FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Designing for Astronomical Scale: Visualization and Interaction with Adaptive Optics Telemetry

Ana Rita Baptista de Oliveira

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Carlos Correia

Second Supervisor: Prof. Paulo Garcia

July 31, 2025

Designing for Astronomical Scale: Visualization and Interaction with Adaptive Optics Telemetry

Ana Rita Baptista de Oliveira

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. João Correia Lopes
Referee: Prof. Carlos Correia
Referee: Prof. Paulo Gordo

July 31, 2025

Resumo

À medida que os telescópios continuam a crescer em tamanho e capacidade, a quantidade e a complexidade dos dados gerados pelos seus subsistemas aumentam significativamente. O Extremly Large Telescope (ELT), atualmente em construção pelo European Southern Observatory (ESO), produzirá grandes volumes de dados de telemetria de Ótica Adaptativa (AO), potencialmente vários gigabytes por gravação, que são críticos para a afinação do sistema, avaliação do desempenho e investigação. No entanto, ferramentas que permitem uma exploração eficiente e interactiva deste tipo de dados continuam a ser escassas.

Esta dissertação apresenta o design, implementação e avaliação preliminar de um protótipo de dashboard centrado na visualização e análise da telemetria AO, tendo especialmente em conta a escala e complexidade dos futuros dados do ELT e o formato de dados Adaptive Optics Telemetry (AOT) emergente. O projeto foi motivado pela necessidade de compreender as necessidades de exploração de dados dos investigadores e engenheiros de AO, identificar técnicas eficazes de visualização e renderização para os conjuntos de dados, e que se adaptem às necessidades do utilizador, e conceber uma arquitetura de sistema capaz de lidar com a dimensão e a estrutura da telemetria AO, mantendo uma boa usabilidade e desempenho.

Para responder a estas necessidades, foi utilizada uma abordagem de engenharia de software. O processo começou com a revisão da informação relevante para o problema em questão e com a análise das ferramentas de visualização astronómica atualmente existentes. Seguiu-se uma fase de recolha de requisitos e de documentação que envolveu entrevistas e sessões de observação com especialistas do domínio para melhor compreender a questão do ponto de vista dos potenciais utilizadores. Os resultados deste processo serviram de base ao design arquitetural e visual de um web-based dashboard, tendo sido dada especial atenção às técnicas de tratamento de dados, às optimizações de desempenho do frontend e às estratégias de renderização adequadas a grandes conjuntos de dados sequenciais, como os produzidos no formato AOT.

O resultado final foi um dashboard capaz de visualizar dados de telemetria AOT em vários domínios de dados AO, permitindo aos utilizadores explorar interactivamente grandes conjuntos de dados com baixa latência. Embora as referências formais de desempenho e os testes de utilizadores em grande escala tenham sido limitados devido às restrições do projeto, o sistema foi verificado com êxito em relação aos requisitos documentados. A avaliação também incluiu o feedback das principais partes interessadas, assegurando que a solução fornecida está em conformidade com as necessidades dos utilizadores. O trabalho futuro deve centrar-se em testes de utilizadores e de desempenho mais amplos, bem como na melhoria da ferramenta através do alargamento das actuais funcionalidades.

Abstract

As telescopes continue to grow in size and capability, the amount and complexity of data generated by their subsystems increase significantly. The Extremely Large Telescope (ELT), currently under construction by the European Southern Observatory (ESO), will produce large volumes of Adaptive Optics (AO) telemetry data, potentially several gigabytes per recording, which are critical for system tuning, performance evaluation, and research. However, tools that allow efficient, interactive exploration of this type of data remain scarce.

This dissertation presents the design, implementation, and preliminary evaluation of a dashboard prototype focused on the visualization and analysis of AO telemetry, with special consideration for the scale and complexity of future ELT data and the emerging AOT (Adaptive Optics Telemetry) data format. The project was driven by the need to understand the data exploration needs of AO researchers and engineers, identify effective visualization and rendering techniques for the datasets that fit the user needs, and design a system architecture capable of handling the size and structure of AO telemetry while maintaining good usability and performance.

A software engineering approach was used to address these needs. The process began with reviewing relevant information for the problem at hand and examining currently existing astronomical visualization tools. This was followed by a requirement elicitation and documentation phase involving interviews and observation sessions with domain experts to better understand the issue from the point of view of potential users. The results of this process informed the architectural and visual design of a web-based dashboard, with special attention given to data handling techniques, frontend performance optimizations, and rendering strategies suitable for large, sequential datasets like those produced in the AOT format.

The final result was a functional dashboard capable of visualizing AOT telemetry data across multiple AO data domains, allowing users to interactively explore large datasets with low latency. Although formal performance benchmarks and large-scale user testing were limited due to project constraints, the system was successfully verified against its documented requirements. The evaluation also included feedback from key stakeholders, ensuring the delivered solution aligns with user needs. Future work should focus on broader user and performance testing, as well as improvement of the tool by extending the current functionalities.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) provide a global framework that promotes a better and more sustainable future by establishing goals to address the world's challenges, including poverty, inequality, climate change, environmental degradation, peace, and justice. The SDGs consist of 17 goals aimed at achieving a more equitable and sustainable future by 2030 [1].

This dissertation focuses on the development of an interactive dashboard for the visualization and exploration of Adaptive Optics Telemetry (AOT) data, with specific application to the European Southern Observatory's Extremely Large Telescope (ELT) project. By improving the accessibility and analysis of complex scientific data, this work supports advancements in astronomical research infrastructure and scientific knowledge production, both of which are recognized as key enablers for sustainable development.

The dashboard facilitates interactive exploration of large-scale telemetry datasets, enabling researchers and engineers to identify trends and patterns within adaptive optics systems more effectively. These capabilities contribute to SDG 4 (Quality Education) by promoting learning and capacity building in data analysis and visualization techniques within the scientific community.

Furthermore, by supporting improved decision-making in the development and optimization of astronomical instrumentation, this research aligns with SDG 9 (Industry, Innovation and Infrastructure) and SDG 17 (Partnerships for the Goals), reflecting the collaborative, innovation-driven nature of large-scale scientific projects like the ELT.

Table 1 summarizes how this dissertation's outcomes align with specific SDG targets.

SDG 4 Ensure inclusive and equitable quality education and promote lifelong learning opportunities for all

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 17 Strengthen the means of implementation and revitalize the Global Partnership for Sustainable Development

[1] For more specific goal details, visit the UN website https://sdgs.un.org/goals

Table 1: SDG goals and targets covered by this dissertation.

SDG	**Target**	**Contribution**	**Performance Indicators and Metrics**
4	4.4	Promotes skill development in scientific data visualization and analysis, especially in the interpretation of adaptive optics telemetry data. This benefits researchers in astronomy and data science.	Number of researchers using the dashboard in academic activities or research projects. Frequency of use in training sessions or educational contexts.
9	9.5	Contributes to scientific research infrastructure by providing tools that improve access to and understanding of complex telemetry data. Supports decision-making in the development and testing of adaptive optics systems.	Inclusion of the dashboard in research workflows. Feedback from engineers and scientists. Number of research projects or system tests supported using the dashboard.
17	17.6	Facilitates international collaboration by providing a shared tool for the exploration of adaptive optics telemetry data, encouraging knowledge sharing between institutions involved in projects like the ELT.	Number of collaborative projects or institutions adopting the dashboard. Level of engagement or feedback from international partners. Participation in workshops or feedback sessions involving multiple organizations.

Acknowledgements

I would like to sincerely thank everyone who supported me throughout this journey.

To my supervisor and co-supervisor, Carlos Correia and Paulo Garcia, thank you for all the advice and guidance you provided throughout the project, and thank you for all the patience and encouragement you gave me during harder times.

To the STARPORT team, thank you all for the welcoming nature you showed since the beginning, and a special thanks to Tiago Gomes, who always made time when I needed help.

To my psychologist, thank you for helping me in my journey of self-acceptance and finding balance when things felt overwhelming.

To my family, thank you for always supporting and believing in me, and thank you for making a genuine effort to be understanding during more difficult times.

To all my friends, thank you for the endless love and encouragement you've given me and for always being there when I needed to laugh or cry.

To my sweet dog, Lee, thank you for always offering me comfort and company in the simplest and most reliable ways.

And to my Aurora, thank you for the love you have shown me over these 8 years together, thank you for the endless patience and comprehension, thank you for always being there, thank you for everything.

Rita Oliveira

"We especially need imagination in science.
It is not all mathematics, nor all logic, but it is somewhat beauty and poetry."

Maria Mitchell

Contents

List of Figures

List of Tables

List of Acronyms

AO	Adaptive Optics
AOT	Adaptive Optics Telemetry
DM	Deformable Mirror
ECSS	European Cooperation for Space Standardization
ELT	Extremely Large Telescope
ESO	European Southern Observatory
FITS	Flexible Image Transport System
FR	Functional Requirements
GAVS	Gaia Archive Visualization Service
HCI	Human-Computer Interaction
NFR	Non-Functional Requirements
SDG	Sustainable Development Goals
SRS	Software Requirements Specification
SUS	System Usability Scale
SVG	Scalable Vector Graphics
UCD	User-Centered Design
UI	User Interface
VLT	Very Large Telescope
WFC	Wavefront Corrector
WFS	Wavefront Sensor

Chapter 1

Introduction

Space observatories are entering a period marked by unprecedented data growth, creating a demand for more advanced data visualization and analysis tools. Effective dashboards will play a crucial role in enabling scientists to interpret this data efficiently and with a high degree of accuracy. This dissertation presents the dashboard's ideation, planning, and development for a significant project, the ELT. Tools like dashboards aim to enhance the usability of complex datasets and, therefore, drive scientific discoveries by aiding in the interaction between the data and the researchers.

1.1 Context

The Extremely Large Telescope (ELT), currently under construction and with a planned technical first light in 2029, will be the largest visible and infrared light telescope in the world, which, according to the European Southern Observatory (ESO), will allow scientists to observe the universe with unprecedented resolutions and contribute to discoveries in cosmology, dark matter, exoplanets, among others. There is also the possibility of unexpected discoveries, since the collecting capabilities of the telescope, coupled with other advancements such as adaptive optics, will significantly surpass the capabilities of previous telescopes [21].

However, reaching this level of observational precision is not solely a matter of telescope size. Ground-based telescopes face the constant challenge of atmospheric turbulence, which distorts incoming light and degrades image quality. Adaptive Optics (AO) systems continuously measure these distortions using wavefront sensors and apply real-time corrections through deformable mirrors (DM). This process generates telemetry data streams, including pixel readouts, wavefront slopes, and control commands sent to actuators.

Effectively exploring and interpreting this type of telemetry is crucial for system tuning, performance evaluation, and scientific validation. The current systems and the systems being developed need comprehensive tools for scientists to access and analyze the data collected.

1.2 Motivation

The motivation for this dissertation arises from both the scientific importance of the ELT and the technical challenges associated with AO telemetry visualization. As telescopes increase in size, so does the volume of data produced by their AO systems. For the ELT, estimates from ESO working groups suggest that AO telemetry data alone could reach up to 10 TB per night [22], placing unprecedented demands on data storage, processing, and visualization infrastructures.

Traditional approaches to visualizing telemetry, which are often based on non-interactive plots, are no longer sufficient. Scientists and engineers need tools that allow them to quickly navigate through large datasets, inspect anomalies, validate system performance, and conduct quick and efficient analysis.

One particular challenge associated with creating visualization tools for AO data is its sheer scale. A single AO telemetry data recording can contain tens of thousands of data points spread across multiple domains, often captured at high frame rates over extended periods. This scale not only places high demands on a backend's data handling capabilities, but also can cause significant issues with the frontend's rendering performance. However, even with performance questions aside, this theme raises important questions about how best to structure visualizations to allow meaningful exploration without overwhelming the user.

By addressing these challenges, this dissertation aims to contribute by exploring rendering techniques and scalable architectures in more technical termsa nd by helping the ELT and adaptive optics communities access and interpret AO telemetry more effectively.

1.3 Problem

Telescopes of the modern era are introducing capabilities to increase the scientific prospects of ground-based astronomy. Consequently, as ambitions and telescopes grow, so does the data produced, and so must the tools to analyze it. With 15 times more light-gathering capabilities than the current largest optical telescope, the ELT will generate considerable multi-dimensional data. The current issue is that there is no intuitive and efficient system to visualize and analyze AO data. Therefore, there is an insufficient capability to effectively present the data, hindering scientists from drawing meaningful inferences from the observations.

With this in mind, the core problem tackled by this dissertation is the lack of existing visualization tools capable of efficiently handling the size and complexity of AO telemetry data in an interactive, efficient environment. The addition of the future possibilities brought on by the ELT introduces considerations related to the magnitude of data.

1.4 Research Questions

The development of a visualization tool for ground-based telescopes involves addressing several complex challenges, as mentioned earlier. To guide this work, several key research questions were

formulated to aid in understanding user needs, selecting appropriate technologies, and addressing the requirements for handling and presenting scientific data.

RQ1 What are the data exploration needs of researchers and engineers working with AO telemetry data?

RQ2 Which visualization and rendering techniques are most effective for interactively exploring large, multi-domain AO telemetry datasets?

