes | Python Ecosystem | Very easy | Growing |
| Dash by Plotly | Yes | Yes | Python Ecosystem | Easy to very easy | Strong |

Although Kibana is open-source and a good option for big data, it only allows data stored in Elastic Search, so it is not a good fit. From the table above, one can select the ones that have Python support, are open-source, and have a good community: Grafana, Metabase, Redash, Streamlit, and Dash.

**Grafana** [22] is an analytics and monitoring solution. It provides an effective way to work with any metrics, enabling one to query, visualize, alert, and understand the data. One of the remarkable features of Grafana is its ability to seamlessly connect with various data sources. It provides a way to query, visualize, alert, and understand data from various sources, including Graphite, Prometheus, InfluxDB, Elasticsearch, MySQL, and PostgreSQL. Users can develop custom plugins for additional flexibility

**Metabase** [27] is a business intelligence tool that simplifies data understanding through metadata storage. It allows users to generate tables and graphs without coding skills and includes features for browsing and filtering data, setting up alerts, and applying color-coded graphs. Metabase provides a platform to set up alerts, which are useful in controlling data. Team members in an organization can receive notifications when data is out of control. The tool can be downloaded and modified to accommodate newer updates.

**Redash** [43] is an open-source collaboration tool for SQL-based data analysis built with Python, Flask, PostgreSQL, and ReactJS. The web-based tool allows users to add, delete, and alter data directly from the tool itself and visualize data using drag-and-drop features. It integrates with databases like MySQL, PostgreSQL, Redshift, and Google BigQuery, providing interactive dashboards and reports for collaborative data analysis.

**Streamlit** [15] is a user-friendly programming interface for creating data visualizations with Python. Streamlit comes with predefined widgets for entering parameters and presenting different data types, such as text, images, and audio. These widgets are defined in JavaScript but have an easy-to-use interface in Python. Streamlit's saved cache optimizes long-running operations, and it supports deployment on cloud services and containerization.

**Dash** by Plotly, [39] is a Python framework for developing web-based analytical applications. It uses Plotly.js and React.js to create interactive data apps with custom user interfaces. One of Dash's strengths lies in its stateless nature. This design significantly enhances application robustness by allowing multiple callbacks to access the same data simultaneously without conflicts and maintaining uninterrupted service, even if individual processes fail. Dash supports various visualizations and input components, and it simplifies deployment on cloud platforms and user servers.

Table 2.3 culminates in a review of the highlighted frameworks. It presents an overview of the advantages and disadvantages associated with each option.

Table 2.3: Comparison of advantages and disadvantages of the highlighted frameworks.

| **Software** | **Advantages** | **Disadvantages** |
|---|---|---|
| Grafana | Wide data sources, extensive customization, alerting, and time series visualizations. | Learning curve for plugins, limited self-management, limited organization and design. |
| Metabase | Easy to use, filtering, advanced querying, color-coded visualizations, alerting, collaboration. | Single SQL data source, crashes reported, limited chart types. |
| Redash | Collaboration, wide data sources, interactive dashboards, data editing, drag-and-drop visualizations. | Not time-efficient with Big Data, unmanageable with large projects, and limited visualization options. |
| Streamlit | User-friendly, pre-built components, real-time updates, data type compatibility, caching, public or cloud deployment. | Primarily for small/medium apps, limited customization, performance with complex calculations, not production-scalable. |
| Dash | Easy to use, supports complex visualizations, integrates well with Plotly, supports real-time updates, strong community support. | Limited customization for appearance, more coding for advanced features, potential performance issues with large datasets. |

From this comparison, we see that each software has its unique strengths and limitations. For instance, Grafana excels in handling multiple data sources and providing extensive customization, but it has a steeper learning curve. Metabase is user-friendly and excellent for collaboration, though it supports fewer data sources. Redash is powerful for SQL-based analysis and collaboration but struggles with big data efficiency. Streamlit and Dash, both Python-based, are ideal for custom applications with real-time updates, though they vary in scalability and ease of advanced customizations.

By understanding these advantages and disadvantages, we can make a more informed decision on which software best fits our needs. This detailed analysis will be crucial in the methodology section, where we will select the most suitable software for our dashboard implementation.

## 2.4 Key takeaways

Dashboards are very important for efficient and comprehensive data exploration. The development of visual designs is helpful in many different contexts due to their various capabilities and functionalities. However, dashboard designers continue to struggle with one-size-fits-all dashboarding tools, which often fail to reflect the diversity of user needs and goals. Users' needs require greater flexibility, customization, adaptability, and deeper analytics [42]. By leveraging these research definitions, various guidelines, and diverse evaluation methods, it becomes feasible to develop an optimal solution. This can become even more streamlined if we use the FAIR data principles, by potentiating collaboration with other researchers.

As presented, there is a vast amount of software available for elaborating on the dashboard, and the comparison between them will make the process of choosing easier.

# Chapter 3

# Methodology

In this chapter, we provide a detailed characterization of the project. This includes a detailed discussion of the project requirements in Section 3.1, followed by an examination of the tool design in Section 3.2. Additionally, Section 3.3 offers an explanation of the dashboard implementation, with a thorough description of the exploratory and analysis components. By the conclusion of this chapter, readers should have a clear understanding of the dashboard development and overall framework.

## 3.1 Requirements

Establishing clear and comprehensive requirements is essential for developing an effective dashboard. By defining these requirements according to our specific needs and objectives, we ensure that the research standards are met. This is crucial for the development process of this tool.

One of the requirements of the dashboard is the need to support telemetry data from the European Southern Observatory (ESO) [1]. The dashboard must be capable of processing these file types and their respective datasets. Handling data from multiple advanced adaptive optics (AO) systems, such as GALACSI [45], CIAO [30], NAOMI [5], and ERIS [40], presents a significant challenge due to the differing AO configurations. Therefore, the dashboard must be capable of adapting these variations while remaining functional and efficient.

The requirements reflect features, guidelines, and considerations that need to be considered when designing and implementing the dashboard. They can be categorized as functional, non-functional, user, and technical requirements. Functional requirements define the system's expected actions, including its specific features and functions, whereas nonfunctional requirements detail the system's performance and overall quality. User requirements describe user needs and wants from the system, while technical requirements specify the technical parameters and constraints that the system must meet to satisfy those user requirements. Applying it to our dashboard needs:

[1] For the user to extract these specific files, it can directly access the ESO Science Archive Facility https://archive.eso.org/eso/eso_archive_main.html, and insert the program IDs 60.A-9278(#), with # = B for GALACSI, C for CIAO, D for NAOMI and E for ERIS.

**Functional Requirements**

- Data Ingestion and Processing: the system must be able to read and process AO telemetry data files in the AOT format, as well as support the ingestion of data from local files;
- Data Visualization: the dashboard must be able to visualize telemetry data in different AO configurations and allow interactive exploration of data by zooming, time-series plots, and selecting key points to evaluate further;
- User Interaction: features such as interacting with specific data points, filtering and sorting based on different criteria, and organizing and manipulating data elements are essential as it is easier for the user to analyze the data;
- Analysis Tools: statistical analysis tools like mean, maximum, and minimum values, as well as multi-dimensional analysis and correlation between different data parameters, should be available to the user.

**Non-functional Requirements**

- Performance: the system must efficiently handle large datasets and optimize performance to ensure responsive interactions and visualizations while minimizing load times and processing delays;
- Usability: the dashboard must have an intuitive and user-friendly interface suitable for both technical and non-technical users, with clear sections for different functionalities;
- Scalability: the system should be scalable to accommodate increasing amounts of data, multiple files, and concurrent users using the dashboard without performance degradation;
- Maintainability: the dashboard must be well-documented and maintainable, adhering to the best clean and reusable code practices.

**User Requirements**

- Image Display: each image should display its name and associated data. Users should be able to view data cubes in an interactive 2D image format, allowing for intuitive exploration and analysis;
- Value Inspection: users should be able to inspect values at specific points (e.g., hover to display values);
- Images Options: Data manipulation should be available in multiple ways, for example, different scales, intervals, colormaps, and more;
- Interactivity: the ability to move throughout the dataset, by zooming, panning, and moving around the image;

- Statistics: display statistics from the file, like minimum, maximum, and average values;
- Time-Varying Data: users should explore time-varying data frame-by-frame or as an averaged image, with options for manual selection or automatic play-through;
- Values Display: the values should be displayed in a straightforward list format within the object's properties (e.g., Altitudes and Shutter Type).

**Technical Requirements**

- Technology Stack: the software needs to be compatible with Python, as it is the primary programming language, compatible with the *aotpy* package for handling AO telemetry data;
- System Design: it is beneficial to utilize a modular design [49] approach, meaning subdividing the system into smaller independent modules for data ingestion, processing, visualization, and user interaction. This facilitates scalability and maintainability by adding new features without impacting existing ones, enabling separate development, testing, and debugging of modules.
- Architecture: has to be able to run on a server to make it public;
- Development Tools: use version control, for example, Git, to keep the tracking updated, maintain the system, ensuring continuous improvement;
- FAIR Data Principles: enables data finding, accessing, interoperability, and reusability. We need to provide metadata and documentation for data discovery and reuse;
- Open Source: the dashboard should be open-source, with the code publicly available on a repository platform like GitHub.

The requirements specified here will be used from now on to guide our decision-making and to evaluate the suitability of the tool being developed.

## 3.2 Tool user experience design

We designed this dashboard through an iterative process that considers user feedback regarding its experience and visual design. Specifically, we gathered feedback by sharing design mockups with AO experts, modified the design based on their concerns, and repeated the process until they considered the tool satisfactory. The design process began with thorough planning to ensure the tool met the researcher's needs while maintaining a user-friendly interface.

### 3.2.1 User Flow

The dashboard layout is carefully crafted to provide a comprehensive view of the AO data, incorporating system parameters, sensor data, and graphical representations. The design emphasizes consistency and simplicity to optimize user experience. According to our needs, the dashboard will have five pages. A user flow for the dashboard is represented in Figure 3.1.

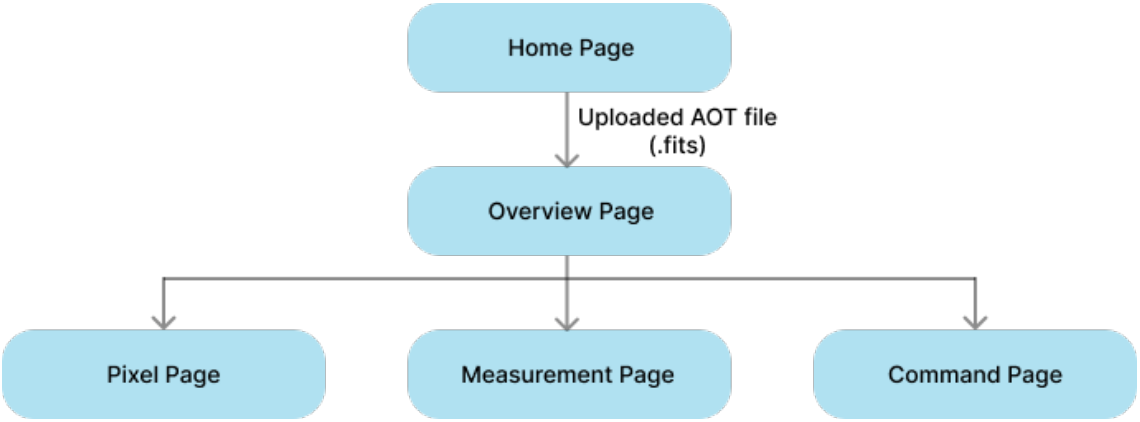



Figure 3.1: User Flow for the dashboard.

To help with the understanding of the user flow, we introduce in more detail the pages containing:

- **Home Page**: serves as the entry point for users to upload their AOT files. After uploading, the page will display the file name to confirm the correct upload or provide a warning message if the file is incompatible.
- **Overview Page**: provides a summary of the operational parameters of the AO system, including system name, mode, and telescope used. Users will be able to navigate the system hierarchy to explore specific sources, wavefront sensors, detectors, and associated measurements and controls.
- **Pixel Page**: displays detailed information about the wavefront sensors and captured detector data. This includes the ability to see the available detectors, sensor properties, light source (such as a Natural Guide Star), associated statistics, and an object section to view the objects associated with the wavefront sensor. Detectors will be represented with 2D visualization over frames (rasterized image), including different options for intervals, colors, scales, and being able to select different features, such as a flat field filter. It will also include visual representations of pixel index over frames (vectorized image) and intensity per pixel through frames represented on line plots to check the mean values of the detector along frames.
- **Measurement Page**: similar to the pixel sections, it displays the different measurements of the wavefront sensor and key information. It includes graphical representations of various measurements of the wavefront sensor, along with relevant statistics. Graphics are similar

to the ones on the pixel page but with different titles and measurements, as well as the associated objects. With the separation of different dimensions $(x, y)$, users should be able to quickly identify trends. The different dimensions will be displayed side by side, for the rasterized images, and for the line plots, distinct lines for clear differentiation.

- **Command Page**: provides a streamlined view similar to the previously explained pages, with information about all the available commands of the control loops of the wavefront correctors but with fewer parameters to manipulate, as it is not needed. In the objects section, it displays objects associated with the wavefront corrector.

### 3.2.2 Visual design

Detailed mockups are crucial for building effective user interfaces, focusing on functionality and visual appeal. Figma [1], a design tool known for its powerful vector graphics and prototyping capabilities, was chosen for its simplicity and efficiency in creating detailed mockups and prototypes. The mockups are seen in Figures 3.3, 3.4, and 3.5 for the different pages.

Visual elements were selected for their thematic relevance and functionality according to the context. The decision to use dark tones reflects the modern dark mode trend, in which software uses a darker design in order to reduce eye strain. The colors were chosen due to personal preference and because they resemble space, with blue dominating the ESO website; the predominant colors in our dashboard are shown in Figure 3.2.

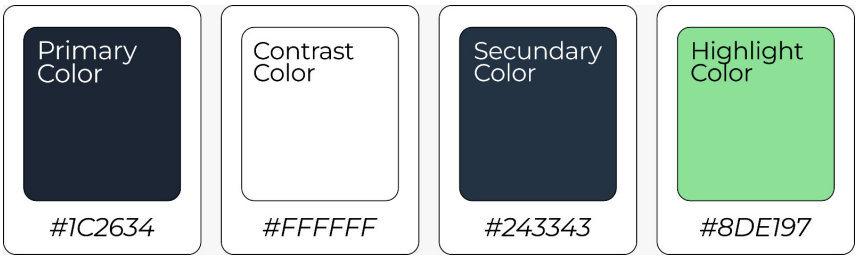



Figure 3.2: Primary Colors used for the dashboard.

A cohesive color scheme is used throughout to create a unified aesthetic, with colors selected for their visual appeal and their ability to highlight important information and differentiate various data points. We decided on a top navigation bar that gives the dashboard a more intuitive appearance and improves usability by providing quick access to different pages, enabling seamless navigation. The chosen font style and size optimize readability and make data interpretation easier, specifically by using a lighter font in contrast to the dark background.

The landing page (Home page) has a minimalist design with a single box for users to submit AOT files, ensuring a distraction-free interface for the primary task. The overview page consolidates the most relevant information into a single, easily navigable screen, which is horizontally aligned for improved readability and quick comparison. Users can easily find specific system information, such as file start date and related data on the top of the page. This layout also helps

the understanding of the AO system's relationships with a hierarchy section of the system at the bottom of the page displaying it.

The detailed pages, pixel, measurement, and command, are designed similarly to create a user-friendly experience. The standardized style facilitates intuitive navigation, especially for new users seeking main commands quickly. Each is dedicated to specific aspects, such as the wavefront sensor detector (pixel page), wavefront sensor measurements (measurement page), and wavefront corrector loop commands (command page). Key information is prioritized by being placed on the left side for quick reference and to align with typical reading patterns. This ensures immediate access to specific details upon entry, such as the uid (unique identifier), and the statistics.

Each page has clear, rectangular divisions for efficient navigation and locating specific information. On all three pages, there are four distinct sections: 1) the key information (including statistics), 2) the rasterized image (dynamic 2D image) white slider and appropriate white marks for different values within a pre-defined range, 3) the vectorized image, and 4) the line plot. These sections maintain a clean, organized layout and prevent information overload. Visual elements such as charts and graphs that require further inspection are on the right and bottom of the page. Since the charts are synchronized, i.e., user actions in one affect the other, they are placed adjacently, although in clearly distinct sections. For the rasterized image that allows for manipulation, such as changing color, type of scale, or interval, there are drop-down menus available above the image to allow the user to select the desired value for the manipulation. Drop-downs are especially efficient since the user only sees the options when interacting with that element, which reduces the amount of visual clutter. In contrast, for inputs that do not have as many options, mainly binary options, checkboxes were used, although originally press-downs were chosen in the mockups, shown in Figure 3.5 (the flat_field, dark, sky_background), they were not utilized as they were not intuitive. The line plots will utilize various colors to distinguish between different values wanted to see and points. We present some mockups, specifically the Home page in Figure 3.3, the Overview page in Figure 3.4, and one of the similar detailed pages, the pixel page, in Figure 3.5

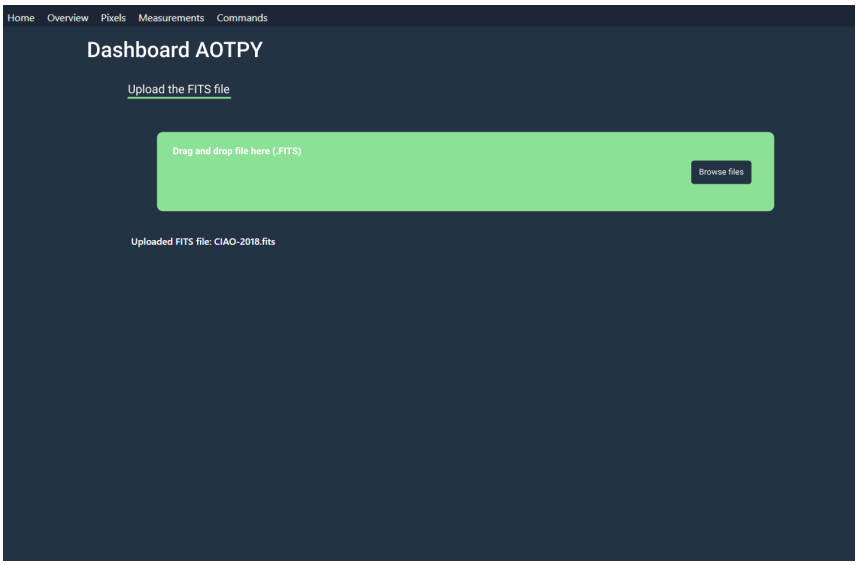



Figure 3.3: Mockup for the Home Page.

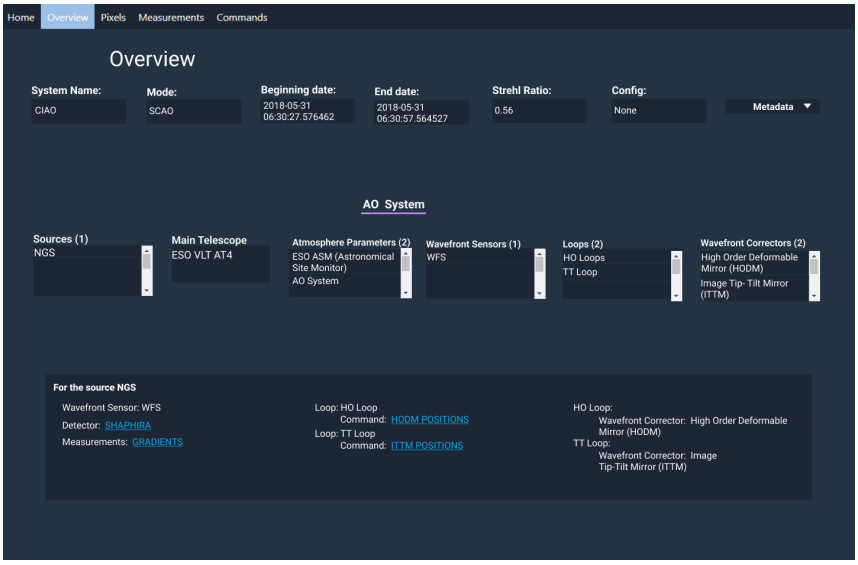



Figure 3.4: Mockup for the Overview Page.

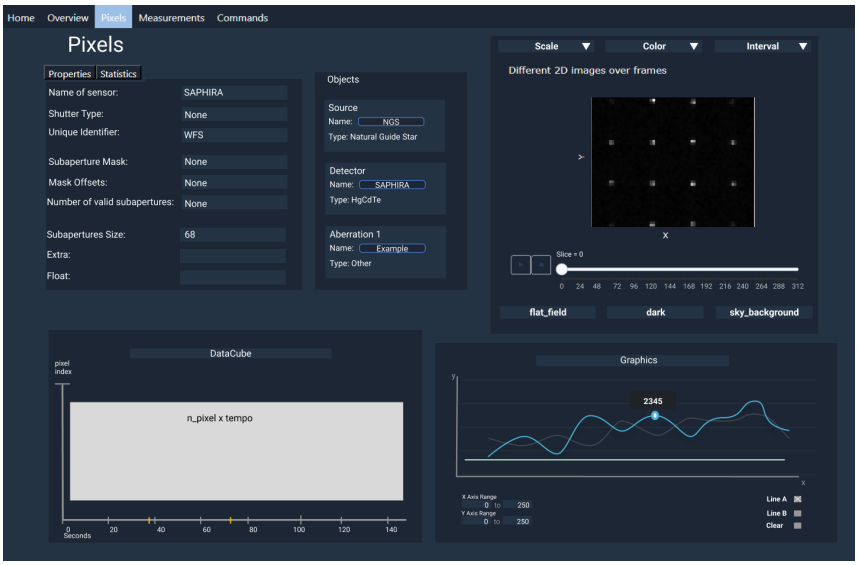



Figure 3.5: Mockup for the Pixel Page.

### 3.2.3 Interaction design

Each page includes a navigation bar with a link to all the pages shown in the previous dashboard user flow. The text of the Navbar changes color when hovered in order to indicate to the user that they are interactable. All pages will have dropdowns to select different objects within their availability if the file contains multiple objects.

For the overview page, users will be able to explore the system hierarchy by first selecting a specific source, appearing its corresponding wavefront sensors, with its detector and measurement associated with that source, and the wavefront corrector associated with the wavefront sensor, with the loops (and its commands) associated, although it is possible to see the offload and control loops on the detailed part of the AO system, only the control loops are presented on this hierarchical form, as they are the only relevant ones. By clicking on the detector, measurements, or loops, on the hierarchy structure, the user can be redirected to the corresponding pages.

On the pixel page, there will be a button, to display the section dedicated to statistics of the image and selected points on the rasterized image (2D image). Detectors will be represented in rasterized and vectorized images, although the first will have different drop-down options for different intervals, colors, and scales. Additionally, the checkboxes for features such as flat field, can be selected and unselected. By clicking on a point in the vectorized image, its frame will be displayed on the rasterized image. Then, selecting a point within that frame (in the rasterized image) will show its intensity through frames on the line plot. In the object section, users will see the objects associated and be able to click on the associated measurement to be redirected to the measurement page.

As stated, the measurement page will be very similar to the pixel page but without checkboxes, since their functions are specific to the page's context. If the measurement does not have a mask offset, a text message will inform the user that it will not be available for the rasterized image due

to the lack of the required parameter (instead of the image and without the slider). For the object section, users will see the associated objects and be able to click on the associated detector to be redirected to the pixel page.

Lastly, the command page, being the simplest of all three, will not have a button for the statistics, as they don't exist on this page. If the command does not have valid actuators, a text message will inform the user that it will not be available for the rasterized image due to the lack of the required parameter(instead of the image and the slider). The object section will display the objects associated with the wavefront corrector, which can be redirected to the pixel and measurement page with a click on the associated wavefront sensor detector and wavefront sensor measurement.

## 3.3 Dashboard Implementation

### 3.3.1 Software architecture

There are three primary architectural structures: Client-side, Server-side, and Client-Server. Choosing the right architecture is critical for optimizing performance and ensuring a smooth user experience.

- **Client-side** refers to everything in a web application that is displayed or takes place on the client side (end user device). This approach, often used in web applications relying on JavaScript, provides quick responses, presents data dynamically, and reduces server load but may expose security risks due to client-side data handling.
- **Server-side** centralizes processing and data storage on dedicated servers. It's ideal for applications needing intensive data processing, secure data handling, and complex business logic. While it ensures robust security and efficient resource management, it may lead to slower response times if not optimized.
- **Client-Server**, integrates aspects of both client-side and server-side approaches. It balances client-side responsiveness with server-side control, scalability, and security.

Based on the requirements for enhanced visualization and efficient data handling for this dashboard, the chosen architecture is Client-Server. This architecture combines the responsiveness of client-side operations with the robustness of server-side processing. It allows for interactive and dynamic user interfaces with real-time updates while ensuring scalability on the server side. We also prefer a stateless framework, which is beneficial because it simplifies the management of user interactions and scales more easily. In a stateless architecture, each request from a client to the server is treated independently, without retaining any session information between requests. This reduces the server's memory load, improves scalability, and simplifies the design and deployment of distributed systems.

### 3.3.2 Technology Stack

When designing a dashboard, besides the importance of the type of design, one must keep in mind the software needed to make the dashboard come to life. These requirements include assessing the ease of operability of the tool, the skills needed to maintain it, the capacity of the tool, its ability to handle multiple processes simultaneously, and whether the tool's functionality helps achieve the desired outcome. Upon careful evaluation of the options specified in Section 2.3, it seems that Grafana and Streamlit are good choices for data visualization. However, it can be challenging the manipulation of layout, charts, and images to achieve the desired outcome. Conversely, while Metabase and Redash are suitable for business intelligence and simpler tasks, they may not be optimal for complex projects and necessitate the acquisition of the premium version for additional features at a cost. Dash, albeit requiring development from scratch, leveraging the guidelines and recommendations presented in the state of the art can lead to superior results. Its robust Python integration, extensive customization options, strong community support, stateless framework, and capacity to effectively handle complex visualizations within a Client-Server architecture are noteworthy. Before finalizing the decision, preliminary tests were conducted with Dash to validate its functionality and performance. These tests included basic operations, such as opening and reading the name of an object from the .FITS files, as well as graphical rendering, confirming Dash's suitability and efficacy.

This framework fulfills the architectural requirements stated above, and, given that it is Python-based, it also allows us to leverage the existing functionalities in *aotpy*. Building this tool on top of *aotpy* enables us to benefit from the package's system-agnostic approach, allowing for compatibility with a wide range of data-producing systems.

Dash leverages both frontend and backend components to facilitate dynamic data visualization and interaction via browsers. This combination ensures scalability so that multiple users can interact with the dashboard independently and simultaneously. Dash is a stateless framework, as desired, so the user's browser acts as a client, and the server handles their requests in parallel. This reduces the load on the client side and ensures consistency. Each client operates independently and receives personalized responses based on their interactions, improving efficiency.

The client-server communication on Dash is facilitated through callbacks. These functions, housed within the app, update the user interface based on interactions. A more detailed explanation about callbacks can be found in section 3.3.3.1. This interaction occurs in a series of steps as follows:

- **User Interaction**: when a user interacts with the dashboard, for example, by clicking on a button or selecting a dropdown, the client sends a request to the server;
- **Server Processing**: the server processes this request using the defined callback functions of Dash in Python, performing necessary computations or data manipulations;
- **Response to Client**: the server then returns the updated information to the client, which updates the display accordingly.