RQ3 How can a system architecture be designed to efficiently handle the scale and complexity of ELT AO telemetry while maintaining good performance and usability in a web-based dashboard?

1.5 Objectives and Expected Results

This project's primary focus is the development of a dashboard prototype optimized for exploring AO telemetry data from the ELT, with attention to both backend performance and frontend user experience.

Specific objectives include:

- Identifying the key requirements and challenges of AO telemetry visualization;
- Exploring and selecting suitable rendering and data handling techniques;
- Designing a system architecture capable of supporting the necessary performance and scalability;
- Developing a functional web-based prototype;
- Validating the solution with realistic test data and stakeholder feedback.

Although the system could not be evaluated using actual ELT data, as the telescope is not currently functional, the expected result is a working prototype that demonstrates the feasibility of interactive AO telemetry exploration at the expected scale, and offers insights that could guide future tool development in this domain.

1.6 Dissertation Structure

In addition to the introduction, this dissertation contains six more chapters, structured as follows:

- **Chapter 2** presents an overview of the research in visualization, including astronomy-related case studies, and an overview of AO systems and visualization challenges.
- **Chapter 3** discusses the requirements elicitation process and system goals.
- **Chapter 4** describes the design decisions in both system architecture and visual design.

- **Chapter 5** details the implementation of the dashboard.
- **Chapter 6** provides an evaluation of the prototype based on testing and stakeholder feedback.
- **Chapter 7** concludes the dissertation and outlines directions for future work.

Chapter 2

State of the Art

The knowledge discussed throughout this chapter was selected through a systematic literature review. The objective of the process carried out was to ensure that the State of the Art would be grounded in recent and high-quality research.

The search covered multiple academic databases, namely the Web of Science, IEEE Xplore, ACM Digital Library, SCOPUS, and Astronomy & Astrophysics. Keywords included terms such as "dashboard", "data visualization", "multidimensional data visualization", and domain-specific phrases like "adaptive optics". Selection criteria prioritized papers published after 2018, favoring highly cited studies and publications in first-quartile (Q1) journals.

Reference management tools like Zotero were used to organize the collected works, and citation chaining helped identify key papers frequently referenced in the fields. This process ensured that the works cited in this chapter reflect both relevance and academic impact.

2.1 Interactive Visualization

Interactivity allows users to explore data by transforming static visualizations into dynamic tools for exploration and insight generation. It allows users to manipulate data and explore relationships in real-time [30].

Zong et al. [36] also make a distinction between interaction and animation, stating that interactions, in response to the input from the user, transform data and update visual properties. In contrast, animations do the same in response to a timer, meaning they belong to a spectrum of dynamic, event-driven behaviors. The authors created a table, Table 2.1, synthesizing interaction intents and animation types, based on Yi et al. [34] and Heer and Robertson [13] respectively, for standard techniques in interaction and animation.

Yi et al. [34] offer a foundational taxonomy, categorizing interaction techniques into seven types: select, explore, reconfigure, encode, abstract/elaborate, filter, and connect. These categories represent user intents in interacting with visual data, providing a vocabulary to describe common actions:

- **Select:** Highlight or isolate specific data points for focused analysis.

- **Explore:** Navigate data through panning, zooming, or rotating views.
- **Reconfigure:** Rearrange data representations (e.g., sorting or swapping axes).
- **Encode:** Change the mapping of data to visual properties (e.g., color, size).
- **Abstract/Elaborate:** Adjust levels of detail or aggregation, like zooming into subgroups.
- **Filter:** Remove unnecessary data to focus on relevant subsets.
- **Connect:** Show relationships between data points across views or layers.

Table 2.1: Techniques common to interaction and animation taxonomies [36].

Example technique	Interaction intent [34]	Animation type [13]
Conditional encoding	Select	—
Panning	Explore	View transformation
Zooming	Abstract / Elaborate	View transformation
Axis re-scaling	Reconfigure	Substrate transformation
Axis sorting	Reconfigure	Ordering
Filtering	Filter	Filtering
Enter/exit	Explore	Timestep
Multi-view	Connect	—
Changing encodings	Encode	Visualization change, Data schema change

Despite its benefits, designing effective interactions presents significant challenges. As by Walny et al. [32], one key difficulty lies in articulating data-dependent interactions. Designers must account for how user actions, such as clicks or hovers, manipulate data views while ensuring these interactions are intuitive and consistent. Additionally, changing datasets poses a challenge, as they require designers to preserve interaction mappings across iterations. In larger teams, communication gaps between designers and developers further complicate interaction design, often leading to inconsistencies or incomplete implementations. Addressing these challenges requires improved prototyping tools and collaboration frameworks that allow for iterative refinement while maintaining interaction integrity.

2.2 Dashboard Fundamentals

Dashboards are important tools in the realm of data visualization [29]. However, as Sarikaya et al. [25] noted, the definition of a dashboard is not concrete, the term is used to describe different concepts. Some definitions, such as Few's, emphasize dashboards as a visual display designed to organize and consolidate the most crucial data required to monitor information and achieve goals at a glance [10]. Others, as Wexler et al. [33] described, focus more on dashboards as tools to "monitor conditions and/or facilitate understandings", opting for a view of dashboards that accommodates a wider range of uses, including broader communication and storytelling purposes.

At their core, dashboards serve as dynamic interfaces that aid in decision-making, communication, and learning. Their functions often align with strategic, tactical, or operational objectives [29]. Beyond their traditional decision-making roles, dashboards also extend to foster communication and educate audiences on specific concepts [25, 26].

Despite their potential, there are also challenges associated with dashboards. Dashboard designers need to overcome some specific issues, namely data accuracy, integration with many varied data sources, and presentation of information to meet users' varying needs. These challenges require creative approaches and deeper research to make the dashboards work efficiently in the face of users' needs.

2.2.1 Dashboard Categories

Dashboards can be generally categorized in multiple ways based on their purpose, audience, and level of interaction. One such way was proposed by Few, where dashboards were classified based on their role in three categories: strategic, operational, and analytical [10].

Strategic dashboards provide an overview of all the data to help with strategic decision-making. They tend to focus on static snapshots of periods, with high-level measures of performance, that is, they tend to offer benchmarks so that the user can understand the state of the data [10, 19, 25].

Operational dashboards, however, are designed to monitor real-time data to provide immediate insight into ongoing processes. They are tools that focus on the present or near past operations, being often used by individuals or teams to track performance indicators, detect anomalies and warnings, and ensure smooth operations. Examples of this type of dashboard include its use in production lines and logistics [10, 25].

Analytical dashboards, in contrast, present data in much more detail to assist in data analysis. These dashboards are often a combination of interactive charts, filters, and features that aid in finding insights, discovering trends, and diagnosing issues. They are exploratory in nature and designed for decision-makers who require detailed data to inform tactical plans and responses [10, 19].

Expanding on Few's classification, contemporary research gives a wider view of additional perspectives on dashboard design. For example:

- **Audience and Purpose:** Dashboards can be designed with a specific audience in mind. Sarikaya et al. propose public, social, organizational, and individual groups. Understanding that each audience tends to have different needs, varying levels of visualization literacy, and domain expertise also holds importance. [25].

- **Functionality:** Dashboards can range from static tools that display fixed information to interactive platforms that incorporate features such as drill-downs, filtering, and alerts. Interactive analytical dashboards, in particular, enable users to explore scenarios and optimize decision-making [19].

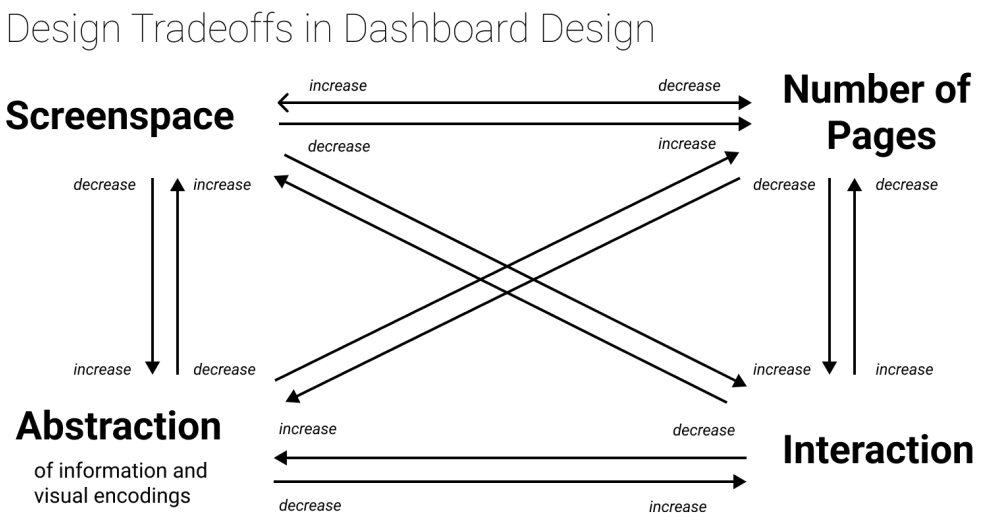


Figure 2.1: A simplified model for design tradeoffs in dashboard design. Reducing one of the parameters (screenspace, abstraction, number of pages, interaction) requires an increase in one of the other parameters. [1]

- **Genres:** Dashboards can also be categorized into different genres based on the intended use, complementing Few's categories. One such genre is the narrative dashboard, which focuses on storytelling and communication, extending the functionality beyond decision support to include learning [1].

By considering these nuances, dashboard design can better align with user needs and contexts, ensuring effectiveness and usability in diverse scenarios.

2.2.2 Dashboard Design

2.2.2.1 Information Architecture

Effective dashboards need to be designed around the user's goals, and the first step is information structuring. This involves creating clear hierarchies of information and displaying them in a useful and relevant order. Dashboards should emphasize critical metrics so that the overall picture can be effectively captured quickly. In [35], the authors highlight the importance of layout order in dashboard design, adding that, besides the idea that "less is more", designers should also adopt the idea that "order is more", focusing on the recommendation of positioning the core chart in the left-center.

Another important aspect is the balance between detail and summary that dashboards should achieve, but this is not simple, as tradeoffs must be made between screenspace, abstraction, interaction, and the number of pages. Bach et al. explain that minimizing these parameters should be the "gold standard", but as shown in Figure 2.1, minimizing one variable often implies the need for an increase in the others [35].

2.2.2.2 Visual Design

A well-crafted dashboard focuses on clarity, simplicity, and usefulness to help in comprehension and decision-making. The visual elements should be easy to understand, and the most important details should stand out. Clear designs ensure every part serves a purpose, avoiding extra clutter. As Stephen Few puts it, "Producing work that looks cool but is incomprehensible or meaningless is a waste" [11]. Of course, dashboards still need to contain information, however, they should lack unnecessary embellishments per Tufte's data-ink ratio, meaning, within reason, the ratio between "ink" devoted to data and the total "ink" should be maximized [28].

Color also plays a key part in dashboard design, not only for aesthetics but also as a way to convey meaning. Good color use involves careful choices to show trends, anomalies, and links in the data. For instance, using a wide color range keeps key differences clear [27]. Sequential color schemes, which transition from light to dark, are particularly effective for ordered data, while diverging schemes work well to emphasize extremes with a neutral midpoint for balance [16]. However, the color's encoding impact changes with mark types. The color encoding in elongated marks like bar charts and line graphs is more discriminating than in the small scatterplot points, which emphasizes the need to choose color and shape carefully [27]. Accessibility is also crucial in visual design. Dashboards should consider visually impaired users by using clear color palettes for all, thereby improving access while keeping functionality.

2.2.2.3 User-Centered Design

A User-Centered Design (UCD) is a technique used in the development of interfaces with the aim of delivering solutions that best suit the user, their requirements, goals, and preferences at the time of designing a product, in this case a dashboard [24]. From the perspective of information visualization and dashboard design, UCD ensures that complex data is represented in an intuitive and accessible manner, enabling users to derive insights effectively.

UCD involves actively engaging with the end-users to tailor the system's design and functionalities to their preferences [24]. This process is vital to ensure that the end users can easily use the final product and that their needs are well catered for, thus ensuring satisfaction. This often entails conducting user surveys to determine the users' needs, desires, and problems. Usability testing and iterative feedback loops in the design process enable the optimization of the visualizations to meet the users' expectations. For example, in the context of participatory citizen science, UCD helps ensure that tools are appropriate for both analytical experts and non-professional people by offering the proper set of features [30].

UCD is closely related to accessibility since it aims to design the interface in a way that will be convenient for users with different abilities and backgrounds. This entails using appropriate color schemes, clearly defined labels, and designing interfaces compatible with visual impairments like color blindness [30]. Accessibility also extends to cultural considerations, ensuring visualizations are comprehensible across different demographics.

An iterative approach ensures continuous improvement by incorporating user feedback at every stage. Each iteration assists in determining the usability problems, enhancing data representation, and improving the interaction techniques [24]. For instance, the dashboards that were developed as part of the GREENGAGE project were refined over several iterations to simplify the features for the general users while at the same time sustaining the complex features for the domain experts [30].

2.2.3 Evaluation

Dashboard evaluation is critical in ensuring that the tools effectively support decision-making and data analysis. The evaluation intends to determine usability, functionality, and the extent to which dashboards meet user needs. By identifying strengths and weaknesses, developers can improve their designs to enhance user experience and overall utility. This section explores the key criteria and methods used in dashboard evaluations.

2.2.3.1 Evaluation Criteria

A set of well-defined criteria ensures that dashboards deliver on their promise of presenting actionable insights in a user-friendly manner. These criteria range from technical functionality to the psychological aspects of user interaction, offering a holistic view of quality. Criteria for what makes a good or bad interface are not something universally established, but the following criteria for software quality standards were defined by ISO [14]:

- **Functional Suitability:** This criterion measures whether the software provides the necessary functionality, performs it correctly, and is relevant to user needs, meaning it measures its completeness, correctness, and appropriateness.
- **Performance Efficiency:** Assesses how efficiently the software uses resources, focusing on time behavior like response times, resource utilization, and its ability to handle loads, that is, the capacity.
- **Compatibility:** Compatibility ensures the software can operate alongside other systems, focusing on co-existence and interoperability.
- **Interaction Capability:** This criterion, called usability in earlier standards, measures how intuitive and user-friendly the software is, focusing on the recognizability of its purpose and its appropriateness, ease of learning and use, called learnability and operability, error protection, interface aesthetics, and accessibility.
- **Reliability:** Reliability refers to the software's ability to function as expected, accounting for its maturity consistently, that is, its stability over time, availability during use, fault tolerance, and recoverability from failure.

- **Security:** Security addresses protecting data and operations, ensuring confidentiality, integrity, accountability, and authenticity, and preventing repudiation of actions.
- **Maintainability:** Maintainability evaluates how easily the software can be modified or updated, focusing on its modularity, reusability, ease of analysis for issues (analyzability), modifiability, and testability.
- **Flexibility:** This quality highlights the software's adaptability to different needs or conditions, including scalability to handle growth and replaceability for integrating new components.
- **Safety:** Safety ensures the software minimizes risks by preventing hazards, controlling potential issues, and implementing fail-safe mechanisms to mitigate harm.

2.2.3.2 Evaluation Methods

Various methods have been developed to systematically assess dashboards, ensuring they meet the defined criteria. These methods range from heuristic reviews by experts to empirical user testing, offering complementary perspectives on a dashboard's performance and usability.

Heuristic evaluation holds significant importance in the field of HCI and consists of a process in which evaluators assess interfaces against a set of established usability principles called heuristics. This approach efficiently identifies flaws while needing fewer resources than large-scale user testing. However, there are some drawbacks to the heuristic evaluation, including the fact that it may not pick out contextual problems that occur when the system is in use [5]. Nielsen [20] proposed ten heuristics that serve as a guide to assess whether an interface aligns with core principles such as usability, consistency, clear feedback, and error prevention, among others. These heuristics not only have significance in the evaluation of an interface but also in the design of the interface.

Another method often used in the evaluation process is user testing, which involves observing users interact with the interface to see how well the system supports different tasks. Different metrics like task success rate, completion time, and error frequency provide quantitative insights, while interviews and think-aloud protocols offer qualitative feedback. As highlighted by Paz and Pow-Sang [23], user testing is invaluable for identifying usability barriers and understanding user behavior in context.

As mentioned, besides the quantitative metrics, qualitative metrics like surveys are a scalable method for gathering feedback from a broad user base. Tools like the System Usability Scale (SUS) provide a standardized way of assessing user satisfaction, ease of use, and perceived utility [2]. These instruments are beneficial for comparing a dashboard's performance with industry standards and for gathering practical recommendations for the incremental improvement of the dashboard. By combining quantitative scores with open-ended questions, surveys capture both general trends and detailed user experiences.

2.3 Case Studies

This section examines three prominent visualization tools in astronomy: the ALADIN Sky Atlas, Gaia Archive Visualization Service (GAVS), and the ESO Archive Science Portal. Each tool was examined in terms of its purpose, features, and relevance to this dissertation.

2.3.1 ALADIN

The ALADIN interactive sky atlas is a powerful tool for visualizing and exploring astronomical data, designed to facilitate multi-wavelength cross-identification and analysis [3]. ALADIN allows astronomers to directly compare observational data across different wavelengths against digitized sky images and reference catalogs. Its versatility has made it a cornerstone for astronomical data visualization, and it is being used as a plugin in other tools, such as the ones presented in the sections below.

ALADIN is available in two modes, ALADIN Desktop[1], as seen in Figure 2.2, and ALADIN Lite[2], the former being a desktop application and the latter being a simpler, lightweight, Web application version of the desktop version. Still, both offer a range of features tailored to the needs of the astronomical community. One of the main features is the ability to visualize digitized sky images. The atlas integrates Digitized Sky Surveys (DSS) data and other high-resolution sources that users can access and explore as skymaps with functionalities like zooming, panning, and color adjustments. Both versions allow the user to see the skymaps with the various projections. The tool also supports simultaneous visualization of objects from different databases such as Simbad[3], VizieR[4], and NED[5]. This integration aids in cross-referencing astronomical sources. ALADIN allows users to analyze the different data types through the use of various operations described in the manual [9], such as color maps, contours, color compositions, pixel calculations, image resampling, different plotting options, drawing, etc.

ALADIN exemplifies some principles in effective dashboard design, particularly in its ability to display complex datasets on single or multiple screens in an intuitive way, as well as its ability to provide intuitive interactions. Its multi-layered approach to data visualization aligns with the goals of this dissertation, demonstrating how sophisticated tools can streamline the exploration of astronomical big data.

However, ALADIN has several limitations that impact user experience. The Java-based interface, though functional, appears outdated and may not leverage modern design aesthetics. Enhanced visual appeal and better use of space could improve usability. Additionally, ALADIN supports around 1.5 million sub-images stored in a relational database, which are retrieved via SQL queries, which can cause some performance issues if not dealt with correctly, especially with highly detailed or large-scale queries.

[1] https://aladin.cds.unistra.fr/AladinDesktop/
[2] https://aladin.cds.unistra.fr/AladinLite/
[3] https://simbad.cds.unistra.fr/simbad/
[4] https://vizier.cds.unistra.fr/viz-bin/VizieR
[5] https://ned.ipac.caltech.edu

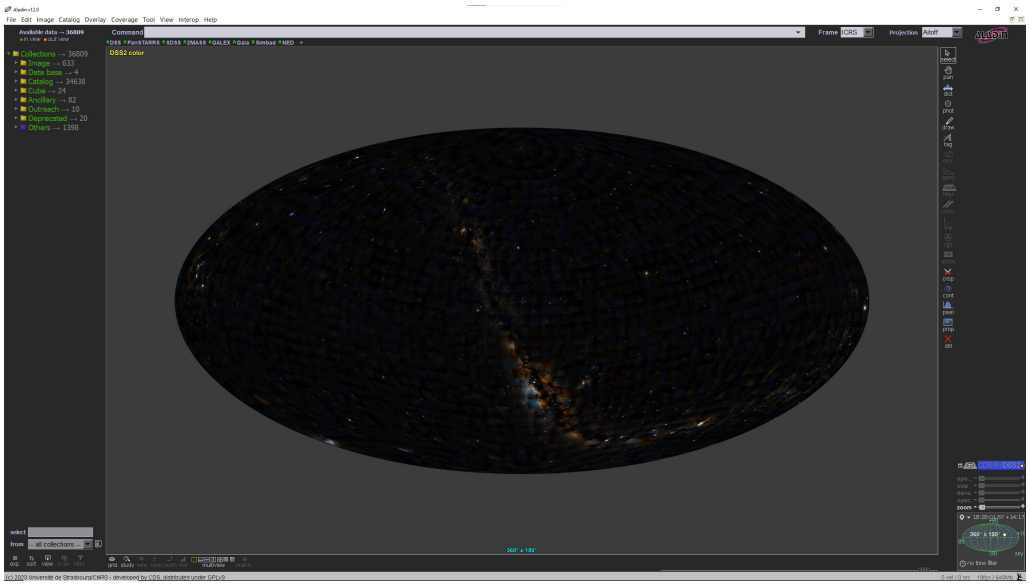


Figure 2.2: The ALADIN Desktop application displaying a skymap

2.3.2 GAVS - Gaia Archive Visualization Service

The Gaia Archive Visualization Service (GAVS) [6] is a specialized tool for accessing and exploring data from the European Space Agency's Gaia mission [17]. This tool was developed to address the challenge of exploring the immense volume of data produced by the Gaia mission, as traditional visualization tools were insufficient in handling such larger datasets [18]. GAVS aims to facilitate intuitive, interactive exploration of Gaia data, empowering researchers to analyze and interpret this information effectively.

GAVS introduces several features designed to overcome the inherent complexity of the Gaia archive, as seen in Figure 2.3. GAVS offers a multi-panel browser-based interface with configurable histograms and scatter plots that allow users to explore various data simultaneously. Another capability of GAVS is its visual querying, which enables users to interact directly with plots, such as selecting regions on scatter plots or histograms, to define areas of interest. The system automatically converts these selections into Astronomical Data Query Language (ADQL) queries, which retrieve the relevant data from the archive, removing the need for manual query writing. Besides this, GAVS incorporates plugins like ALADIN Lite[7] and JS9[8] to provide visualization of skymaps.

GAVS exemplifies best practices in handling astronomical big data, balancing high performance with usability. Its visual query interface aligns closely with the goals of this dissertation by showcasing the power of intuitive, user-centered dashboard design. The tool's ability to integrate

[6] https://gea.esac.esa.int/archive/visualization/
[7] https://aladin.cds.unistra.fr/AladinLite/
[8] https://js9.si.edu

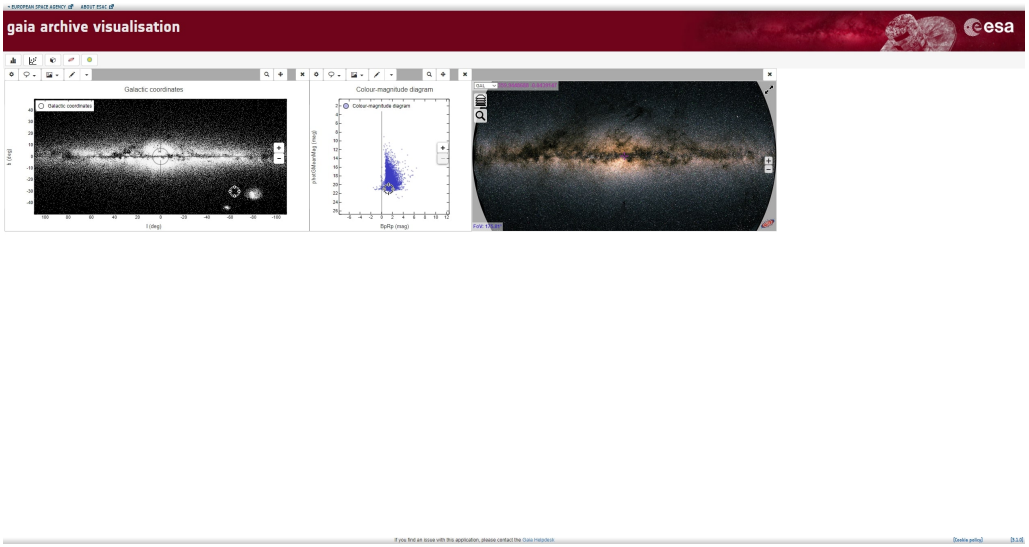


Figure 2.3: A screenshot of Gaia Archive Visualization Service displaying a skymap, a scatter plot with the color-magnitude diagram, and a 3D Skymap using HEALPix

and link multi-dimensional data visualization serves as a benchmark for interactive and scalable scientific exploration.

Despite this, GAVS has a few shortcomings when it comes to design. One example of this is the unclear icons regarding the reset graph and download features. The tool does not use the standard download symbol, such as the arrow pointing down, opting instead for the less clear "image" symbol, and the reset functionality uses a common "move" icon instead of something more appropriate. The interface, besides not being very visually appealing and having a lot of empty space, also has small and closely packed buttons, which not only makes them harder to interact with, but also contributes to a cluttered appearance. This could be improved by increasing icon size and spacing, and logically grouping related tools.

Furthermore, while the graphs themselves are informative, their titles could benefit from better styling to improve readability. A bolder font or increased size for graph titles would make them stand out while ensuring axis labels are easily legible. Additionally, the interface could utilize colors and color contrast better to make the graphs stand out from the background. Overall, these design shortcomings, while not impairing functionality, impact the user experience by making the tool less intuitive and harder to navigate.

2.3.3 ESO Archive Science Portal

The ESO Archive Science Portal[9] provides access to observational data from the European Southern Observatory, including facilities such as the Very Large Telescope (VLT) and the Atacama Large Millimeter/submillimeter Array (ALMA). It is a premier resource for ground-based astronomical data.