The client side is a web application that involves HTML, CSS, and JavaScript (plotly.js, react.js, with react being already built on Dash) to render the user interface directly in users' browsers. Dash also has `dash_html_components` for HTML components incorporated and Graph class from `dash_core_components` to easily display graphics.

On the backend, Python code manages backend processing, data management, and logic with the Dash callbacks. This architecture ensures scalability to support multiple concurrent users interacting independently with the dashboard. To make the dashboard publicly accessible, it needs to be deployed in a server environment, which can be done using, for instance, Render [3], which is a cloud application that allows for application deployment. To summarize, we present the Figure 3.6.

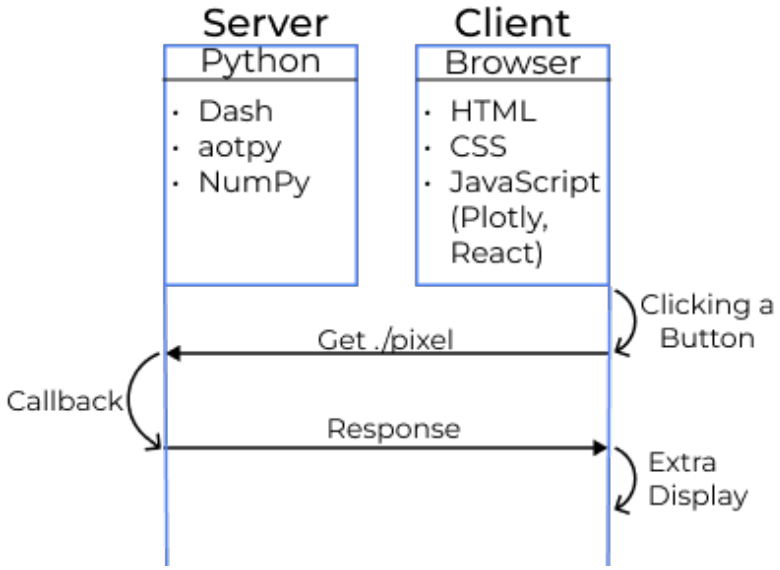



Figure 3.6: Simplified diagram of the architecture used to build the dashboard, with a practical example of what happens when the user clicks a button.

### 3.3.3 Implementation details

#### 3.3.3.1 Callbacks

By definition, callbacks are functions that can be passed as an argument to another function and be called by it. In the context of Dash, these are functions within the app that are called when a user event occurs, such as a button click and trigger a change in the app, with the use of, for example, a Python function. This behavior allows for an interactive experience for the user.

An "input" in Dash is any component property based on user interaction, such as the value of a dropdown, the content of a text box, or the state of a button. An "output" is any component property that can be updated based on a callback function, such as the contents of a graph, or the style of a component. These callbacks connect input components and output components. A key constraint of Dash is that any output can have multiple input components. However, only a single callback can be defined for a specific output to avoid update conflicts.

One challenge encountered was the need to preserve information within the same page refreshes. This was particularly important due to Dash being stateless, which means it does not retain memory or knowledge of previous calls or requests. To address this, we used the `dcc.Store` component to locally store information. `dcc.Store` is a versatile component that allows the data to be saved on the client side. This component can store data in three different locations: memory (default), local storage, or session storage. Memory storage is short-lived and will disappear when the user closes the browser tab. Local storage persists even when the browser is closed and reopened. Session storage persists as long as the browser tab is open but is cleared when the tab is closed. By using this solution, data can be stored locally and activated with URLs, ensuring continuity of information even if the page is refreshed.

Another issue encountered was the necessity for an effective method to store data across different pages, also due to Dash being stateless. Given the focus on the visualization and the time constraint, we opted to use the *Pickling* method (also called the Pickle method) in Python. This method allows the data to be saved as an object, thereby maintaining the program's state within the session. The pickle module implements binary protocols for serializing and de-serializing a Python object structure. Pickling involves converting a Python object hierarchy into a byte stream, and unpickling is the reverse process, converting the byte stream back into an object hierarchy. Consequently, the data does not need to be reprocessed each time it is accessed, significantly improving efficiency and performance.

#### 3.3.3.2 Data Flow

Implementing the dashboard involves several essential stages, each employing diverse technologies and methodologies to ensure the efficient processing and visualization of Adaptive Optics (AO) telemetry data. The process is delineated as shown in Figure 3.7 and explained below.

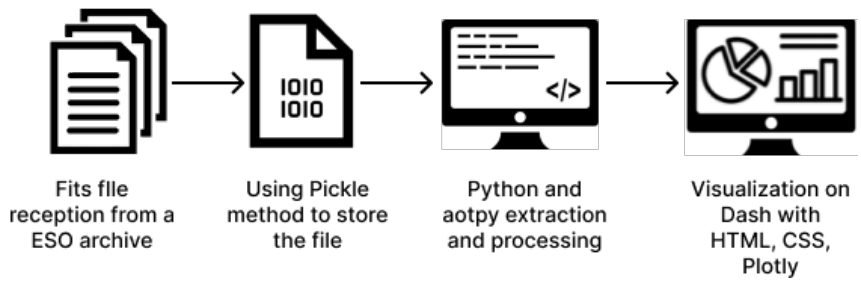



Figure 3.7: Simplified diagram of the data flow in the dashboard.

To demonstrate the implementation process of our dashboard platform, we have included the following workflow diagram 3.7. The process begins with the reception of AOT FITS files from the ESO observatory. These files are then preserved and transferred using Pickle files to ensure data integrity across various application sections. Subsequently, we utilize Python to extract and process the data, making use of a variety of packages, including *aotpy* and *Numpy* for data manipulation and visualization. The dashboard's development primarily involves using HTML, CSS,

and *Plotly* within the Dash framework. This framework incorporates interactivity and enables dynamic data updates through its stateless nature and callback functions.

#### 3.3.3.3 Exploration and Analysis

This section explores the detailed techniques used for image manipulation and visualization in developing our dashboard. It also explores the implementation of statistical analyses and interactive features within the dashboard. The main focus is on improving the interpretability and usefulness of data from adaptive optics telemetry.

The manipulation of the 2D image visualization was carried out as required, providing insights into pixel intensity variations across frames using different scales, colors, and intervals. Using different scales is beneficial because it allows for a more flexible and detailed image data analysis. It helps to highlight various features and patterns that may not be visible under a single scaling method, thereby improving the interpretability and accuracy of the visualizations. For the scales, the *Numpy* library already had built-in options, except for Histogram Equalization. Histogram equalization is an image processing method used to improve contrast in the images by utilizing the image's histogram. We first need to convert the image to a NumPy array to compute the histogram using *NumPy* library. We used 256 bins, the standard, that correspond to the intensity levels of the image and then activated density to normalize the image. Normalization removes noise and brings the image into a range of intensity values that follow a normal distribution. According to the article [52], the use of normalization helps in faster convergence of datasets, achieving better results. Next, we employed the Compute Cumulative Distribution Function (CDF) and used the interpolate function to enhance the image's contrast by redistributing the pixel intensities. Finally, we restored the image to its original size.

The required color mapping improves graphical representations, interpretability, and visual attractiveness. Different colormaps can effectively accentuate various data features, enabling users to distinguish and analyze the underlying patterns. This feature is applied in the function `px.imshow()` of *Plotly*. For the intervals, we used the *astropy* library, a community Python library for astronomy, which has all the intervals needed available. Using different intervals is crucial because it allows the visualization to adapt to various data distributions and characteristics, enhancing the accuracy and clarity of the displayed images. The MinMax interval stretches the data between the minimum and maximum values, ensuring that the full range of data values is displayed. ZScale allows for the scale to adjust automatically based on the data distribution, which often results in a more accurate visual representation for images with a wide range of intensity values; we used the default values provided by the referred library for this interval. The percentile interval method adjusts data visualization by focusing on a specific range of data values as defined by a given percentile. This technique helps in managing outliers, drastically different values and improving the visibility of the main data features. In this case, we choose the 30th percentile, a common value, meaning that the interval will exclude the bottom 30% and the top 30% of the data, focusing on the middle 40%.

We also use dark, sky_background, and flat_field data of `aotpy` from the class Detector of the Wavefront Sensor on the pixel page, as it is the only one that has this parameter. All three have a relation with the pixel intensity data, that is, the intensity detected in each pixel. For the dark feature which is the intensity detected in each pixel when there is no light being observed, we made a subtraction of the pixel intensities data to the dark data. The sky background is the detector pixel intensities from a source-less direction in the sky, also with a subtraction between the pixel intensities data and the sky background data. The flat field is the inverse of the detector pixel-to-pixel sensitivity, so as the description says it is the inverse, so we applied multiplication between the flat field data and the pixel intensities data.

In the statistical analysis, we employed the maximum, minimum, and median values for both the image and a specific point selected by the user, using *NumPy*. Specifically on the pixel page, by selecting specific pixel coordinates on the rasterized image, one can observe the evolution of a specific pixel over the frames. Additionally, on the measurements and commands page, it is possible to select a point of intensity on the vectorized image and track its progression over frames on the line plot.

As referred to in Section 3.2, we integrated three different types of graphics: one for the rasterized image (a 2D image that varies over frames), a vectorized image (indices of the objects in y and frames in x), and a line plot illustrating the evolution of the median value of the image across frames, allowing also the user to select and deselect a line it wants to see, it displays different intensities over frames depending on its page and related object. The primary exception is the measurements page, where we separate two dimensions for the graphics to provide a more detailed analysis for the user.

The rasterized image function is designed to handle the visualization of pixel intensities from the detector of the wavefront sensors in the AO system. The function utilizes several inputs that control various aspects of the visualization, including, for example, the colormap, dark mode, and a slider for navigating through different frames. The `ctx.triggers` are used to detect changes in these inputs, prompting the image to update accordingly. Additionally, the Click-Data input allows users to select specific frames on the vectorized image, enabling detailed examination of individual frames. Both rasterized images and vectorized images allow for the zooming, rescaling, and downloading of pictures, as Plotly provides. To help with the evaluation of the images and graphics, as mentioned previously, we used Click-Data, which allows the user to select an intensity of a frame on the vectorized image, and then it directly shows on the rasterized image the selected frame.

We utilize a hierarchical structure to allow users to understand the relationships and dependencies between sources, sensors, correctors, and control loops, providing a comprehensive view of the AO system. To do so, we use the classes of `aotpy` and their relations. Firstly, we need to compare the existing sources of the AO system with each wavefront sensor source, also displaying the corresponding detector uid, unique identifier, and the measurement name of the wavefront sensor. Secondly, we need to check which loops are control loops and which are offload loops, as the valid ones are the control loops, we iterate over these control loops, and compare their input sensor

to see if it matches with the displayed wavefront sensor if it does then we proceed to check their commanded corrector to the available ones, having then the source, wavefront sensor, wavefront corrector, loop. Thirdly, we showcase the command name associated with the loop.

To determine if the measurement page can incorporate a rasterized image, and how to display it, it is necessary to establish a specific procedure. The measurements obtained from the wavefront sensors are processed and visualized using subaperture masks. This procedure guarantees that only corrected values are showcased, providing precise visual representations. The following is a comprehensive explanation of the procedures involved:

1. Firstly, the system validates the availability of a suitable subaperture mask. If unavailable, a text box will be displayed, indicating that rasterization cannot proceed without the mask;

2. If it has a subaperture mask, the measurements are extracted from the sensor data, and the initial dimensions (x and y) are isolated for processing;

3. An output array is created and initialized with None values to handle missing data. The subaperture mask is then utilized to identify valid measurement indices, determining the placement of the measurements within the output array;

4. For each set of measurements, the values are integrated into the output array at the mask-specified positions. This process is reiterated for each set of measurements, ultimately resulting in a processed array of measurements.

For the commands page, the procedure to determine if it can incorporate a rasterized image is similar to the measurements page. However, instead of checking for subaperture masks, we need to check for the presence of actuator coordinates. Also, on this page, we verify if the command from the loop (control loop) is a deformable mirror; if it is, we add a section on the object's properties about its actuator coordinates and stroke. If the command is not a deformable mirror, it can only have two actuators for a tip-tilt mirror or one actuator for a linear stage, which is not suitable for a rasterized image.

This section has shown the implementation of advanced image manipulation and visualization techniques on our dashboard. These techniques include histogram equalization for improved contrast, interactive interval adjustments, and interactive features. These tools help users extract important insights and make informed decisions during scientific analysis.

# Chapter 4

# Results

In this chapter, the results of the dissertation are documented. Firstly, we present the dashboard and its features, showing the analysis of AOT datasets. Secondly, we present the case study used. The code is available at the following link `https://github.com/bea11/AOT-Dashboard`.

## 4.1 Tool Results

To give an extensive overview and understanding of the dashboard, we will use the **ERIAO.2023-03-31T04_11_56.632.fits** as an example for the AOT fits file due to its comprehensive information, thereby facilitating a more detailed evaluation.

We will start by examining the Home page, presented in Figure 4.1. As shown, there is a navigation bar on the top of the page that allows the user to navigate directly to the desired page. If the uploaded file is an AOT FITS file, its name is displayed. Otherwise, a red warning message appears, as in Figure 4.2, and the rest of the page does not update.