[9] https://archive.eso.org/scienceportal/home

The portal, which can be seen in Figure 2.4, offers a range of features designed for exploring and analyzing space data. It includes an interactive skymap using ALADIN, accompanied by a search bar for querying celestial objects. Users can filter datasets by observatory, data type, spectral range, filter/band, spectral resolution, etc., using selection on charts and check boxes. A detailed dataset table provides metadata such as spectral range, signal-to-noise ratio (SNR), observation date, instrument, exposure time, etc., with options for previewing or selecting datasets for analysis. Additional features include real-time dataset counts, target selection using the coordinate systems J2000, and download functionality for selected datasets.

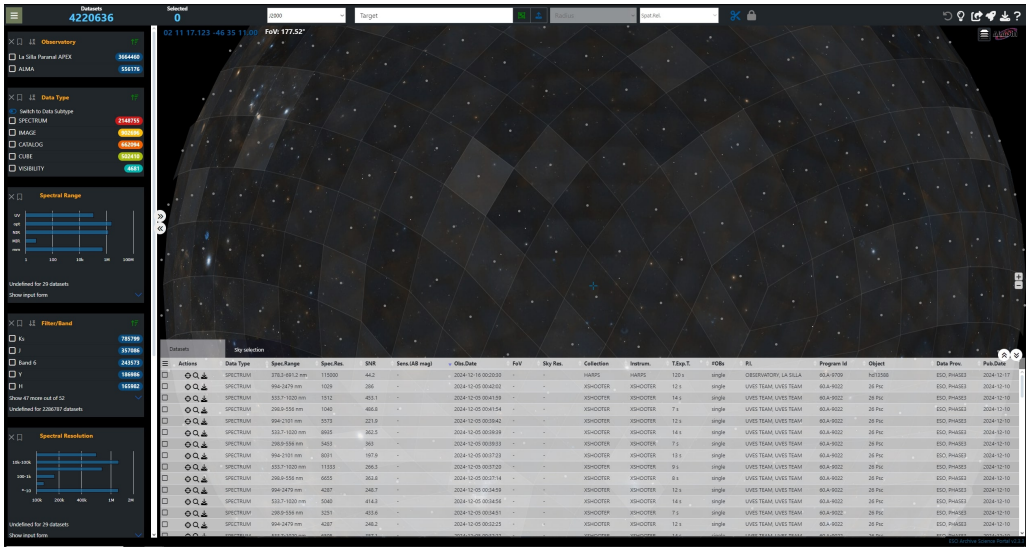


Figure 2.4: A screenshot of the ESO Archive Science Portal interface, showcasing the interactive sky map, dataset filters, spectral range sliders, and metadata table for exploring and analyzing astronomical data.

While feature-rich, the portal has some design shortcomings that can impact usability. The interface appears cluttered, with a dense display of datasets, filters, and the sky view, which can overwhelm users and make navigation challenging. The data table at the bottom of the page is densely packed with information, making it difficult to read and interpret at a glance. Despite the section with tips for use, the absence of clear interactive cues linking the sky view to the data table and the lack of contextual help or onboarding guidance make the interface less intuitive, particularly for new users. The overall color scheme, with its dark background and muted text, poses potential readability issues, especially for users with visual impairments.

2.4 The ELT and Adaptive Optics Telemetry

The Extremely Large Telescope (ELT), a project of the European Southern Observatory (ESO), is currently under construction in Cerro Armazones, Atacama Desert, Chile, which has been proven to be one of the best astronomical sites in the world [7]. The telescope is a major advancement in ground-based astronomical observation. The current projections point to a scientific first light, i.e,

Figure 2.5: The ELT compared to the Padrão dos Descobrimentos in Lisbon, Portugal. [8]

the first observations with scientific instruments, in December 2030. Once finalized, the ELT will be the largest of its kind (optical/near-infrared extremely large telescopes) with a primary mirror of 39 meters in diameter. In Figure 2.5, the telescope is compared to the famous Portuguese landmark, Padrão dos Descobrimentos, in its true size.

For comparison, this size will allow the ELT to collect 15 times more light than the current largest optical/infrared single-telescope, the Gran Telescopio Canarias (GTC), located in La Palma, Canary Islands, which has a primary mirror diameter of 10.4 meters. Additionally, the site in the Atacama Desert where the ELT will be located is near ESO's existing Very Large Telescope (VLT) facility at Cerro Paranal. The VLT consists of four 8.2-meter Unit Telescopes and has been one of ESO's flagship observatories for decades. Still, the ELT will mark a significant leap of 20 times more light-gathering capabilities.

With its site and size, the ELT will allow for the capture of incredibly detailed observations. However, to achieve observations at such high resolutions, the telescope has to face a fundamental challenge: the Earth's atmosphere. Atmospheric turbulence distorts incoming light waves, blurring astronomical images and severely limiting the effective resolution of even the largest telescopes. This is where Adaptive Optics (AO) systems play a critical role.

Adaptive Optics mitigates the effects of atmospheric distortion by dynamically adjusting the shape of deformable mirrors in real time. These corrections occur hundreds or even thousands of times per second. As present in the diagram of Figure 2.6, light from a distant source, such as a star, enters the telescope as a distorted wavefront due to atmospheric turbulence. This incoming wavefront is first directed onto a mirror, which can be adjusted based on the type of wavefront corrector (WFC) present. A beam splitter then redirects part of the light toward a wavefront sen-

sor (WFS), which measures the distortions in the wavefront in real time. These measurements are sent to the Real-Time Control System, which calculates the necessary adjustments and sends commands back to the WFC to make the adjustments accordingly. As a result of this continuous feedback loop, the outgoing wavefront becomes corrected before it reaches the final science detectors, allowing for high-resolution imaging even under atmospheric turbulence.

The scale and complexity of the ELT's AO systems amplify the engineering and operational challenges involved in managing telemetry data. With multiple deformable mirrors and wavefront sensors operating in parallel, the system produces a high volume of data. This telemetry is essential not only for the real-time control loop but also for analysis and fault diagnosis.

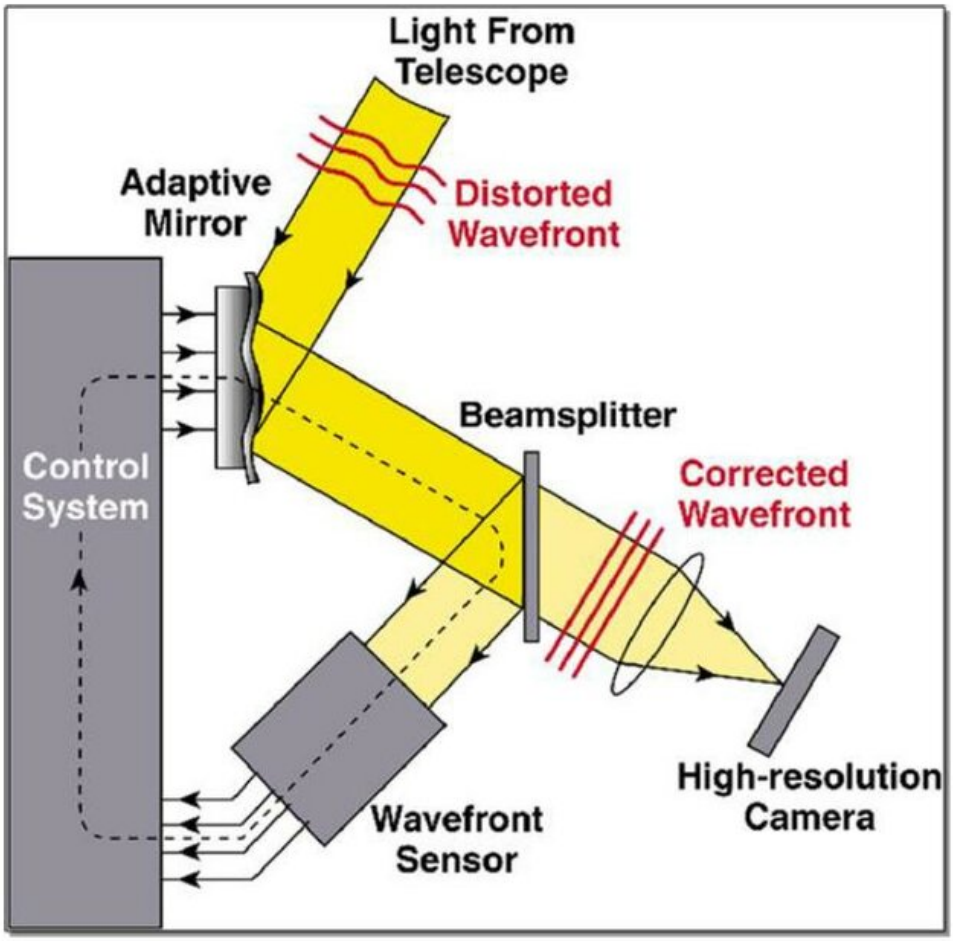


Figure 2.6: Conventional adaptive optics system. [4]

Historically speaking, AO telemetry data was either not kept at all or stored in various proprietary, often undocumented, formats differing between observatories. The lack of standardization in AO telemetry made analyzing the data and developing tools for said analysis quite laborious. Gomes et al. have recently proposed the Adaptive Optics Telemetry (AOT) [12] data exchange format to address this gap in standardization, providing a consistent and well-documented way to store, share, and process AO telemetry across different systems.

The recent introduction of the AOT format not only helps standardize how AO telemetry is stored and shared but also brings clarity to how this data can be conceptually organized. In the context of this dissertation, the AOT telemetry format provides three primary domains for visualization that also reflect the different aspects of the AO System:

- **Pixels:** This domain relates to science detector readings, where each "pixel" corresponds to a single point on the sensor and captures the intensity received at that location at a specific time. In simpler terms, the pixel data is what the science detector receives after the corrections made by the WFC. In the AOT format, this data comes in the format of a 3D array with shape (frames, columns, rows), where each value gives the intensity of a point positioned at (col, row) at a given recording frame.

- **Measurements:** This domain relates to the WFS's measurements. For each frame in time, the sensor captures the slopes of the incoming wavefront along the X and Y axes for each subaperture of the sensor. Essentially, the measurements describe how much the incoming wavefront tilts at different points across the sensor grid, which is used to determine the optical aberrations introduced by the atmosphere or the telescope lens. In the AOT format, this data is stored as a 3D array with shape (frames, 2, indexes), where, for each subaperture index at a given recording frame, there is one value for each slope component (X and Y). A WFS subaperture mask may be provided to reconstruct this data.

- **Commands:** This domain relates to the commands sent to the WFC to change the wavefront. For each frame in time, the system calculates and applies a set of command values that adjust the shape of the mirror to correct for the measured distortions. In other words, the commands represent the real-time corrections applied to compensate for atmospheric turbulence and optical errors. In the AOT format, this data is stored as a 2D array with shape (frames, index), where each value corresponds to the command for a specific actuator index at a given recording frame. For a Deformable Mirror (DM), a specific type of WFC, this data can be reconstructed into a 3D array if an influence function is provided in the file.

The combination of the ELT's scale, the complexity of its AO systems, and the introduction of the AOT telemetry format highlights both the opportunities and challenges in visualizing and analyzing AO telemetry data. As telescopes continue to grow in size and capability, the need for visualization and analysis tools is becoming an ever-present part of the adaptive optics field.

2.5 Discussion

The research conducted highlights a central reality: while visualization has advanced considerably in recent years, the specific challenge of exploring and analyzing AO telemetry data remains largely underexplored.

Existing tools like ALADIN, GAVS, and the ESO Archive Science Portal demonstrate that the community values interactive visual environments for data analysis. These platforms showcase

how interfaces can support complex data exploration to promote research and discovery. However, their scope focuses predominantly on observational data rather than the more low-level telemetry, which, up until recently, was considered more "engineering data".

Furthermore, the review of interaction design, dashboard categories, and visualization taxonomies makes it clear that most existing astronomical visualization platforms either lean towards static data products (as seen in ALADIN) or focus on query and selection of observational datasets (as in GAVS and the ESO Portal), none of the tools addresses the need for frame-based visualizations of data. In reality, no tool that enabled the exploration of AO telemetry was found.

In this context, the recent introduction of the AOT format represents an important step toward data standardization within adaptive optics telemetry. This alone is a major step toward aiding the development of tools to fill the gap. However, the introduction of a standard format alone does not solve the challenges of practical exploration. The volume and dimensionality of AOT data introduce significant demands both at the interaction level (how users navigate and filter large sequential datasets) and at the rendering level (how systems maintain performance when visualizing multi-gigabyte telemetry files).

In terms of visual design, the research offered valuable insights into visual choices and user-focused design, which was rarely mentioned in the context of space-related tools.

Ultimately, the gap identified is twofold. Firstly, there is a lack of domain-specific visualization tools tailored to the nature of AOT data, and secondly, the absence of well-defined solutions that accommodate AO telemetry's frame-sequential, multi-domain structure while still aligning with established best practices from the dashboard visualization research.

This dissertation aims to bridge this gap by proposing and developing an interactive visualization system, i.e., a dashboard, designed explicitly for exploring AOT data, with special consideration for the future adaptive optics that the ELT will produce on large scales. The work draws some inspiration from the existing dashboards explored, particularly GAVS, for its more similar approach to visualization, and from well-established paradigms of effective data and information visualization. However, it adapts and extends them to fit the unique demands of AO telemetry and its scale.

Chapter 3

Requirement Elicitation and Definition

A successful software system is grounded in a deep understanding of its users' needs, goals, and operational context. In projects involving scientific tooling, such as this adaptive optics telemetry dashboard, requirements gathering plays a pivotal role in shaping a solution that is both usable and technically robust. This chapter presents the process used to elicit, document, and validate the software requirements for the dashboard. It also highlights how these requirements guided early architectural decisions and design priorities.

The chapter is organized into four main sections. First, it outlines the elicitation process, including the methods used and the stakeholders involved. It then presents the approach to formalizing the gathered requirements, including their categorization and prioritization. Following this, it describes how these requirements were validated with domain experts. Finally, the chapter discusses specific high-impact requirements and how they influenced key design choices for the dashboard.

3.1 Requirement Elicitation

Before designing any software, a crucial step is understanding the stakeholders' needs and pain points. Without first listening and interacting with the people who depend on tools, we risk solving non-existent problems. Thus, the first stage of this process is requirement elicitation, where an effort is made to capture what researchers and developers need from an adaptive optics data exploration tool.

Since this dashboard is still a prototype, and not a fully operational and deployed tool, and the number of qualified users is small, the participants for requirement gathering were few. When choosing the appropriate gathering technique, this is an essential factor to consider. For this reason, some methods were quickly discarded, such as multi-participant workshops. If the project scope were larger, other techniques, such as extensive surveys, could be used to reach a larger base, but that is not the case with this project. Other factors, like schedule issues, also limit the viability of smaller-scale workshops. With these considerations in mind, two major techniques were used to obtain requirements.

The chosen approach combines guided observation with think-aloud narration and a semi-structured interview in a single sitting. The plan started by having the participants use the existing tool with a preset file, while verbalizing what they like and do not like about the tool. The intent was to expose major pain points while also understanding their view of possible improvement. This initial observation also allowed for a good transition towards the semi-formal interview conducted afterwards, where the participant answered a prepared set of open questions regarding features, priorities, and constraints, and explored in more depth the themes raised during the walk-through.

Two elicitation sessions were conducted: one with an AO researcher who performs data analysis and is the intended user of the dashboard, and one with an AOT file developer who designed the data exchange format. Each session was intended to be one hour. Still, the latter extended to around 4 hours, as the developer kindly gave an extensive overview of how adaptive optics works, why the data format is significant, and how to use the aotpy library created to facilitate access to the data format.

Although the sample of participants is small, the pair covers both ends of the workflow: scientific interpretation and data generation. Therefore, their input is sufficient for a first requirements baseline. During the sessions, notes were taken, ensuring no details were missed or forgotten before moving to the next step of documenting the requirements.

3.2 Formal Documentation of Requirements

Documenting requirements is essential in establishing a common understanding of a product's scope and features amongst stakeholders. Formal documents provide a detailed and structured reference that guides the development process and ensures users' expectations align with the project goals. This document also aids in minimizing ambiguities and in a more accurate implementation and verification process, as everything is objectively stated and labeled.

The documentation approach followed industry standards by closely aligning with the ECSS-E-ST-40C Rev.1 guidelines [6], specifically for a Software Requirements Specification (SRS) document. The process entailed structuring the requirements into clearly identifiable functional and non-functional categories, ensuring consistency and comprehensiveness. Given the extensive nature of the requirements, this section will only discuss key highlights of the document. The complete SRS document produced during this project stage is available in Annex A.

The requirements fell into two primary groups: functional and non-functional requirements. The functional requirements detail the behaviors and functionalities expected of the software, and, for this reason, each subsection focuses on overall data handling, user management, and features per page. On the other hand, the non-functional requirements, which outline the system's capabilities and constraints, were sectioned into the most salient capabilities the system must have.

Besides categorizing and labeling requirements appropriately, using "shall" or "may" becomes imperative in reducing mismatched expectations, especially regarding implementation priorities. If a requirement is deemed essential for the dashboard's success, the term "shall" indicates the

feature is mandatory, and thus must be present even in terms of a minimum viable product. In opposition, "may" indicates that a feature is optional. While an optional feature may enhance the dashboard, it is not strictly necessary for the tool's basic operations. Prioritizing these optional features should occur only after addressing other, more critical features.

3.3 Requirements Validation

Building upon this requirements-driven design process, the next phase involved validating the accuracy and relevance of the requirements produced. Given the limited but highly specialized pool of stakeholders involved in the elicitation process, their feedback was critical in establishing whether the document reflected the stakeholders' needs and expectations.

To carry out this validation, the SRS document was shared with the stakeholders who participated in the elicitation phase. Each was asked to review the document and identify possible ambiguities, omissions, or misinterpretations of the insights they had previously provided. Their feedback led to several minor but meaningful adjustments, such as refining the clarity of specific requirements and adjusting the priorities of certain features, such as the Sandbox page.

Following this review and revision process, both stakeholders approved the final version of the document, which provided confidence that the requirements were grounded in real user needs and domain-specific constraints. While this validation step helped ensure that the right problem was being addressed, verifying the implemented system against these requirements is part of the broader verification and evaluation process, and thus is further discussed in Chapter 6.

3.4 Key Requirements and Their Impact on Dashboard Design

The structured documentation and its clear categories and priorities served as a foundation for identifying key challenges and areas that would require special attention during implementation. While this section highlights vital requirements and requirement-driven considerations, the specific solutions and architectural decisions to address them will be discussed in Chapter 4.

One of the most important themes from the document was the performance under large data loads. As specified in the requirement **NFR-PRF-01**, the system had to be able to parse large files quickly, which impacted how the backend dealt with data handling to avoid performance bottlenecks. Similarly, the frontend rendering pipeline needed to be optimized for speed and responsiveness, especially when dealing with high-volume visualization tasks, as captured in **NFR-PRF-03**, which dictates chart redraw latencies.

The need for multi-domain data representation (Measurements, Pixels, and Commands, as defined in Sections 5.3 to 5.5 of the SRS) led to a clear separation of concerns in the application architecture. Each domain-specific section required its dedicated components and API endpoints, contributing to a modular, component-based UI. This also affected how state management was handled across the frontend, prompting a design that supports cross-component synchronization while maintaining responsiveness.

The user-driven interaction model, captured in requirements like **FR-MSR-20**, **FR-PXL-22**, and **FR-CMD-19**, which require the system to respond to user clicks by displaying point-specific statistics, influenced the dashboard's event-handling and interaction tracking logic. This pushed the development towards an architecture that could quickly query and visualize localized data points without reloading entire datasets.

Finally, the non-functional constraint of using only open-source technologies (**NFR-CST-01**) and the requirement to use the `aotpy` library (**NFR-CST-02**) narrowed the selection of backend tools and visualization frameworks. This had a tangible impact on technology stack decisions and integration workflows.

By closely mapping these key requirements to design choices, the resulting dashboard remains well-aligned with stakeholder expectations and project constraints.

3.5 Summary

This chapter detailed the end-to-end process of eliciting and formalizing the requirements for the adaptive optics telemetry dashboard. Through targeted stakeholder interviews and observational methods, the project identified essential functional and non-functional needs, which were then documented in a structured format. The resulting SRS document, available in Annex A, served as a critical reference throughout development and was validated by domain experts to ensure its relevance and accuracy.

Several key requirements directly influenced major design and implementation decisions. These included performance considerations for large datasets, multi-domain visualization needs, user interaction capabilities, and the constraint of using open-source technologies. The alignment between these requirements and the dashboard's architecture detailed ensured that development efforts were focused on delivering a tool that meets expectations. The next chapter builds upon this foundation by discussing how these requirements shaped the system's overall design and architecture.

Chapter 4

System Architecture and Visual Design

This chapter describes the overall architecture and design decisions made during the idealization of the AOTrack Dashboard. It covers both the software architecture and the user interface design process. Together, these aspects illustrate how technical implementation choices were shaped by the requirements identified earlier.

4.1 System Overview

The AOTrack Dashboard adopts a client-server architecture with a clear separation of concerns between the frontend user interface and the backend data handling services. This separation ensures server-side execution of computationally intensive data processes, while the client-side remains focused on presenting the data and allowing the users to interact with the interface smoothly.

The architecture follows a modular design, aligning with established best practices for dashboard detailed throught Section 2. The frontend communicates with the backend exclusively via HTTP-based API endpoints, simplifying deployment and scalability while enabling frontend-backend decoupling.

Figure 4.1 illustrates the high-level architecture of the AOTrack Dashboard, following the Logical View of the "4+1 View Model" [15] proposed by Philippe Kruchten. The diagram presents the main system modules and their dependencies. The User Interface and Visualization layer manages user interactions and data presentation, triggering session management and telemetry file processing operations. The `Session and State Management` module maintains user session context and tracks uploaded files. `Telemetry File Processing` handles the parsing of AOT files, relying on the previously mentioned `aotpy` library. The processed data then passes through the `Data Processing and Transformation` module, which is aggregated and formatted for visualization. This diagram emphasizes the dependency chain between modules.

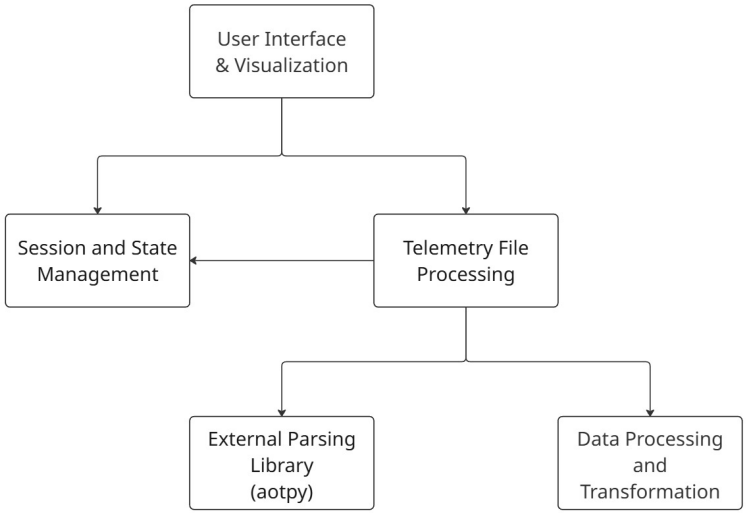


Figure 4.1: Logical View of the Dashboard AOTrack

4.2 Technology Stack

4.2.1 Frontend

The frontend uses React [1] as the main JavaScript library, due to its flexibility, component-based architecture, a primary necessity identified in the requirements phase, and strong ecosystem support, making it well-suited for building a modular and highly interactive dashboard. Vite is the tool for project setup, building convenience, and faster iteration and deployment cycles. For styling, shadcn/ui [2] combined with Tailwind CSS [3] enabled rapid and consistent user interface development. The former provided highly customizable components due to its "Open Source. Open Code" nature, while the latter provided a more utilitarian framework for consistent styling. To meet the visualization demands of the project, D3 [4] was used for its fine-grained control over complex and custom data-driven visualizations. Alternatives such as Streamlit [5], Dash [6], and Grafana [7] underwent assessment during the design phase but were ultimately rejected due to several concerns.

Dash, while is based on Python [8], which is important as discussed in the following paragraphs regarding backend, and is convenient for simple data apps, follows a primarily stateless architecture, meaning that any UI interaction typically requires a round trip to the server, making it inefficient for highly interactive applications with frequent state changes, as is the case. In contrast, React, which has simple ways to create state management and context-sharing mechanisms,

[1] https://react.dev
[2] https://ui.shadcn.com
[3] https://tailwindcss.com
[4] https://d3js.org
[5] https://streamlit.io
[6] https://dash.plotly.com
[7] https://grafana.com
[8] https://www.python.org

some of them built-in, enables local handling of UI state and interactivity without constant backend involvement.

Streamlit was also seriously considered, given its Python-first approach and ease of development. However, concerns arose regarding its ability to handle large-scale data processing efficiently, especially under scenarios involving multiple concurrent users. Streamlit's session-based model and server-side rendering could introduce bottlenecks when working with high-frequency telemetry data or when attempting to deploy the application in a more distributed, production-grade environment. Given the project's long-term goal of scalability and responsiveness, React offered a more future-proof solution, providing granular control over rendering logic, frontend performance optimization, and the flexibility to build a highly interactive and responsive dashboard.

Grafana was also a potential solution, given its strong reputation for data visualization and usage in astronomy. However, its architecture is primarily optimized for continuous data streaming and live telemetry sources, rather than for systems centered around file uploads and on-demand data processing. Since the AOTrack dashboard focuses on displaying uploaded files, Grafana's design presented a mismatch with the project's scope.

Other JavaScript visualization tools besides D3, namely Plotly.js [9] and Chart.js [10], entered the ideation process, but both fell short when meeting the dashboard's needs. Plotly.js, while capable of producing interactive charts, relies on predefined chart types and expects the entire dataset to be passed up front for each rendering cycle. This limits control over memory management when choosing what to render, making it unsuitable for visualizing large telemetry datasets, as is the case for this project. Chart.js, primarily focused on simple chart types like bar and line graphs, offers even less flexibility and lacks low-level control over data handling and rendering logic. In comparison, D3.js provides complete control over the visualization, both in terms of what data to load, when to load it, and how to render it. Overall, the combination of React and D3 offered more control over rendering logic and UI customization than its counterparts.