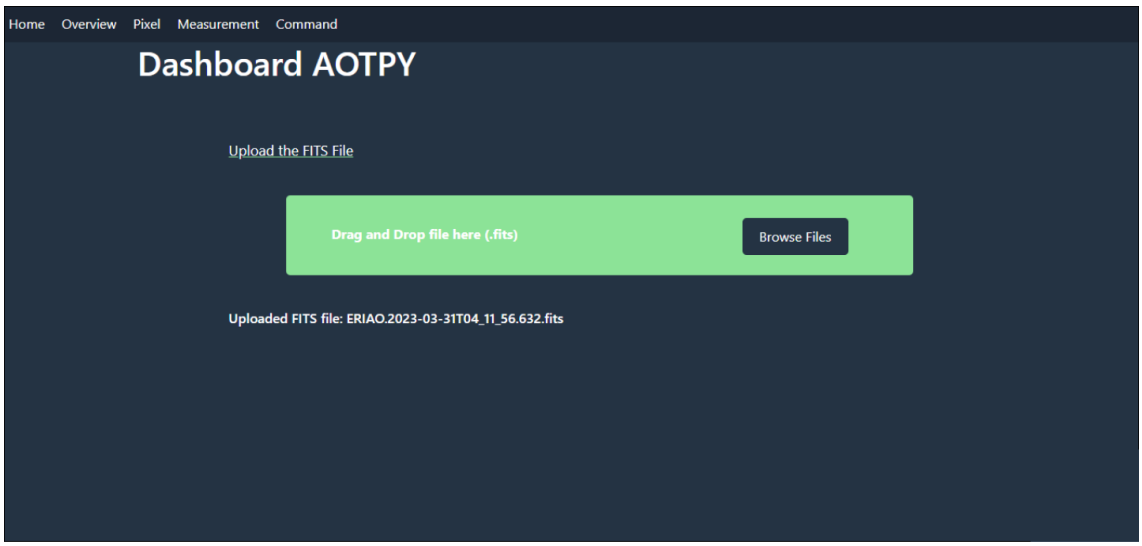



Figure 4.1: Screenshot of the Home page, the initial page.


Figure 4.2: Screenshot of the red warning showed instead of the file's name if an incorrect AOT FITS file is received on the Home page.

After uploading the file, the user can navigate to the Overview page using the navigation bar. This page displays all the main features associated with the file and the characteristics of the AO system. The hierarchy section, which is the bottom section of the display, allows users to choose a source to view the related wavefront sensors, their detectors and measurements, wavefront correctors, loops, and their commands. The user can then navigate to the corresponding pages for more detailed information. This is the Overview page displayed in Figure 4.3.

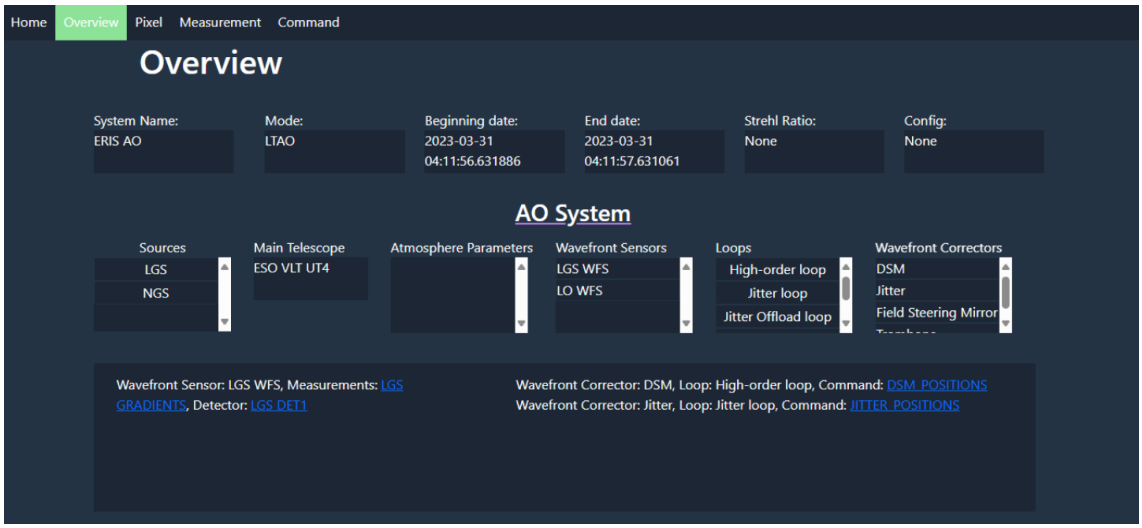



Figure 4.3: Screenshot of the Overview page, that provides an overview of the analysis of the FITS file.

By clicking on the detector of the wavefront sensor shown on the hierarchy section of the previous page or using the navigation bar, the user is redirected to the Pixel page. It contains detailed information about the detector properties and statistics of the selected wavefront sensor (from the dropdown menu). It includes various options for visualizing pixel intensities per frame and the intensity per pixel over time. Users can interact with the visualizations by selecting frames, applying different color scales, and adjusting visualization parameters to gain insights into the data. The Pixel page is shown in Figure 4.4.

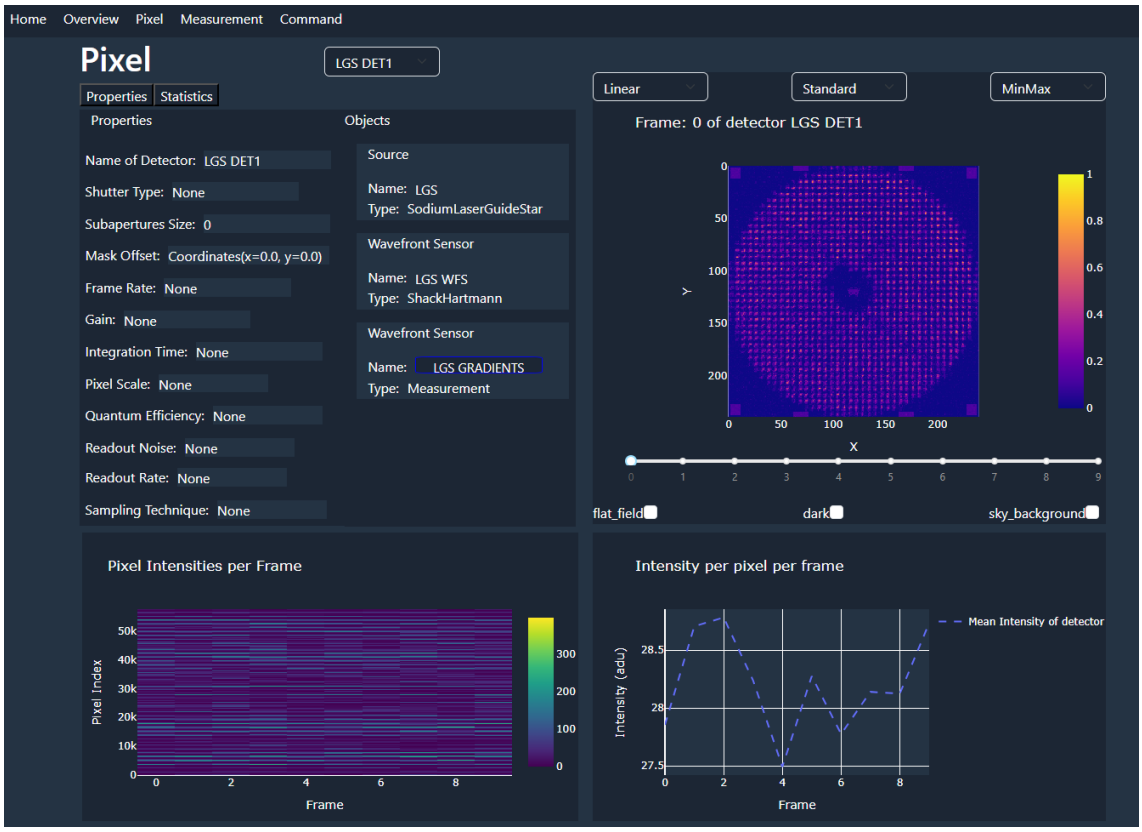



Figure 4.4: Screenshot of the Pixel Page, that displays information and different images.

Now, I will proceed to review various sections of the Pixel page. When a specific point is selected in the vectorized image (bottom left corner), shown in Figure 4.5, its corresponding frame is directly displayed in the rasterized image (top right corner), displayed in Figure 4.6. Additionally, choosing a particular pixel in the rasterized image allows for observing its evolution over time on the line plot (bottom right corner), shown in Figure 4.7 using the coordinates (x, y), facilitating comparison with the detector's mean value, also passing the cursor over the lines allows the user to view the values in more detail. The data representation can be manipulated by changing the scale, color, and interval. The flat_field, dark, and sky_background checkboxes can be used to toggle settings.

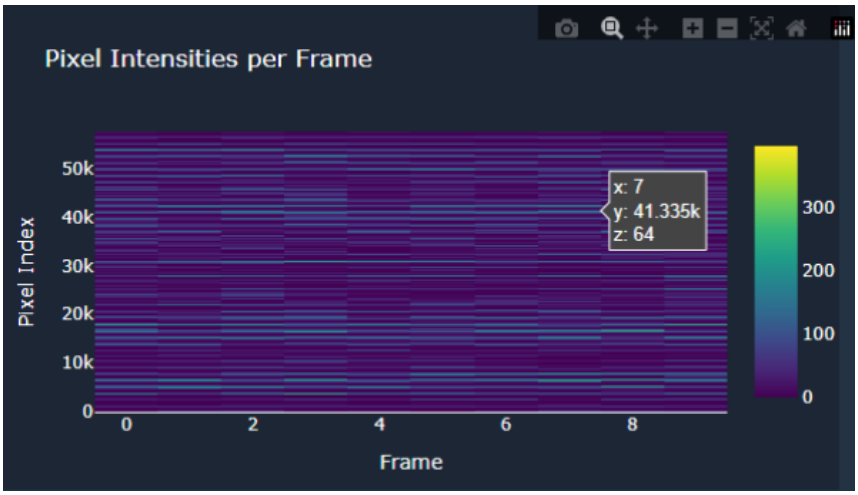



Figure 4.5: Screenshot of the vectorized image of the detector on the Pixel page, while selecting the point with the coordinates $(7, 41.335k)$ being x the number of frames and z (64) the intensity of the pixel.

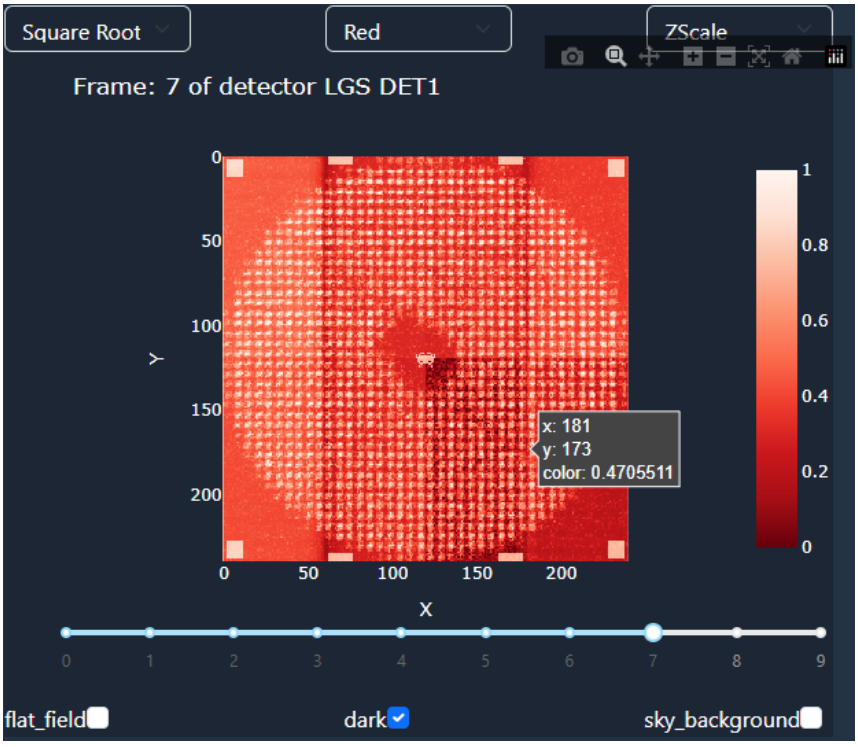



Figure 4.6: Screenshot of the rasterized image of the detector on the Pixel page, with the checkbox dark activated, the frame is the previously selected 7. Selecting the pixel with the coordinates $(181, 173)$ while being able to see exactly the value of the pixel's intensity (0.4705511). Scale: Square Root, Color: Red, Interval: ZScale.

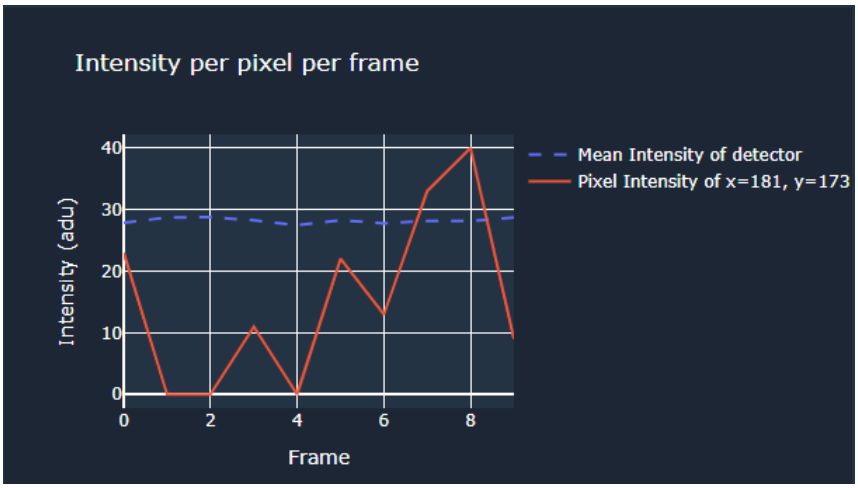



Figure 4.7: Screenshot of the line plot of the detector on the Pixel page, with *adu* meaning analog digital unit. It displays the mean intensity of the detector and the intensity of the previously chosen pixel $(181, 173)$ for each frame.

If the user wants to see in detail the Statistics section, it needs to click on the Statistics button on the right side of the button Properties of the Pixel page. In Figure 4.8, we see a statistical analysis of the pixel intensity data.