4.2.2 Backend

The backend system uses Python with FastAPI [11] and Uvicorn [12] as the ASGI (Asynchronous Server Gateway Interface) server. Python was a natural choice for the backend programming language since the requirements mandate using the Python library aotpy [13] as the primary tool for reading and processing AOT files. This dependency was a key architectural driver, ensuring the backend could efficiently handle the specialized file formats used in the project.

For the web framework, FastAPI was chosen in combination with Uvicorn for its asynchronous capabilities, automatic data validation with Pydantic, and OpenAPI documentation generation, all of which facilitated the development of a simple, highly concurrent, well-documented API layer.

[9] https://plotly.com/javascript
[10] https://www.chartjs.org
[11] https://fastapi.tiangolo.com
[12] https://www.uvicorn.org
[13] https://pypi.org/project/aotpy

Compared to Django [14] and Flask [15], which were other web framework options, FastAPI provided many benefits. Compared to Django, which is more geared towards server-rendered websites and full-stack development and not designed for high concurrency, FastAPI is a much lighter, more scalable solution for a dashboard in which performance is imperative. Django's limited asynchronous support makes it less suitable for low-latency API interactions. On the other side of the spectrum, Flask follows a more minimalist micro-framework philosophy, providing only basic routing and request handling capabilities. While this approach offers flexibility, Flask, just like Django, lacks native support for asynchronous capabilities, which, if adopted, would imply the need for significant boilerplate to implement features that FastAPI already does well.

4.3 Data Flow

Understanding the data flow of the dashboard is essential for further explaining the system and its user-facing behavior. The software follows a request-response pattern, supported by both session and state management, and temporary file storage to handle large files efficiently.

The data flow begins when the user uploads a .fits file through the `UI`. This upload triggers an HTTP request from the `Frontend API Client` to the `Backend API`'s upload endpoint. Upon receiving the file, the backend validates the upload, parses the file using the `aotpy` library, and extracts relevant metadata such as telescope name, mode, wavefront sensors, detectors, and correctors. This metadata and parsed data are stored in a `Temporary File Storage` area on the server. The `Session Manager` is handled by the backend, which ensures that each user interaction references the correct dataset and maintains state consistency across multiple requests. Sessions are automatically cleaned up after a predefined period of inactivity to optimize server resources.

Once the file is processed and the session is initialized, the backend responds with the previously mentioned summary metadata in JSON format. This information populates the frontend `UI`, allowing users to interact with various visualization components such as heatmaps, line charts, and histograms.

As presented in Figure 4.2, which is part of the Process view of the "4+1 View Model" [15], subsequent user interactions, such as adjusting visualization parameters, navigating through frames, or selecting different domains (Measurements, Pixels, Commands), trigger additional HTTP requests. These requests are also handled by the `Frontend API Client` and forwarded to the appropriate backend endpoints. The backend validates each request against the active session state and processes the data as needed. After processing, the backend returns the requested data, which the frontend uses to update the visualizations in real time.

The system also includes error handling at multiple levels. Upload validation checks ensure that only correctly formatted AOT files are processed. Session validation prevents users from making data requests against expired or non-existent sessions. `Backend API` endpoints return

[14] https://www.djangoproject.com
[15] https://flask.palletsprojects.com

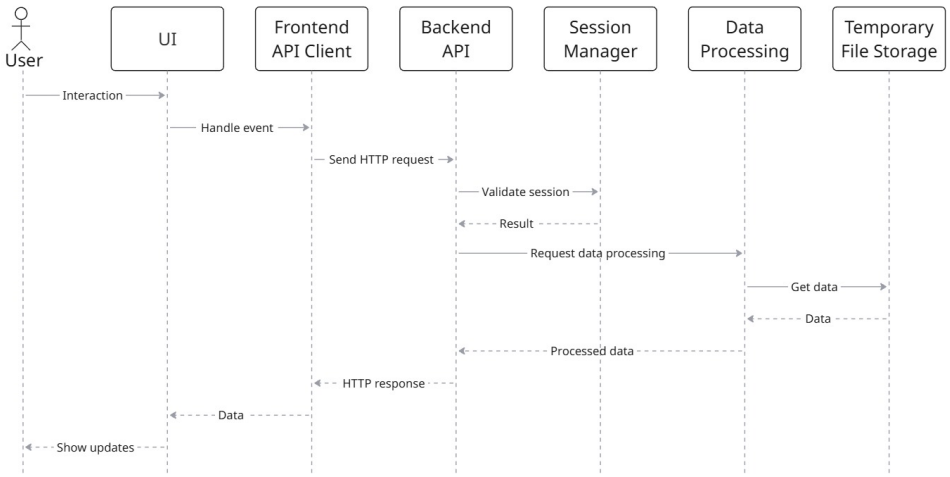


Figure 4.2: Process View of the Dashboard AOTrack

appropriate HTTP error codes and JSON error messages for invalid requests, which the frontend can parse and display as user-friendly error notifications.

4.4 Visual Design and Prototyping

Designing the user interface is just as critical in a software project's planning stages as defining the underlying architecture. While the system architecture ensures the technical feasibility and performance of the dashboard, the visual design process focuses on carefully defining the UI, ensuring all the requirements can be met during implementation, and fewer usability issues appear down the line. While the high-fidelity mockups discussed in this chapter guided the overall design direction described in 5, minor adjustments were made during implementation to better address performance considerations and user feedback encountered during development.

4.4.1 Design Process and Decisions

The user interface design process combined low-fidelity and high-fidelity prototyping stages. Initial brainstorming and hand-drawn sketches were used to quickly explore layout concepts and navigation structures. These sketches provided a flexible environment for experimenting with different design ideas without the constraints of software tools. Following this initial phase, more refined, high-fidelity mockups were created using Figma [16], allowing for detailed visualization of user flows, UI element placement, and overall aesthetic considerations.

Throughout the design process, insights gathered during requirement elicitation played a crucial role. For example, stakeholder feedback highlighted the importance of hierarchical navigation

[16] https://www.figma.com

and interactive data exploration. This led to the inclusion of features like clickable heatmaps and dynamically updating charts, aimed at making the dashboard feel responsive and data-driven.

Accessibility was also a consideration, especially regarding color selection for both the website color palette and the charts, ensuring that visualizations remained interpretable to all users, including those with vision impairments. To achieve this, color choices were evaluated using online accessibility tools. One such tool was "Who Can Use" [17], which allows testing color pairings against different types of color blindness and conditions such as glaucoma and cataracts. The tool checks the compliance with Web Content Accessibility Guidelines standards [31], and only color combinations that achieved at least a WCAG AA rating for visual contrast were selected for the final designs. This proactive approach helped reduce the risk of introducing inaccessible color mappings and ensured that key visual distinctions would remain visible and interpretable across a wide range of users.

As mentioned in Section 2.2.2, there is also a need for clear functional designs with minimal clutter, as per Tufte's data-ink ratio concept. With limited screenspace and high volumes of data, there is a need to mitigate the issue. This can be done in several ways. The first way this issue was addressed was by choosing a multi-page layout. While not necessarily ideal, having a multi-page dashboard allows the design to be more readable and to have a better data-ink ratio. Another way this was done was by having a simple yet effective collapsible sidebar. While this is reasonably common in dashboards, it is still important to recognize why it is so prevalent.

Regarding layout, control elements such as the control bar for charts, which was also collapsible to ensure maximum space while maintaining functionality, were placed near their associated visualizations to reduce confusion when adjusting settings. The visual organization of charts aimed to support quick recognition of importance, positioning the main chart on the left, as also discussed in Section 2.2.2.

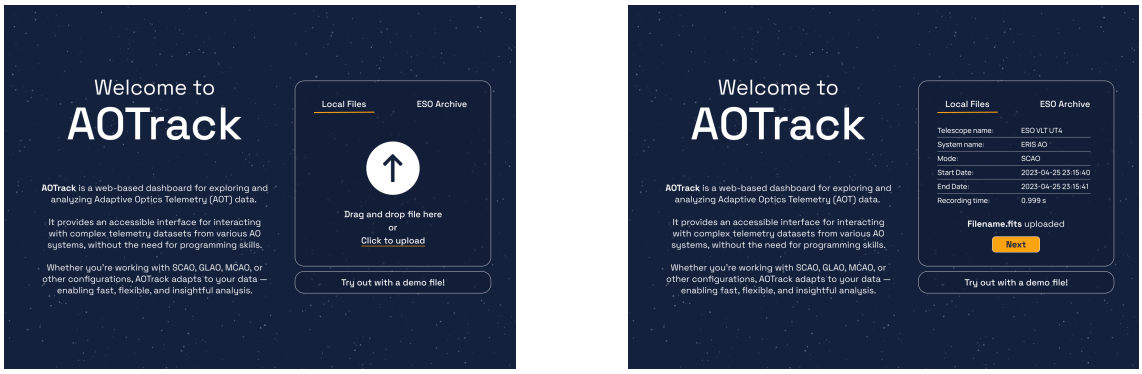
Some of the design choices discussed may seem trivial initially, but they can often be overlooked as obvious and sometimes even forgotten. However, it is important to acknowledge that each decision, no matter how small, contributes to the overall usability and accessibility of the system. By making a conscious design to address these details during the design phase, the project can avoid potential issues during its development.

4.4.2 Final Layouts and User Flow

The result of a couple of iterations of the design process led to the creation of the mockups present in the Figures 4.3, 4.4, 4.5, 4.6, and 4.7.

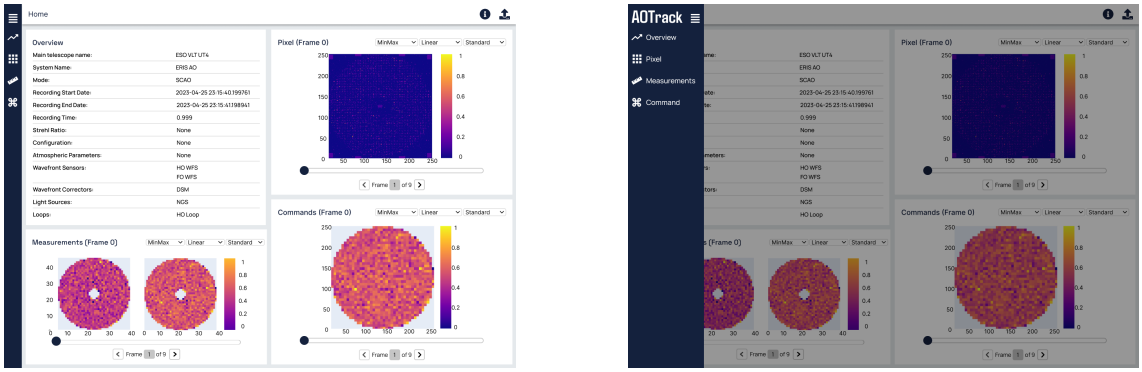
The expected user journey begins on the Landing page in Figure 4.3, where users can upload AOT files from local storage or access data directly from the ESO Archive. After a successful file upload, the user transitions to the Overview section, where key dataset metadata is displayed along with main components from each other page, which are accessible through the sidebar navigation.

[17] https://www.whocanuse.com



(a) (b)

Figure 4.3: Mockups for the Landing Page: (a) Landing page with no uploaded file. (b) Landing page with file uploaded and preview showing.



(a) (b)

Figure 4.4: Mockups for the Overview Page: (a) Overview page when opened. (b) Overview page showing the collapsible sidebar open.

There are three domain-specific pages: Measurements, Pixels, and Commands, whose mock-ups are present in Figures 4.5, 4.6, and 4.7 respectively. Users can interact with control elements within each domain-specific page, like dropdown menus for data scaling, interval selection, and color map changes. Frame navigation is also supported in all pages through a frame slider and navigation buttons, allowing users to move through the data smoothly. Once a frame is selected, the system updates the displayed visualizations accordingly, as described in Section 4.3.

Given the dashboard's exploratory data analysis nature, mapping out how each UI element and chart would respond to inputs like point selection, frame changes, and parameter adjustments, even when the interaction is in another element, was a key focus. This mapping helped ensure the layouts, features, and interactions aligned with the various functional requirements described in the SRS document.

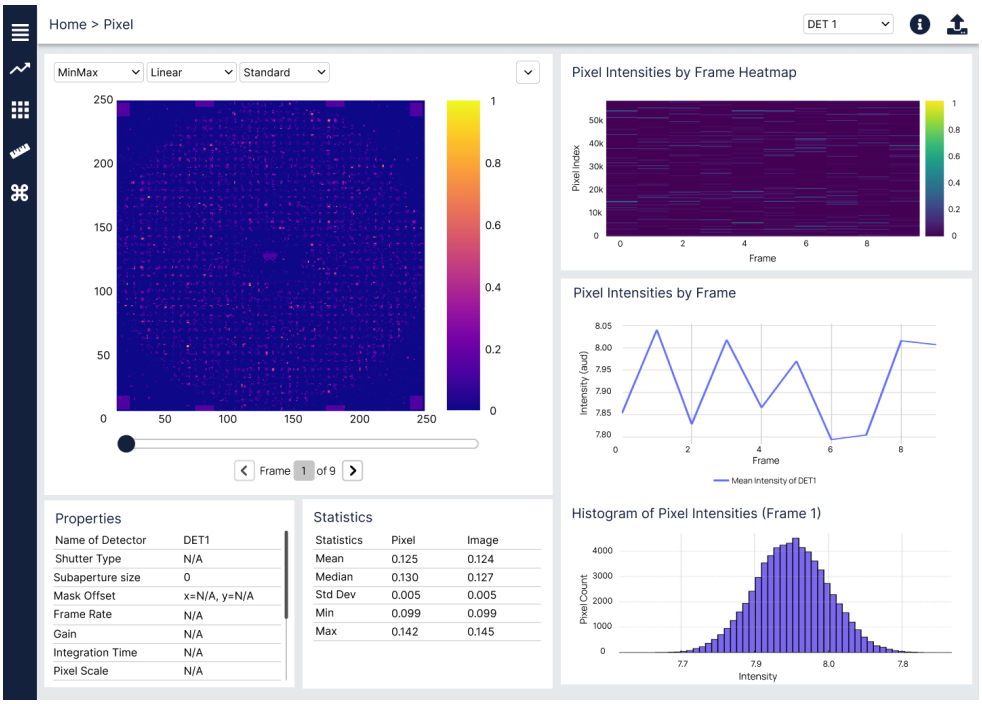


Figure 4.5: Mockup for the Pixel Page

4.5 Summary

This chapter detailed the architectural and design decisions behind the AOTrack Dashboard, explaining how the system was structured to meet the technical and usability requirements previously identified. The dashboard follows a client-server model, separating backend processing from frontend interaction to ensure responsiveness under large data volumes. The logical and process views presented in this chapter, following the "4+1 View Model", illustrated the modular structure and the flow of data across components.

The chosen technology stack, comprising React, D3.js, FastAPI, and aotpy, was carefully selected to balance performance, scalability, and maintainability. Alternatives such as Dash, Streamlit, and Grafana were considered but ultimately rejected due to architectural mismatches or performance concerns.

The chapter also outlined the user interface design process, emphasizing accessibility, clarity, and responsiveness. It presented the prototyping journey from sketches to high-fidelity Figma mockups and highlighted the importance of design principles such as the data-ink ratio and interface modularity. The resulting layout and user flow were designed to support domain-specific navigation and intuitive data exploration.

Together, these design and architectural choices formed a robust foundation for the dashboard's implementation, which is discussed in more detail in the following chapter.

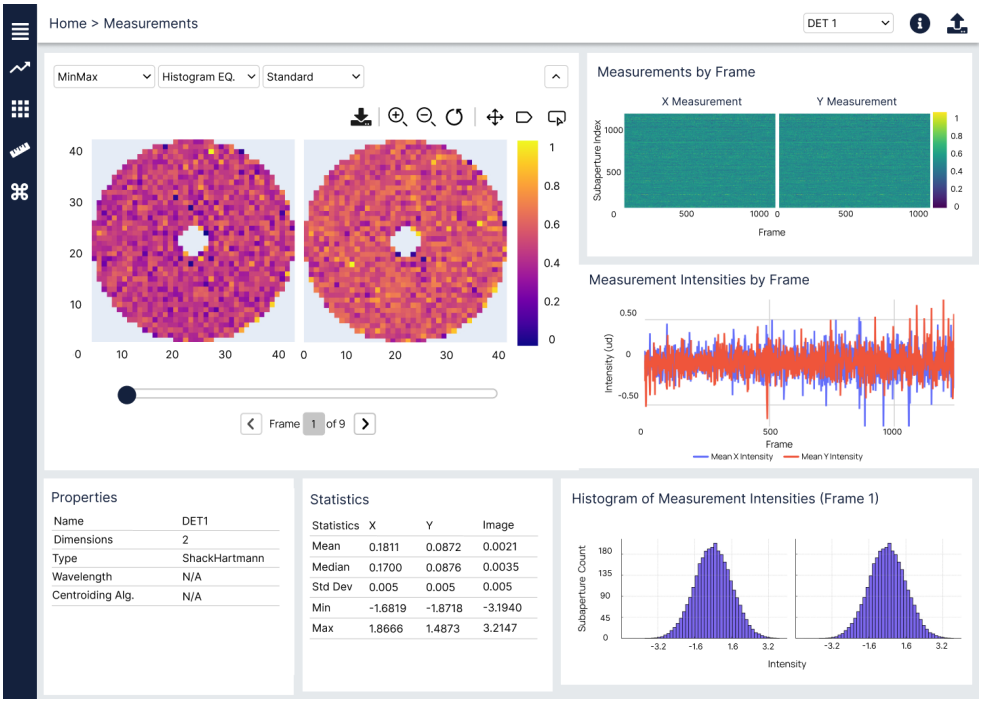


Figure 4.6: Mockup for the Measurements Page

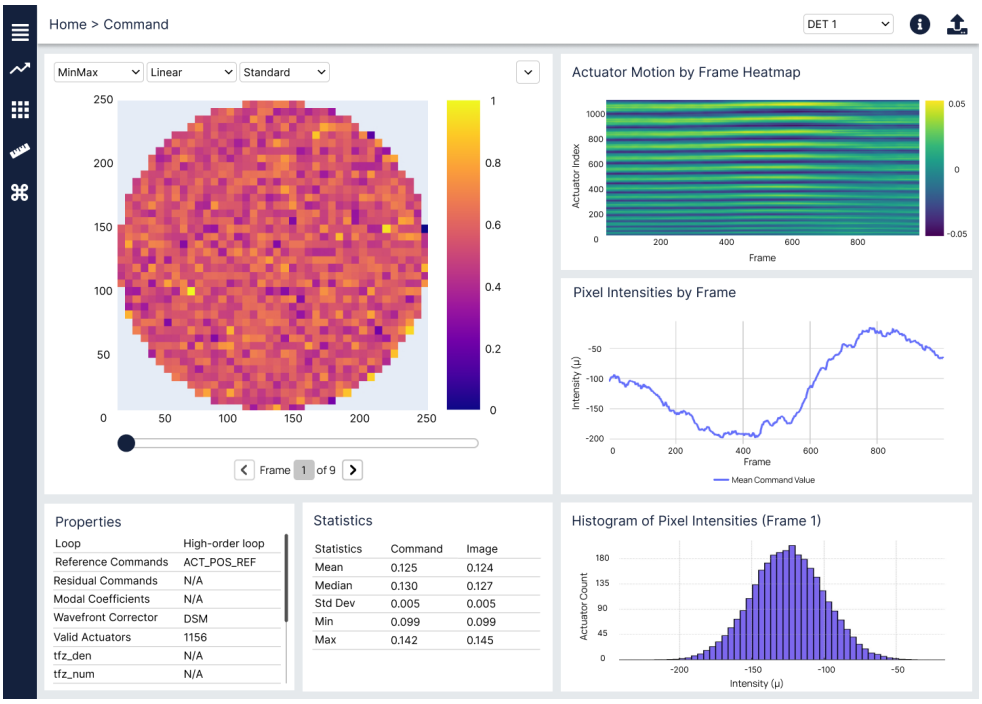


Figure 4.7: Mockup for the Commands Page

Chapter 5

Implementation

The AOTrack Dashboard implementation had as its main outcome objective an interactive web-based platform for exploring AO data. The implementation is the culmination of every previous step taken. Nevertheless, following rigid guidelines can stunt improvements, so this development followed an iterative and problem-driven approach, with continuous refinement based on technical experimentation, performance constraints, and user feedback. Rather than following a strictly linear software development model, the implementation evolved in parallel with some design decisions, often requiring adaptation as new challenges emerged.

User feedback played a critical role in guiding some implementation decisions. Informal sessions were held regularly so prospective users could see early builds and provide input. This feedback loop led to several changes in functionality and design, helping align the system with user needs as it progressed.

This chapter will focus on the most salient implementation details, with a heavy focus on particularly challenging issues, considerations made, and why decisions were taken.

The repository for the code of this implementation is available at https://github.com/RitaBaptistaOliveira/AOTra

5.1 Data Handling

Handling large telemetry datasets efficiently was the challenge throughout the implementation. As this dashboard is entirely focused on displaying large amounts of data in a smooth, but interactive fashion, most challenges emerged as a result. When users upload an AOT file, the backend processes the file using the `aotpy` library. Metadata and dataset summaries are extracted and stored temporarily in backend session memory, reducing redundant parsing for repeat queries within the same session.

5.1.1 File-Based Storage vs. Database

From the start, an important design decision involved how the server would access and manage uploaded AOT files. Rather than importing data into a database, the system was designed to read directly from the uploaded file stored temporarily on disk. Several factors drove this decision.

It has been established throughout this dissertation that AOT files can be large, often several gigabytes, and are sequential and frame-based. Storing every frame, every pixel, and every measurement into a database would require designing a custom schema capable of handling this data efficiently. In contrast, keeping the file on disk and reading from it on demand avoided this bottleneck. The system could extract just the needed data slice during each frontend request.

While database storage may benefit long-term archival or multi-user spaces in future system versions, the temporary session-based file handling approach was more appropriate for the project's immediate needs, considering the idea was a session-based exploration. Setting up a whole database for short-lived, single-session data would have been over-engineering. Furthermore, given the limited project timeline, implementing and tuning a scalable database solution capable of handling such telemetry files would have diverted focus from the visualization needs of the user.

5.1.2 Moving Frame Buffer

During the initial phases of frontend development, the backend adopted a simple approach to data serving, designed to speed up early visualization testing. For example, one of the first features developed, the pixel frame visualization, relied on loading the entire 3D array of pixel values across all frames and sending it to the frontend in a single API response. This method worked adequately for small test files, allowing rapid development of frontend rendering logic without complex data-slicing mechanisms. However, as development progressed and the dashboard began to integrate additional visualizations, this strategy quickly became problematic.

With multiple components requesting large datasets simultaneously, browser memory usage increased dramatically, causing performance degradation and, in some cases, complete crashes. Network transfer times for substantial payloads also became a bottleneck, slowing down the overall user experience.

In response to these limitations, the data loading strategy had to be redesigned around a moving frame buffer approach. Similarly to how video streaming works, instead of sending complete datasets, the backend now delivers a limited range of frames centered around the user's current frame of interest. For instance, when a user views a specific frame in the Pixel Page, the backend sends a small window of neighboring frames. As the user moves through the frames, new frames at the edges are loaded, while frames outside the window are discarded from the frontend memory. This not only ensures the dashboard remains responsive because only a fraction of frames are held in memory at once, but also serves as a simple way to allow for smooth and flicker-free visualizations.

5.1.3 WebSocket Considerations

At a particular stage of development, the idea of using WebSockets to communicate between the frontend and backend was explored. The aim was to enable real-time streaming of telemetry

frames, especially for components like the dynamic heatmaps that could benefit from continuous updates.

However, implementing WebSocket handlers in FastAPI, especially with proper error handling, session management, and control for large payloads, would have introduced significant additional complexity. Additionally, early frontend implementations revealed that efficiently rendering and visualizing the large datasets in the browser was a more urgent performance bottleneck than network delivery speed. There was no point in optimizing the delivery pipeline if the frontend could not render data fast enough once it arrived.

Given limited time and resources, the decision was made to focus on optimizing data transformation and rendering performance first, leaving WebSocket integration as a possible future enhancement.

5.2 Rendering

Once again, the large scale of telemetry data leads to possible issues discussed in this section, but now in terms of rendering. One of the most technically challenging aspects of the dashboard implementation was how to make an interactive, responsive, and resource-efficient way to render such high volumes of data.

5.2.1 Graphics: Canvas, SVG or WebGL

Given the possible issues rendering could cause, choosing the right rendering technology and strategy became critical, as it could make or break the dashboard. Three technologies were evaluated before selecting the final implementation approach.

SVG was initially considered because it is quite simple to use, has CSS styling, and is the usual choice when using with D3.js. With D3, SVG makes most features related to interactivity and styling straightforward. However, SVG's performance limitations became clear during early experiments, especially for large-scale visualizations. Rendering thousands of individual elements (such as pixels or data points) led to excessive memory use and significant frame rate drops.

Canvas emerged as the next alternative due to its better handling of a large number of objects. Unlike SVG, which creates DOM nodes for each graphical element, Canvas renders all graphics within a single bitmap context. This made it well-suited for high-density heatmaps. The final implementation relied heavily on Canvas for most custom visualizations. This choice allowed the optimization of the rendering and easier manipulation of data. However, there is one clear downside of using Canvas: no native support for interactions. This means that any events related to interaction the user made with graphs, as is the case for clicking, using the mouse wheel, and using keys, had to be implemented manually and coordinated across the multiple charts, as one of the key requirements of this project was cross-chart interactions.

Despite the implemented solution being Canvas, another option appeared later in development and, due to the project's time constraint, could not be fully explored. WebGL was not considered

during the early implementation stages, as it is generally more complex to implement and, while capable of better performance, comes with even higher development overhead than Canvas. Moreover, a lack of a well-optimized WebGL implementation can result in worse performance than a carefully constructed Canvas solution.