Figure 4.8: Screenshot of the statistics section of the Pixel page containing the maximum, minimum, and average value of the image and of the pixel chosen on the previous rasterized image $(181, 173)$.

The Measurement page provides detailed information about the wavefront sensor measurements and the statistics, similar to the pixel page. It includes options for visualizing X and Y

dimensions on the rasterized image (top right corner) and for the measurements per subaperture per frame with the vectorized image (bottom left corner) and the intensities per measurement per frame on the line plot (bottom right corner). Users can interact with the visualizations by selecting frames and applying different color scales and intervals to gain insights into the measurement data. This page is shown in Figure 4.9. The statistics section of the Measurement is similar to the Pixel, figure 4.8, but instead of "Pixel" it says and uses "Measurement" and the measurement is from the chosen vectorized image.


Figure 4.9: Screenshot of the Measurement page.

By selecting different or, in this case, equal points on the vectorized image, we can see them represented on the rasterized image, or by moving the slider, it updates the two images of different dimensions onto the same frame. We have an example of the vectorized image in Figure 4.10 and the rasterized images in Figure 4.11, also related to the line plot on Figure 4.12.

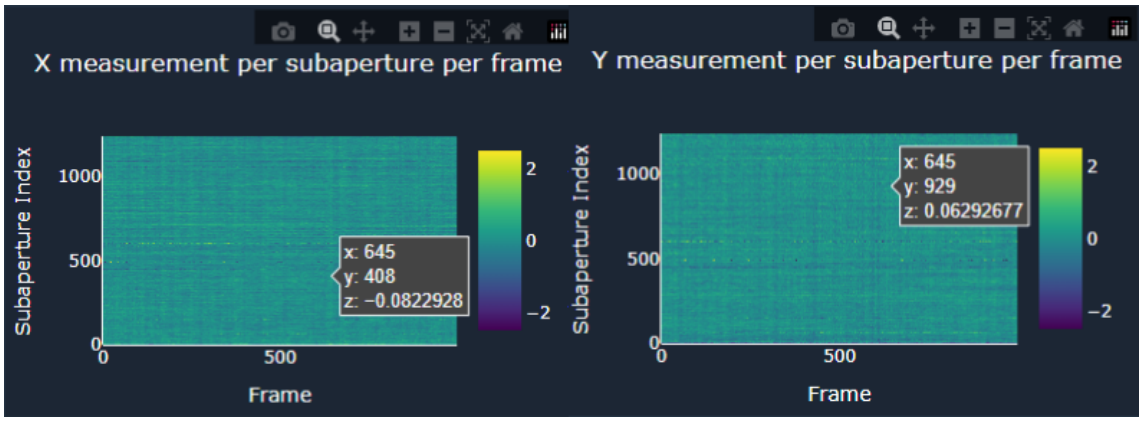



Figure 4.10: Screenshot of the vectorized image of the measurements from the wavefront sensor on the Measurement page, while selecting the point $(645, 408)$, z (-0.0822928) on the X dimension; and the point $(645, 929)$, with the subaperture intensity being higher, z (0.06292677), on the Y dimension. The x coordinate is the number of the frame. z is the intensity of the subaperture.

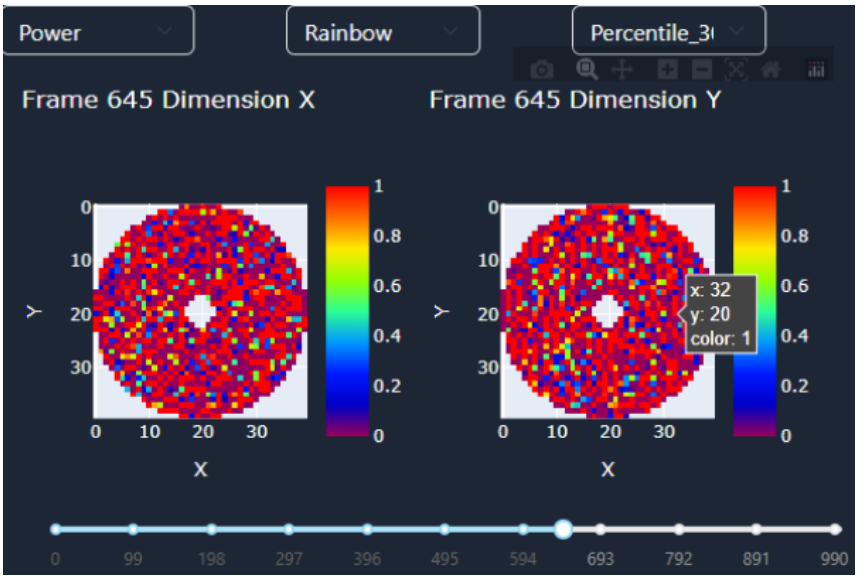



Figure 4.11: Screenshot of the rasterized image of the measurements from the wavefront sensor on the Measurement page, the frame being the previously selected, 645, for both dimensions. On the Y dimension the subaperture intensity, is 1, for the coordinate $(32, 20)$. Scale: Power, Color: Rainbow, Interval: Percentile 30.

In the line plot shown in Figure 4.12, we are observing different measurements from the points selected in Figure 4.10. This is because most of the AOT files do not contain rasterized images due to the absence of the required parameter, as previously explained. Therefore, we relied on the coordinates selected from the vectorized image.

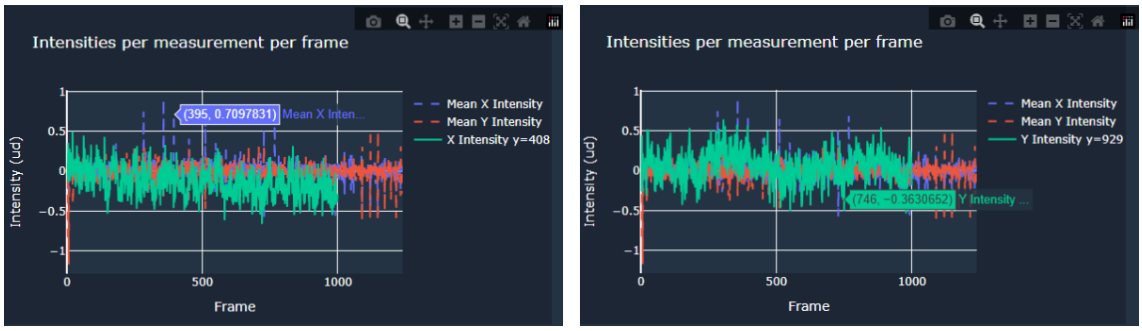



Figure 4.12: Screenshot of two line plots of the measurements from the wavefront sensor on the Measurement page, with *u.d* meaning user defined (according to the AOT specification). Being able to see the mean intensity of the measurements from the wavefront sensor for both dimensions and the evolution of the previously chosen subaperture, 408 for the x dimension (left image) and 929 for the y dimension (right image) for each frame. Passing the cursor over the lines allows the user to view the values in more detail.

The Command page, Figure 4.13 provides comprehensive information regarding the commands of the loops associated with wavefront correctors. It includes options for the rasterized image (top right corner), which shows the warning of the impossibility of displaying the image should actuator coordinates be absent. Additionally, it presents data on the motion per actuator, with a color bar representing the motion per frame on the vectorized image (bottom left corner) and the intensities per actuator motion per frame on a line plot. Users have the capability to interact with the visualizations, including the ability to zoom in for closer inspection.


Figure 4.13: Screenshot of the Command page.

The previous figure does not show the rasterized image due to the file not containing data for the actuator coordinates, so we present in more detail the vectorized image and the line plot on the corresponding Figures 4.14, 4.15.

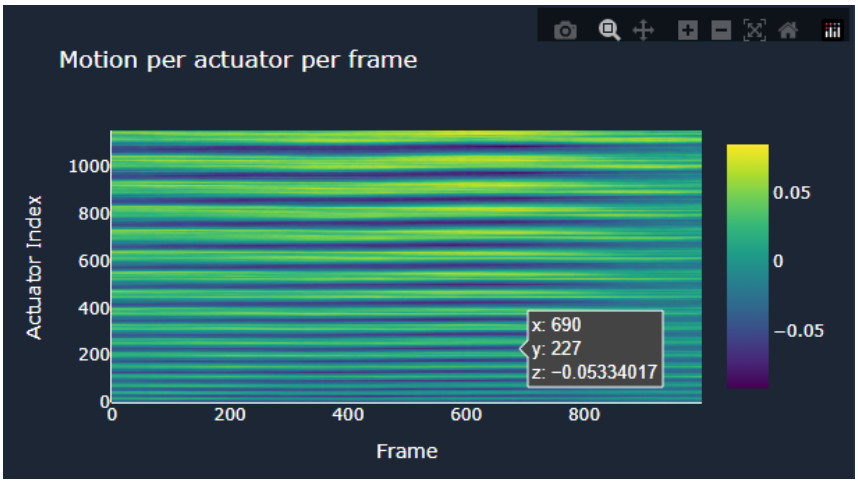



Figure 4.14: Screenshot of the vectorized image of the command on the Command page while selecting the point with $(690, 227)$, being x the number of the frame, and the z (-0.05334017) the motion.

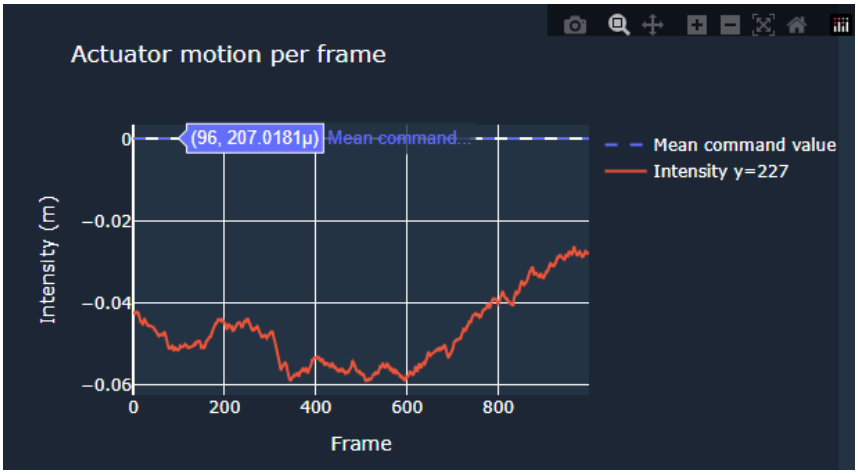



Figure 4.15: Screenshot of the line plot of the command on the Command page, with m meaning meters. Seeing the mean intensity of the command value and the evolution of the previously chosen actuator y=227. Passing the cursor over the lines allows the user to view the values in more detail.

## 4.2 Evaluation

As a case study, we showcased the dashboard results to a team of five individuals, some of whom are adaptive optics experts. The evaluation was conducted to assess the design, functionalities,

and performance of the dashboard using different AO telemetry data files. As part of the evaluation process, we also employed heuristic evaluation as described in subsection 2.1.2 to assess the usability and user experience of the dashboard, albeit in an informal manner integrated into the case study.

### 4.2.1 Case study

For the case study,to guarantee a proper evaluation process, we used the requirements defined at 3.1:

1. **Participants**: Five people, some of them adaptive optics experts involved in the evaluation process who provided valuable feedback on the assessment criteria, and overall effectiveness of the dashboard;

2. **Testing with Different Files**: Various types of AO telemetry data files from distinct instruments (GALACSI, CIAO, NAOMI, and ERIS) were used to test the dashboard. Each instrument presents unique requirements and features for data processing and visualization;

3. **Assessment Criteria**:

    - Data Ingestion and Processing: Ability to read and process different AO telemetry data files;
    - Data Visualization: Quality and interactivity of visualizations, including time-series plots and overlays;
    - User Interaction: Ease of interacting with data points, filtering, and sorting data;
    - Analysis Tools: Availability and accuracy of statistical analysis tools;
    - Performance: Responsiveness and handling of large datasets;
    - Usability: Intuitiveness and user-friendliness of the interface;
    - Consistency: Consistency in layout and design elements;
    - Compatibility: Capacity to display the proper dashboard on different browsers;
    - Scalability: Ability to scale with increasing amounts of data.

Following the evaluation and testing, it was confirmed that the dashboard effectively opened AOT FITS files and processed data from all four instruments that were tested, showcasing robust data handling capabilities. The visualizations were clear and interactive, delivering time-series plots, overlays, and lists of values associated with the objects that met user expectations. The dashboard's interface proved to be user-friendly,consistent and intuitive, enabling users to seamlessly interact with specific data points while also being able to filter and sort data. Additionally, warnings for missing data further enhanced clarity. It was also tested in different browsers, demonstrating compatibility with the most popular options (Microsoft Edge, Firefox, Safari, and Chrome) some of the most popular. The dashboard demonstrated good scalability. It allowed for

updates and modifications, ensuring that best practices for clean and reusable code were followed. Options for filtering and sorting data based on user preferences were available, and the design ensured consistency in typography and iconography for a unified visual theme.

However, some areas for improvement were identified. While the dashboard handled large datasets effectively, users noted that the processing speed could be enhanced. Additionally, although the dashboard provided the necessary statistical features, such as standard deviation, users suggested that adding more statistical analysis tools would further enhance its capabilities. The visual design, while consistent, could benefit from incorporating more diverse shapes to enhance the overall user experience. Users also suggested adding a light mode to accommodate different user preferences.

Overall, the evaluation confirmed that the dashboard adheres to the FAIR data principles and open-source requirements, as it is publicly available, and described. It successfully processed various AO telemetry data files and provided clear, interactive visualizations. User feedback highlighted the dashboard's ease of use, scalability, and robust performance, with suggestions for further enhancements in processing speed and additional statistical features.

# Chapter 5

# Conclusions

In this dissertation, a solution was developed to address the gap in an intuitive and efficient tool for visualizing Adaptive Optics (AO) telemetry data. The resulting dashboard effectively displays and visualizes large datasets using the AOT format, addressing their complexity and that of the *aotpy* package and filling the demand for a more user-friendly platform for AOT file manipulation.

While dashboards have seen widespread adoption, there is no one-size-fits-all approach to their design. Therefore, understanding project requirements and adhering to discussed procedures is critical for successful dashboard implementation. Designing this dashboard required careful planning, including consideration of user experience and visual design. We utilized mockup software to solicit user feedback and ascertain that the tool met the needs of its intended audience. User satisfaction with the current state of the framework is positive, with the requirements of the analyzed cases accomplished with the current offer of data information and visualizations. While we present a complete solution, there is still some room for improvements and to explore the implementation of new features that are described as future work in Section 5.

Choosing an optimal technology stack posed a significant challenge, given the vast options available in the market. However, with clearly defined requirements and a well-structured architecture, we selected a solution that met our visualization needs. As a result, the current dashboard supports public deployment and access across various browsers, demonstrating robust stability and performance.

At this time, there is no solution that integrates the required functionalities for visualizing AO telemetry data as effectively as our proposed dashboard. This tool not only addresses the complexities inherent in the *aotpy* package but also provides a user-friendly platform that enhances data accessibility and usability for the AO community. By integrating advanced visualization techniques with an intuitive interface, this dashboard bridges the gap between complex data handling and user experience. Our work lays a robust groundwork for future developments, ensuring that the tool remains valuable and relevant in the rapidly evolving field of deep space telemetry applications.

## Future Work

Despite having met the primary requirements, there are still opportunities for improvement. In this research, we have identified the following limitations and areas for potential enhancement: performance, a wider range of data types, and increased testing.

In terms of performance enhancement, optimizing dashboard speed could involve transitioning from pickle storage to a database solution such as Redis. This approach would be more efficient in diminishing data retrieval times and improving overall responsiveness, especially under heavier datasets. However, efficiency was not prioritized during this prototype phase, as the current prototype prioritizes functionality and user accessibility.

Additionally, expanding the dashboard's capabilities to include other types of data contained in the telemetry files would broaden its applicability. This could involve integrating new visualization and analysis functionalities, such as computer vision techniques and 3D visualizations. These extensions would enable users to extract deeper insights from the data and leverage the tool for a broader range of applications. The incorporation of account creation and authentication functionalities to facilitate data preservation across sessions is another important aspect to consider. This aspect would enhance user experience and the convenience of seamlessly continuing their tasks across multiple sessions.

Furthermore, while Dash has great scalability, extensive testing with a larger user base is necessary to validate its performance under varied loads. Conducting extensive usability studies, including A/B testing with more participants, would provide deeper insights and ensure comprehensive feedback from diverse user perspectives.

# References


[1] Figma. https://www.figma.com/. Accessed: 2024-03-12.

[2] Nasa eyes. https://eyes.nasa.gov/. Accessed: 2024-03-15.

[3] Render. https://render.com/. Accessed: 2024-05-20.

[4] Giuseppe Aiello. The appearance of diversity: Visual design and the public communication of eu identity. In John Bain and Martin Holland, editors, *European Union Identity: Perceptions from Asia and Europe*, pages 147–181. Nomos, Baden-Baden, 2007. Accessed: 2024-02-18.

[5] Emmanuel Aller-Carpentier, Reinhold Dorn, Francoise Delplancke-Stroebele, Jérome Paufique, Luigi Andolfato, Christophe Dupuy, Enrico Fedrigo, Philippe Gitton, Paul Jolley, Paul Lilley, Miska Le Louarn, Than P. Duc, Andrew Rakich, Javier Reyes, Robert Ridings, Julien Woillez, Enrico Marchetti, Marcos Suarez Valles, Christian Schmid, Norbert Hubin, Jean-Philippe Berger, Jutta Quentin, Bernard-Alexis Delabre, Stewart McLay, and Luca Pasquini. NAOMI: a new adaptive optics module for interferometry. In Jayadev K. Rajagopal, Michelle J. Creech-Eakman, and Fabien Malbet, editors, *Optical and Infrared Interferometry IV*, volume 9146, page 91461C. International Society for Optics and Photonics, SPIE, 2014. Accessed: 2024-04-10.

[6] L. Lawson-Body Assion Lawson-Body and A. Illia. Data visualization: Developing and validating dashboard measurement instruments. *Journal of Computer Information Systems*, 63(3):479–491, 2023. Accessed: 2024-01-20.

[7] Marcos D. Assunção, Rodrigo N. Calheiros, Silvia Bianchi, Marco A.S. Netto, and Rajkumar Buyya. Big data computing and clouds: Trends and future directions. *Journal of Parallel and Distributed Computing*, 79-80:3–15, 2015. Accessed: 2024-01-11.

[8] D. Baines, G. de Marchi, B. Merín, B. Steltzer, H. Bouy, C. Conselice, I. Georgantopoulos, S. Larsen, M.-A. Miville-Deschenes, S. Savaglio, and E. Villaver. Results from esa astronomy space science archives user survey. In F. Pierfederici J. E. Ruiz and P. Teuben, editors, *Astronomical Data Analysis Software and Systems XXX, ASP Conference Series*, volume 532, ESAC Science Data Centre, Quasar Science Resources for ESA-ESAC, Spain, 2021. Astronomical Society of the Pacific, Astronomical Society of the Pacific. Accessed: 2024-03-20.

[9] Deborah Baines, Fabrizio Giordano, Elena Racero, Jesús Salgado, Belén López Martí, Bruno Merín, María-Henar Sarmiento, Raúl Gutiérrez, Iñaki Ortiz de Landaluce, and Ignacio León. Visualization of multi-mission astronomical data with esasky. *Publications of the Astronomical Society of the Pacific*, 129(972):028001, December 2016. Accessed: 2024-05-20.

[10] David Beer and Roger Burrows. Popular culture, digital archives and the new social life of data. *Theory, Culture & Society*, 30(4):47–71, July 2013. Accessed: 2024-01-10.

[11] F. Bonnarel, P. Fernique, O. Bienaymé, D. Egret, F. Genova, M. Louys, F. Ochsenbein, M. Wenger, and J. G. Bartlett. The ALADIN interactive sky atlas. A reference tool for identification of astronomical sources. *Astronomy & Astrophysics Supplement*, 143:33–40, April 2000. Accessed: 2024-05-20.

[12] Prof. B. R. Boruah. Wavefront sensing group. https://www.iitg.ac.in/physics/fac/brboruah/htmls/wfs.html, 2023. Accessed: 2024-03-15.

[13] Maarten A. Breddels and Jovan Veljanoski. Vaex: big data exploration in the era of gaia. *Astronomy & Astrophysics*, 618:A13, October 2018. Accessed: 2024-03-15.

[14] T.O.F. Conrad, E. Ferrer, D. Mietchen, et al. Making mathematical research data fair: Pathways to improved data sharing. *Scientific Data*, 11:676, 2024. Accessed: 2024-03-20.

[15] DataCamp. Datacamp tutorial: Streamlit. https://www.datacamp.com/tutorial/streamlit, Year of access. Accessed: 2024-03-03.

[16] Micheline Elias and Anastasia Bezerianos. Annotating bi visualization dashboards: Needs & challenges. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, page 1641–1650, New York, NY, USA, 2012. Association for Computing Machinery. Accessed: 2024-02-27.

[17] Stephen Few. Common pitfalls in dashboard design. Principal, Perceptual Edge, February 2006. https://www.perceptualedge.com/articles/Whitepapers/Common_Pitfalls.pdf.

[18] Stephen Few. What ordinary people need most from information visualization today. *Perceptual Edge: Visual Business Intelligence Newsletter*, 2008. Accessed: 2024-01-10.

[19] Adam Ginsburg, Brigitta Sipőcz, C. Brasseur, Philip Cowperthwaite, Matthew Craig, Christoph Deil, James Guillochon, Giannina Guzman, Simon Liedtke, Pey Lim, Kelly Lockhart, Michael Mommert, Brett Morris, Henrik Norman, Madhura Parikh, Magnus Persson, Thomas Robitaille, Juan Segovia, Leo Singer, and the collaboration. astroquery: An astronomical web-querying package in python. 01 2019. Accessed: 2024-03-05.

[20] Tiago Gomes, Carlos Correia, Lisa Bardou, Olivier Beltramo-Martin, Thierry Fusco, Caroline Kulcsár, Timothy Morris, Nuno Morujão, Benoît Neichel, James Osborn, and Paulo Garcia. Towards virtual access of adaptive optics telemetry data. *arXiv preprint arXiv:2207.14344*, 2022. Accessed: 2023-11-25.

[21] Tiago Gomes, Carlos M. Correia, Lisa Bardou, Sylvain Cetre, Johann Kolb, Caroline Kulcsár, François Leroux, Timothy Morris, Nuno Morujão, Benoît Neichel, Jean-Luc Beuzit, and Paulo Garcia. Adaptive optics telemetry standard: Design and specification of a novel data exchange format. *Astronomy & Astrophysics*, 686:A7, 2024. Accessed: 2024-06-20.

[22] Grafana. Grafana documentation. https://grafana.com/docs/grafana/latest/introduction/, 2023. Accessed: 2024-03-03.

[23] Alexander B. Gurvich and Aaron M. Geller. Firefly: A browser-based interactive 3d data visualization tool for millions of data points. *The Astrophysical Journal Supplement Series*, 265(2):38, March 2023. Accessed: 2024-05-20.

[24] Triana R. Hadiprawoto and Arran L. Ridley. Transparent dashboards: Open data practices for promoting competition-as-motivation in business dashboards. In *2023 IEEE 16th Pacific Visualization Symposium (PacificVis)*, pages 142–146, 2023. Accessed: 2023-12-12.

[25] A. H. Hassan, C. J. Fluke, D. G. Barnes, and V. A. Kilborn. Tera-scale astronomical data analysis and visualization. *Monthly Notices of the RAS*, 429(3):2442–2455, March 2013. Accessed: 2024-05-20.

[26] R. Hausen and B.E. Robertson. Fitsmap: A simple, lightweight tool for displaying interactive astronomical image and catalog data. *Astronomy and Computing*, 39:100586, 2022. Accessed: 2024-05-20.

[27] Metabase Inc. Metabase. https://www.metabase.com/, Year of access. Accessed: 2024-03-03.

[28] Mahtab Karami, Mostafa Langarizadeh, and Mohammad Fatehi. Evaluation of effective dashboards: Key concepts and criteria. *The Open Medical Informatics Journal*, 11:52–57, 2017. Accessed: 2024-01-20.

[29] Daniel Keim, Florian Mansmann, Jörn Schneidewind, and Hartmut Ziegler. Challenges in visual data analysis. volume 4, pages 9– 16, 08 2006. Accessed: 2024-02-27.

[30] S. Kendrew, S. Hippler, W. Brandner, Y. Clénet, C. Deen, E. Gendron, A. Huber, R. Klein, W. Laun, R. Lenzen, V. Naranjo, Udo Neumann, J. Ramos, R.-R. Rohloff, P. Yang, F. Eisenhauer, A. Amorim, K. Perraut, G. Perrin, C. Straubmeier, Enrico Fedrigo, and Marcos Suarez Valles. The GRAVITY Coudé Infrared Adaptive Optics (CIAO) system for the VLT Interferometer. In Ian S. McLean, Suzanne K. Ramsay, and Hideki Takami, editors, *Ground-based and Airborne Instrumentation for Astronomy IV*, volume 8446, page 84467W. International Society for Optics and Photonics, SPIE, 2012. Accessed: 2024-04-10.

[31] Helen Kennedy, Rachel L. Hill, Goffredo Aiello, and William Allen. The work that visualisation conventions do. *Information, Communication & Society*, 2016. Accessed: 2024-01-10.

[32] Cole Nussbaumer Knaflic. *Storytelling with Data: A Data Visualization Guide for Business Professionals*. Wiley, Hoboken, NJ, 2015. Accessed: 2024-01-15.

[33] S. Malik. Enterprise dashboards: Design and best practices for it. 2005. Accessed: 2024-02-27.

[34] A. Moitinho, A. Krone-Martins, H. Savietto, M. Barros, C. Barata, A. J. Falcão, T. Fernandes, J. Alves, A. F. Silva, M. Gomes, J. Bakker, A. G. A. Brown, J. González-Núñez, G. Gracia-Abril, R. Gutiérrez-Sánchez, J. Hernández, S. Jordan, X. Luri, B. Merin, F. Mignard, A. Mora, V. Navarro, W. O'Mullane, T. Sagristà Sellés, J. Salgado, J. C. Segovia, E. Utrilla, F. Arenou, J. H. J. de Bruijne, F. Jansen, M. McCaughrean, K. S. O'Flaherty, M. B. Taylor, and A. Vallenari. Gaia data release 1: The archive visualisation service. *Astronomy Astrophysics*, 605:A52, September 2017. Accessed: 2024-03-20.

[35] Mario Nadj, Alexander Maedche, and Christian Schieder. The effect of interactive analytical dashboard features on situation awareness and task performance. *Decision Support Systems*, 135:113322, 8 2020. Accessed: 2023-12-12.

[36] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '90, page 249–256, New York, NY, USA, 1990. Association for Computing Machinery. Accessed: 2024-01-20.

[37] Koen Pauwels, Tim Ambler, Bruce Clark, Pat LaPointe, David Reibstein, Bernd Skiera, Berend Wierenga, and Thorsten Wiesel. Dashboards as a service : Why, what, how, and what research is needed? *Journal of Service Research*, 12:175–189, 10 2009. Accessed: 2024-02-10.

[38] Federico Quin, Danny Weyns, Matthias Galster, and Camila Costa Silva. A/b testing: A systematic literature review. *Journal of Systems and Software*, 211:112011, 2024. Accessed: 2024-06-10.

[39] Real Python. Build your first dashboard with python and dash, 2023. Accessed: 2024-03-03.

[40] A. Riccardi, S. Esposito, G. Agapito, J. Antichi, V. Biliotti, C. Blain, R. Briguglio, L. Busoni, L. Carbonaro, G. Di Rico, C. Giordano, E. Pinna, A. Puglisi, P. Spanò, M. Xompero, A. Baruffolo, M. Kasper, S. Egner, M. Suàrez Valles, C. Soenke, M. Downing, and J. Reyes. The ERIS adaptive optics system. In Enrico Marchetti, Laird M. Close, and Jean-Pierre Véran, editors, *Adaptive Optics Systems V*, volume 9909, page 99091B. International Society for Optics and Photonics, SPIE, 2016. Accessed: 2024-04-10.

[41] Thomas Sander, Joel Freyss, Modest von Korff, and Christian Rufener. Datawarrior: An open-source program for chemistry aware data visualization and analysis. *Journal of Chemical Information and Modeling*, 55(2):460–473, 2015. PMID: 25558886.

[42] Alper Sarikaya, Michael Correll, Lyn Bartram, Melanie Tory, and Danyel Fisher. What do we talk about when we talk about dashboards? *IEEE Transactions on Visualization and Computer Graphics*, 25(1):682–692, 2019. Accessed: 2024-01-20.

[43] Secoda. Secoda glossary: Redash. https://www.secoda.co/glossary/redash, Year of access. Accessed: 2024-03-03.

[44] Gayane Sedrakyan, Erik Mannens, and Katrien Verbert. Guiding the choice of learning dashboard visualizations: Linking dashboard design and data visualization concepts. *Journal of Computer Languages*, 50:19–38, 2019. Accessed: 2024-02-18.

[45] Remko Stuik, Roland Bacon, Ralf Conzelmann, Bernard Delabre, Enrico Fedrigo, Norbert Hubin, Miska Le Louarn, and Stefan Ströbele. Galacsi – the ground layer adaptive optics system for muse. *New Astronomy Reviews*, 49(10):618–624, 2006. Adaptive Optics-Assisted Integral-Field Spectroscopy.

[46] ThoughtSpot. Best dashboard software. https://www.thoughtspot.com/data-trends/dashboard/best-dashboard-software. Accessed: 2024-03-03.

[47] Alan T. Tokunaga. Chapter 51 - new generation ground-based optical/infrared telescopes. In Tilman Spohn, Doris Breuer, and Torrence V. Johnson, editors, *Encyclopedia of the Solar System (Third Edition)*, pages 1089–1105. Elsevier, Boston, third edition edition, 2014. Accessed: 2024-05-25.

[48] Jay Trimble and George Rinker. Open source next generation visualization software for interplanetary missions. *NASA Ames Research Center, Mountain View, CA, 94035, USA and Jet Propulsion Laboratory, Pasadena, CA 91109, US*. Accessed: 2024-03-15.

[49] Mitchell Tseng, Yue Wang, and Roger Jiao. *Modular Design*. 03 2018. Accessed: 2024-04-8.

[50] UseMotion Team. Dashboard software, 2023. Accessed: 2024-03-12.

[51] Han Wang, Tao Huang, Yuan Zhao, and Shengze Hu. The impact of dashboard feedback type on learning effectiveness, focusing on learner differences. *Sustainability*, 15(5):4474, 2023. Accessed: 2024-01-20.

[52] Xianheng Wang, Veronica Liesaputra, Zhaobin Liu, Yi Wang, and Zhiyi Huang. An in-depth survey on deep learning-based motor imagery electroencephalogram (eeg) classification. *Artificial Intelligence in Medicine*, 147:102738, 2024. Accessed: 2024-05-03.

[53] Victor Weiner, Venkat Balijepally, and Mohan Tanniru. Integrating strategic and operational decision making using data-driven dashboards: The case of st. joseph mercy oakland hospital. *Journal of Healthcare Management*, 60(5):319, 2015. Accessed: 2024-02-27.

[54] Steve Wexler, Jeffrey Shaffer, and Andy Cotgreave. *The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios*. John Wiley & Sons, 2017. Accessed: 2024-01-20.

[55] Dumontier Wilkinson M. and I. et al. M., Aalbersberg. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3:160018, 2016. Accessed: 2024-02-10.

[56] Ben Williamson. Digital education governance: An introduction. *European Educational Research Journal*, 2015. Accessed: 2024-02-18.

[57] Nathan Yau. *Visualize This: The FlowingData Guide to Design, Visualization, and Statistics*. Wiley, Hoboken, NJ, first edition, 2015. Accessed: 2024-01-15.

[58] Ogan M. Yigitbasioglu and Oana Velcu. A review of dashboards in performance management: Implications for design and research. *International Journal of Accounting Information Systems*, 13:41–59, 2012. Accessed: 2023-12-12.

[59] Rafael N. Zambrano and York Engelhardt. Diagrams for the masses. In Gem Stapleton, John Howse, and John Lee, editors, *Diagrams 2008*, pages 282–292, Berlin, Heidelberg, 2008. Springer Verlag. Accessed: 2024-01-20.

# Appendix A

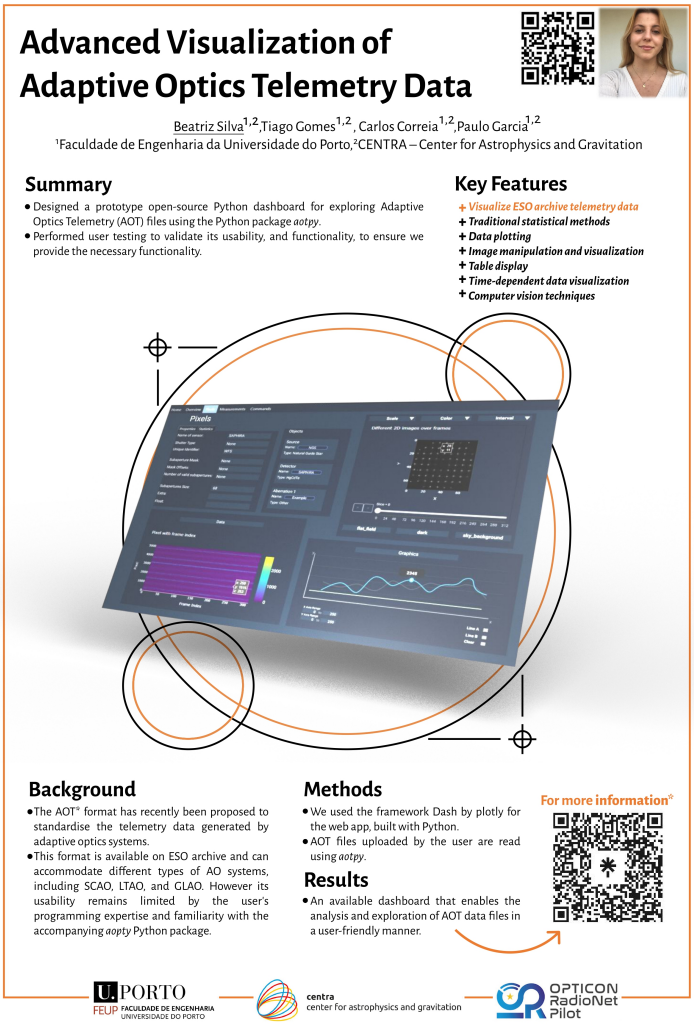
# Repository

The repository for the code of this dissertation is available at https://github.com/bea11/AOT-Dashboard.

# Appendix B

# Poster

This poster was presented at the "Adaptive Optics Systems IX" conference of the SPIE Astronomical Telescopes + Instrumentation 2024, from the 16th to the 21st of June 2024, in Yokohama, Japan. Although this poster was selected for the best poster prize, it was later disqualified because the regulations required the lead author to be present at the conference.



# Appendix C

# Article

This article was submitted to the SPIE Astronomical Telescopes + Instrumentation 2024 conference proceedings.

# Advanced visualization of adaptive optics telemetry data

Beatriz Silva[a,b], Tiago Gomes[a,b], Carlos M. Correia[a,b], and Paulo J. V. Garcia[a,b]

[a]Faculdade de Engenharia da Universidade do Porto, Rua Dr. Roberto Frias s/n, 4200-465 Porto, Portugal
[b]Center for Astrophysics and Gravitation, Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa, Portugal

## ABSTRACT

The Adaptive Optics Telemetry (AOT) format has recently been proposed to standardize the telemetry data generated by adaptive optics systems. Yet its usability remains limited by the user's programming expertise and familiarity with the accompanying Python package. There is an opportunity for substantial improvement in data accessibility by offering users an alternative tool for conducting exploratory data analysis in a visual and intuitive manner. We aim to design and develop an open-source Python visualization tool for exploring AOT data. This tool should support researchers and users by offering a broad set of interactive features for the analysis and exploration of the data. We designed a prototype dashboard and performed user testing to validate its usability. We compared the prototype with existing data visualization and exploration tools to ensure we provided the necessary functionality. We made publicly available a user-friendly dashboard for analyzing and exploring AOT data.

**Keywords:** Adaptive Optics, Telemetry, Visualization, Dashboard

## 1. INTRODUCTION

In today's data-driven world, the effective visualization and interpretation of data play a critical role in numerous scientific and technological domains. Although there have been significant advancements in data collection and processing from ground-based observatories, visualizing complex Adaptive Optics (AO) datasets from the world's largest telescopes is an important area that remains poorly explored. In fact, to our knowledge, no tool designed specifically for the visualization of AO telemetry has ever been made publicly available. This paper will describe how we aim to fill that gap by developing an interactive dashboard displaying any AO telemetry data available in the Adaptive Optics Telemetry (AOT) format.[1]

While AOT and its accompanying Python package (*aotpy*[2]) have laid the groundwork for allowing uniform access to AO telemetry data produced by any instrument or observatory, the authors do not propose any methods or techniques for visualizing the data in such files. Consequently, users who are inexperienced with Python or unfamiliar with the format may struggle with interpreting the data it contains. The AO community will, therefore, benefit from a ready-made tool that allows for the visualization of key telemetry data without requiring any prior Python or AOT experience.

In this paper, we will discuss the user experience design concerns that were considered during the development of the dashboard, some relevant implementation details and the results we obtained.

## 2. USER EXPERIENCE DESIGN

While dashboards have seen widespread adoption, there is no one-size-fits-all approach to their design, and therefore, understanding the project's requirements is critical for a successful dashboard implementation. With this in mind, we designed this dashboard through an iterative process that considers user feedback regarding its experience and visual design. Specifically, we gathered feedback by sharing design mockups with AO experts, modified the design based on their concerns and repeated the process until they considered the tool satisfactory. Of particular interest to us was ensuring that users felt the tool was accessible (easy to understand and use) and that it contained all the necessary features to enable useful insights and efficient research.

Further author information: P.J.V.G.: E-mail: pgarcia@fe.up.pt

### 2.1 User Flow

After user feedback, we converged on having 5 distinct pages, whose flow is represented in Figure 1.

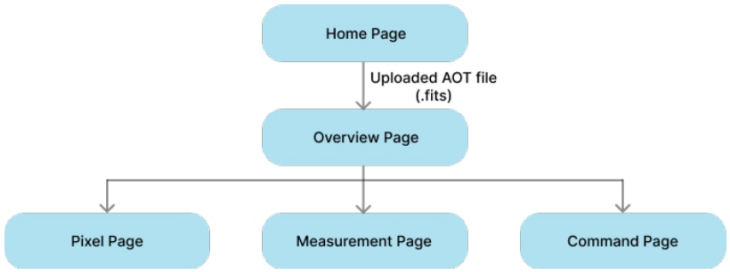



Figure 1. User Flow for the dashboard.

The dashboard layout is carefully crafted to provide a comprehensive view of the AO data, incorporating system parameters, sensor data, and graphical representations. The design emphasizes consistency and simplicity to optimize user experience. The purpose of each page is as follows:

- **Home Page**: serves as the entry point for users to upload their AOT files. After uploading, the page will display the file name to confirm the correct upload or provide a warning message if the file is incompatible.
- **Overview Page**: provides a summary of the operational parameters of the AO system, including system name, mode, and telescope used. Users can navigate the system hierarchy to explore specific sources, wavefront sensors, detectors, and associated measurements and controls.
- **Pixel Page**: displays detailed information about the wavefront sensors and captured detector data. This includes the ability to see the available detectors, sensor properties, light source (such as a Natural Guide Star), associated statistics, and an object section to view the objects associated with the wavefront sensor. Detectors will be represented with 2D visualization, including different options for intervals, colors, scales, and being able to select different features, such as a flat field filter. It will also include visual representations of pixel data over time, vectorized and rasterized images, intensity index through frames, and line plots to check the mean values of the detector along frames.
- **Measurement Page**: similar to the pixel sections, it displays the different measurements of the wavefront sensor and key information. It includes graphical representations of various wavefront sensor measurements, along with relevant statistics. Graphics are similar to the ones on the pixel page but with different titles and measurements, as well as the associated objects. With the separation of different dimensions $(x, y)$, users should be able to identify trends quickly. The different dimensions will be displayed side by side, with distinct colors on the line plot for clear differentiation.
- **Command Page**: provides a streamlined view similar to the previously explained pages, with information about all the available commands of the control loops of the wavefront correctors but with fewer parameters to manipulate. In the objects section, it displays objects associated with the wavefront corrector.

### 2.2 Visual and Interaction Design

By focusing on creating an intuitive and visually appealing interface, we aimed to minimize the learning curve and improve the overall usability of the platform.

Visual elements were selected for their thematic relevance and functionality according to the context. The decision to use dark tones reflects the modern dark mode trend, in which software uses a darker design to


Figure 2. Primary Colors used for the dashboard.

reduce eye strain. The colors were chosen due to personal preference and because they resemble space, with blue dominating the ESO website; the predominant colors in our dashboard are shown in Figure 2.

A cohesive color scheme is used throughout to create a unified aesthetic, with colors selected for their visual appeal and their ability to highlight important information and differentiate various data points. We decided on a top navigation bar that gives the dashboard a more intuitive appearance and improves usability by providing quick access to different pages, enabling seamless navigation. The chosen font style and size optimize readability, reducing eye strain and making data interpretation easier, specifically using a lighter font in contrast to the dark background.

All pages include a navigation bar with a link to all the pages shown in the user flow. The text of this bar changes color when hovered to indicate to the user that they are interactable.

The landing home page has a minimalist design with a single box for users to submit AOT files, ensuring a distraction-free interface for the primary task.

The overview page consolidates the most relevant information into a single, easily navigable screen horizontally aligned for improved readability and quick comparison. Users can easily find specific system information, such as file start date and related data, on the top of the page. This layout also helps the understanding of the AO system's relationships with a hierarchy section of the system at the end of displaying it. For the overview page, users will be able to explore the system hierarchy by first selecting a specific source, revealing a structured view, the wavefront sensor, with its detector and measurement, associated with that source, and the wavefront corrector associated with the wavefront sensor, with the loops (with its commands) associated. Although it is possible to see the offload and control loops on the detailed part of the AO system, only the control loops are presented in this hierarchical form. On the hierarchy structure, the user can be redirected to the corresponding pages by clicking on the detector, measurements, or loops.

The detailed pages, pixel, measurement, and command, are designed similarly to create a user-friendly experience. The standardized style facilitates intuitive navigation, especially for new users seeking main commands quickly. Each is dedicated to specific aspects, such as the wavefront sensor detector (pixel page), wavefront sensor measurements (measurement page), and wavefront corrector loop commands (command page). Key information is prioritized by being placed on the left side for quick reference and to align with typical reading patterns. This ensures immediate access to specific details upon entry, such as the uid (unique identifier) and the statistics.

Each of these 3 pages has clear, rectangular divisions for efficient navigation and locating specific information. On all three pages, there are four distinct sections: 1) the key information (including statistics), 2) the rasterized image with a white slider and appropriate white marks for different values within a pre-defined range, 3) the vectorized image and 4) the line plot. These sections maintain a clean, organized layout and prevent information overload. Visual elements such as charts and graphs that require further inspection are on the right and bottom of the page. Since the charts are synchronized, i.e., user actions in one affect the other, they are placed adjacently, although in clearly distinct sections. For images that allow for manipulation, such as changing color or type of scale, there are drop-down menus available above the image to allow the user to select the desired value for the manipulation. Drop-downs are especially efficient since the user only sees the options when interacting with that element, which reduces the amount of visual clutter. In contrast, for inputs that do not have as many options, mainly binary options, checkboxes were used. The line plots utilize various colors to distinguish between different

values being tracked. These pages also have dropdowns to select different objects if the file contains multiple objects. This ensures users can easily switch between different views and data points, enhancing the dashboard's overall user experience and functionality.

On the pixel page, there is a dropdown to select the desired detector, and with a button, it will display the section dedicated to image statistics and selected points on the 2D image. Detectors will be represented in the 2D graphics and have different drop-down options for intervals, colors, and scales to maximize space efficiency. Additionally, there are checkboxes for features such as flat-field to make it easier and more intuitive for the user. Clicking on a point on the rasterized image will display its intensity through frames on the line plot. In the object section, users will see the objects associated and be able to click on the associated measurement to be redirected to the measurement page.

As stated, the measurement page is very similar to the pixel page but without checkboxes, since their functions are specific to the page's context. If the measurement does not have a mask offset, a text message will inform the user that it will not be available for the rasterized image due to the lack of the required parameter (instead of the image and the slider). For the object section, users will see the associated objects and be able to click on the associated detector to be redirected to the pixel page.

Lastly, the command page, the simplest of all three, does not have a button for the statistics. If the command does not have valid actuators, a text message will inform the user that it will not be available for the rasterized image due to the lack of the required parameter (instead of the image and the slider). The object section will display the objects associated with the wavefront corrector, which can be redirected to the pixel and measurement page with a click on the associated wavefront sensor detector and wavefront sensor measurement.

## 3. SOFTWARE ARCHITECTURE AND TECHNOLOGY STACK

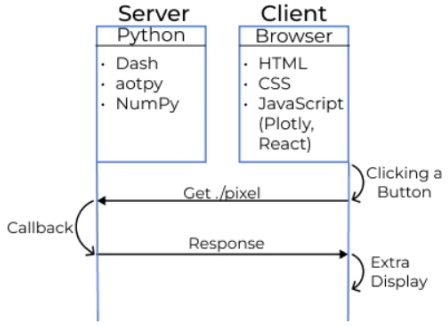



Figure 3. Simplified diagram of the architecture used to build the dashboard, with a practical example of what happens when the user clicks a button.

Based on the requirements for enhanced visualization and efficient data handling for this dashboard, we opted for a client-server architecture. This architecture combines the responsiveness of client-side operations with the robustness of server-side processing. It allows for interactive and dynamic user interfaces with real-time updates while ensuring scalability on the server side. We also decided to build a stateless framework, which is beneficial because it simplifies the management of user interactions and scales more easily. In a stateless architecture, each request from a client to the server is treated independently, without retaining any session information between requests. This reduces the server's memory load, improves scalability, and simplifies the design and deployment of distributed systems.

We built our dashboard's server-side using the Dash framework.[3] This framework fulfils the architectural requirements stated above, and given that it is Python-based, it also allows us to leverage the existing functionalities in *aotpy*. Building this tool on top of *aotpy* enables us to benefit from the package's system-agnostic

approach, allowing for compatibility with a wide range of data-producing systems. The client side is a web application developed using HTML, CSS, and JavaScript (particularly *plotly.js* and *react.js*, which are both integrated into Dash). These implementation details are summarized in Figure 3.

## 4. RESULTS AND CONCLUSION

A prototype of this dashboard was implemented following the FAIR Guiding Principles for scientific data management and stewardship.[4] It has been deployed publicly* and its source code has also been made available on GitHub†. Screenshots of all the pages mentioned in this paper can be found in Appendix A.

Currently, no solution integrates the required functionalities for visualizing AO telemetry data as effectively as our proposed dashboard. This tool addresses the complexities inherent in the *aotpy* package and provides a user-friendly platform that enhances data accessibility and usability for the AO community. This dashboard bridges the gap between complex data handling and user experience by integrating advanced visualization techniques with an intuitive interface. Our work lays a robust groundwork for future developments, ensuring that the tool remains valuable and relevant in the rapidly evolving deep space telemetry applications field.

## ACKNOWLEDGMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no.101004719 (OPTICON–RadioNet Pilot) and from the FCT – Fundação para a Ciência e a Tecnologia through the grant CENTRA UIDB/00099/2020. C.M. Correia acknowledges the financial support provided by FCT with grant 2022.01293.CEECIND.

## REFERENCES

[1] Gomes, T., Correia, C. M., Bardou, L., Cetre, S., Kolb, J., Kulcsár, C., Leroux, F., Morris, T., Morujão, N., Neichel, B., Beuzit, J.-L., and Garcia, P., "Adaptive optics telemetry standard - design and specification of a novel data exchange format," *A&A* **686**, A7 (2024).

[2] Gomes, T., Correia, C., Bardou, L., Beltramo-Martin, O., Fusco, T., Kulcsár, C., Morris, T., Morujão, N., Neichel, B., Osborn, J., and Garcia, P., "Towards virtual access of adaptive optics telemetry data," in [*Adaptive Optics Systems VIII*], Schreiber, L., Schmidt, D., and Vernet, E., eds., *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* **12185**, 121850H (Aug. 2022).

[3] Shammamah Hossain, "Visualization of Bioinformatics Data with Dash Bio," in [*Proceedings of the 18th Python in Science Conference*], Chris Calloway, David Lippa, Dillon Niederhut, and David Shupe, eds., 126 – 133 (2019).

[4] Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.-W., da Silva Santos, L. B., Bourne, P. E., Bouwman, J., Brookes, A. J., Clark, T., Crosas, M., Dillo, I., Dumon, O., Edmunds, S., Evelo, C. T., Finkers, R., Gonzalez-Beltran, A., Gray, A. J. G., Groth, P., Goble, C., Grethe, J. S., Heringa, J., 'T Hoen, P. A. C., Hooft, R., Kuhn, T., Kok, R., Kok, J., Lusher, S. J., Martone, M. E., Mons, A., Packer, A. L., Persson, B., Rocca-Serra, P., Roos, M., van Schaik, R., Sansone, S.-A., Schultes, E., Sengstag, T., Slater, T., Strawn, G., Swertz, M. A., Thompson, M., van der Lei, J., van Mulligen, E., Velterop, J., Waagmeester, A., Wittenburg, P., Wolstencroft, K., Zhao, J., and Mons, B., "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data* **3**, 160018 (Mar. 2016).

## APPENDIX A. DASHBOARD SCREENSHOTS

*Public deployment available at https://aotpy-dashboard.onrender.com/

†Source code available at https://github.com/bea11/AOT-Dashboard

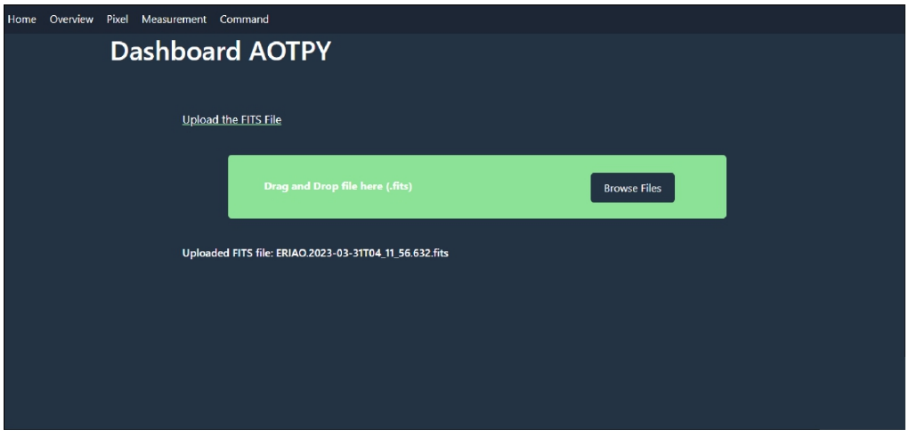



Figure 4. Screenshot of the Home page, the initial page.

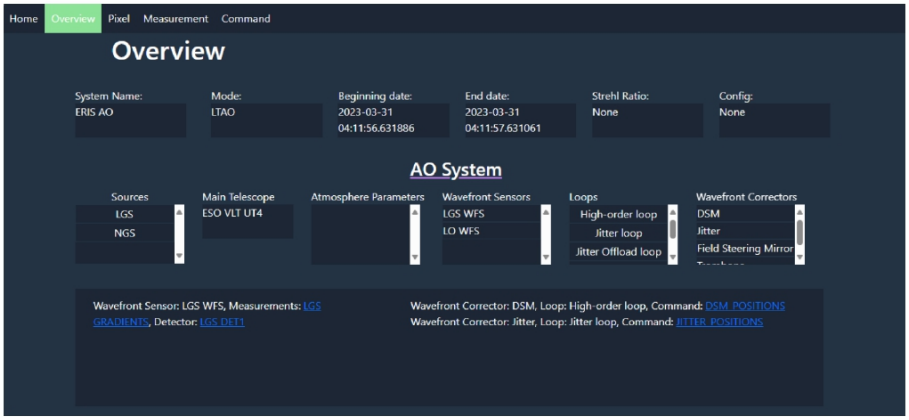



Figure 5. Screenshot of the Overview page, that provides an overview of the analysis of the FITS file.

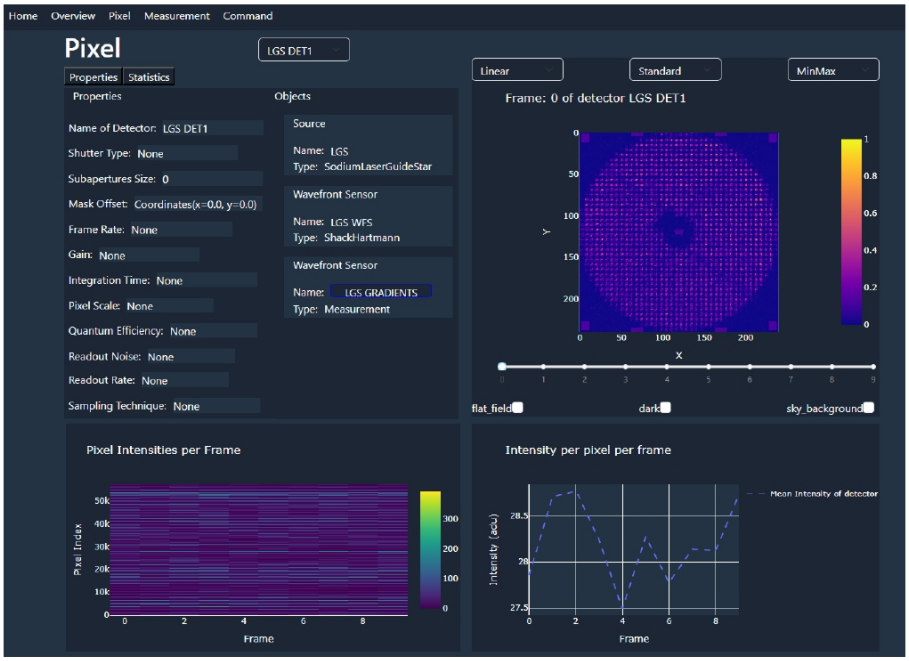



Figure 6. Screenshot of the Pixel Page, that displays information and different images.


Figure 7. Screenshot of the Measurement page.


Figure 8. Screenshot of the Command page.