However, the idea of using PixiJS [1], a rendering library built on WebGL also known as Pixi,, emerged partway through the project as a way to make WebGL more approachable without requiring low-level programming. Pixi provides a higher-level WebGL renderer that abstracts much of the complexity of raw WebGL development, potentially allowing faster drawing speeds for large datasets while offering flexibility.

By the time this option was identified, though, significant parts of the dashboard's rendering logic had already been built and optimized around Canvas, making a late-stage technology switch impractical. Still, PixiJS remains an interesting candidate for future iterations of the dashboard, particularly if scalability demands continue to grow. Of course, any future transition would require careful benchmarking and testing to ensure that the expected performance gains from WebGL would work for the project's specific characteristics.

5.2.2 Tile-Based Rendering

Another issue caused by the large dataset appeared when trying to keep in memory and render a specific 2D dataset in a heatmap. For the pixel data, transforming the 3D arrays with shape (frames, column, row) into the shape (frame, column * row), where the new value is the pixel index, can be quite useful to check specific pixel intensities across frames, as explained by the stakeholders during the requirement elicitation sessions. One of the examples they gave for why this can be useful is the detection of "dead" pixels that are full intensity for the entire recording duration. The challenge comes when some resolutions are on the order of hundreds of frames by over 15000 indexes.

For command data, this heatmap becomes particularly important as the data comes by default in this 2D shape, and some files do not have the necessary fields to transform it into the 3D shape previously mentioned. This means that, for these files, the 2D visualization is the only reliable way to analyze the commands. Furthermore, the measurement data shares the same concerns as the command data, but doubled, since this data contains two of these 2D arrays, one for measurement X and one for measurement Y, and sometimes the information needed to transform it is not present.

Considering that the prospects for future files are an increase in size, rendering the entire data as a single flat image or drawing all the elements at once was not feasible for performance and memory reasons. Even with Canvas, attempting to render millions of pixels simultaneously would cause the browser to freeze or crash.

To address this, the frontend adopted a tile-based rendering strategy, inspired by techniques commonly used in mapping and large-scale image viewers such as Leaflet [2]. Just as the backend loads only a subset of frames at a time, the frontend renders only the currently visible areas of

[1] https://pixijs.com
[2] https://leafletjs.com

the dataset. For example, if the user zooms or moves into a section of the whole dataset, only the visible tiles are processed and drawn.

As the user pans and zooms across the dataset, new tiles are loaded from the backend and rendered, while previously visible tiles are discarded to conserve memory. To ensure a lower chance of lag and flicker effects, the data for each visible tile is first processed into offscreen buffers, which are then drawn onto the visible Canvas. Besides, this technique also helps avoid repeated, unnecessary, full redraws of the canvas pixel by pixel.

Despite being a solid solution that significantly improved the performance of the frontend, this solution added some complexity to the program due to the tile-mapping and loading logic. Some edge cases, such as rapid zooming and panning, would need further testing and improvements to ensure this solution is as solid as it can be.

5.3 Interactivity

Unlike static data visualization dashboards, this dashboard had to allow users to actively explore and manipulate multiple data views simultaneously, with responsive feedback and coordinated state changes across different components.

Interactivity presented trade-offs in terms of performance. Every new user interaction could trigger backend data fetches, frontend re-renders, or both. As a result, several performance optimizations were implemented to reduce unnecessary re-renders and API calls. Examples include debouncing user requests for a while to ensure only one request is sent instead of flooding the backend, caching recently loaded data, and minimizing redraws.

As mentioned before, using Canvas added extra complexity to interactivity. Unlike SVG, which provides native event support on individual graphical elements, Canvas requires all interaction logic to be handled manually at the coordinate level. Even though implementing manually all interactive behaviors added considerable development overhead, doing so was essential to meet the dashboard's core goals. The resulting system delivers a fluid exploratory experience, enabling users to gain insights from complex telemetry data in ways that would not have been possible with a more static or less integrated design.

Future iterations could expand interactivity features by adding more advanced interaction patterns (such as lasso selections, linked brushing, etc.) and optimizing rendering pipelines.

5.4 Final Results

The implementation work described throughout this chapter resulted in a functional, web-based dashboard designed for interactive exploration of AO telemetry data. This section provides an overview of the final system state, focusing on the key user-facing pages, core features, and the overall navigation workflow that ties them together.

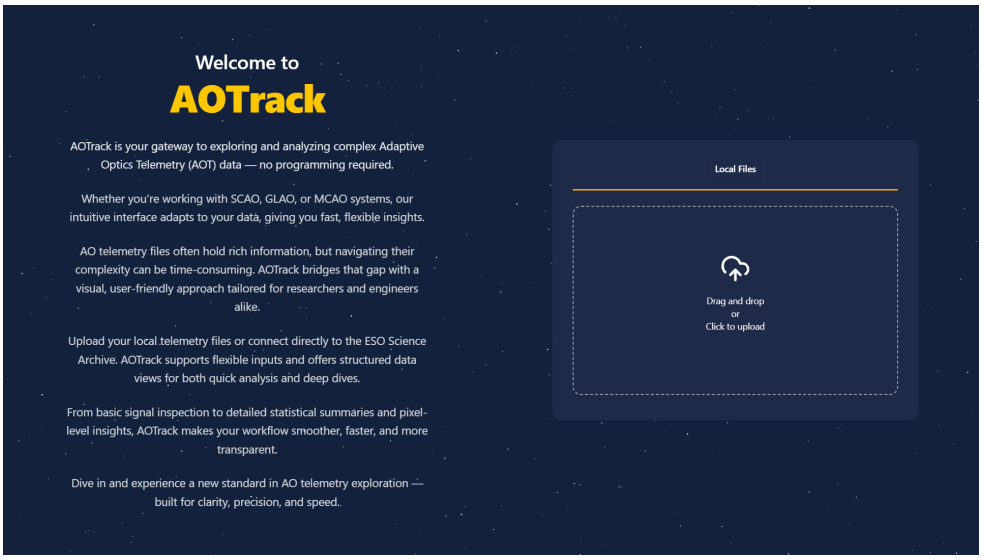


Figure 5.1: Finalized Landing Page

5.4.1 System Pages and Navigation

The dashboard is structured around four main user-facing pages: the Landing Page, Pixel Page, Measurements Page, and Commands Page. Together, they provide a comprehensive set of tools for exploring and analyzing different aspects of the uploaded AO data.

5.4.1.1 Overview Page

The Overview provides a broad, high-level summary of the AO telemetry data, helping users quickly spot general trends and patterns across the session. While not as ambitious as the original mockup, it serves as a starting point for system-wide exploration.

5.4.1.2 Landing Page

The Landing Page, as seen in Figure 5.1 serves as the dashboard entry point, and for that reason, when the user arrives, they have a short description of the dashboard and its goals, as well as the option to upload files for analysis. The page provides real-time feedback on upload progress and handles error reporting in cases where an unsupported or corrupted file is submitted. Once a file is successfully uploaded and processed by the backend, users are presented with a small overview of the file and can choose to either change the file or be directed to the main data exploration interface.

5.4.1.3 Data Domain Specific Pages

The Pixel page, seen in Figure 5.2 focuses on visualizing pixel intensity values across frames. The main focal point of the page is the big left chart, where users can travel through frames using

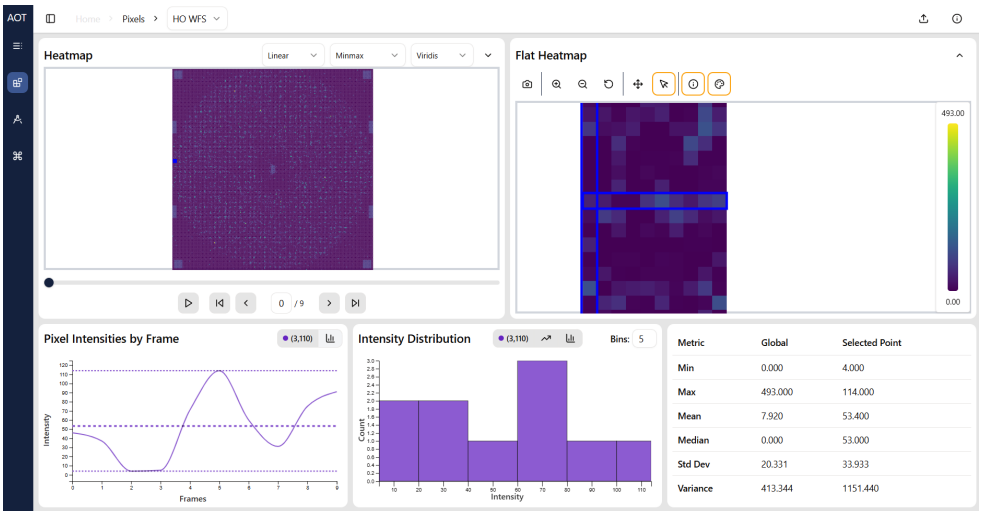


Figure 5.2: Finalized Pixel Page

both the slider controls, the input fields, or the arrows provided in the UI, and the visualizations update accordingly. The page implements both the moving frame buffer and tile-based rendering strategies described earlier, ensuring that large pixel datasets can be explored smoothly without overloading browser memory. Besides these heatmaps, the page displays various charts and statistics that are relevant when analyzing the data. All charts have various ways to interact with them, including zooming, panning, hovering for data inspection, and clicking. Most of these interactions can be performed in multiple ways, such as panning with the mouse or arrow keys, and zooming with the mouse wheel or dedicated buttons in the control bar.

In Figure 5.3, the Measurements page presents the telemetry data related to the measurements X and Y of the wavefront sensors. The charts provided on this page are much like the Pixel page charts, except for one core difference: there is an extra data dimension. This presented some issues in terms of the data shown, as the page had to show twice as much in the same amount of space. While not ideal, this option is still better than increasing page numbers, mainly because they are highly connected.

Lastly, as seen in Figure 5.4, the Commands page displays data related to the actuators of the wavefront corrector. This page shares a similar layout with the other two, which was planned as a way to make the dashboard as cohesive as possible. However, some files do not contain the necessary fields to transform command data into a 3D frame structure, making this 2D visualization the primary way to analyze commands for those cases. The chart that cannot be displayed in these situations is entirely replaced with the other heatmap.

5.4.2 Core Features

Beyond individual pages, the dashboard offers several system-wide features and interaction mechanisms that enhance usability and performance. A more formal look into what was and was not implemented will be discussed in Section 6 through a compliance matrix.

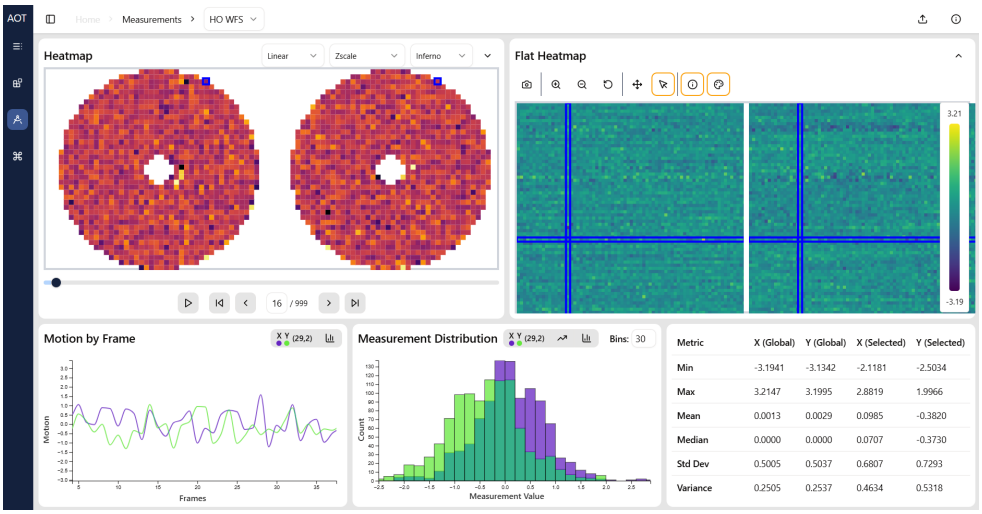


Figure 5.3: Finalized Measurements Page

Some of the core features and strategies implemented across pages were:

- **Session-Based File Handling:** Uploaded files are stored temporarily on the backend and linked to a user's session. This prevents redundant parsing, limits server resource usage, and ensures that inactive sessions are automatically cleared after some time of inactivity.
- **Cross-Chart Synchronization:** Changes made in one chart, such as updating the current frame or clicking a point, propagate to other charts on the page, ensuring all the important charts and statistics are being displayed.
- **Custom Interactivity Implementation:** Due to the choice of Canvas as the rendering technology, all interactive behaviors were manually implemented.
- **Performance Optimizations:** Several strategies were applied to maintain efficient rendering and interaction while handling large datasets:
  - Moving frame buffer that ensures that only a subset of frames is loaded into memory at any time to avoid extensive memory usage in the front end.
  - Tile-based rendering, which allows for the efficient rendering of large 2D heatmaps by processing only the visible tiles
  - Backend data slicing, which minimizes network transfer sizes by sending only the required data for each frontend request.

These core features collectively make the dashboard responsive, scalable, and suitable for exploring AO telemetry data.

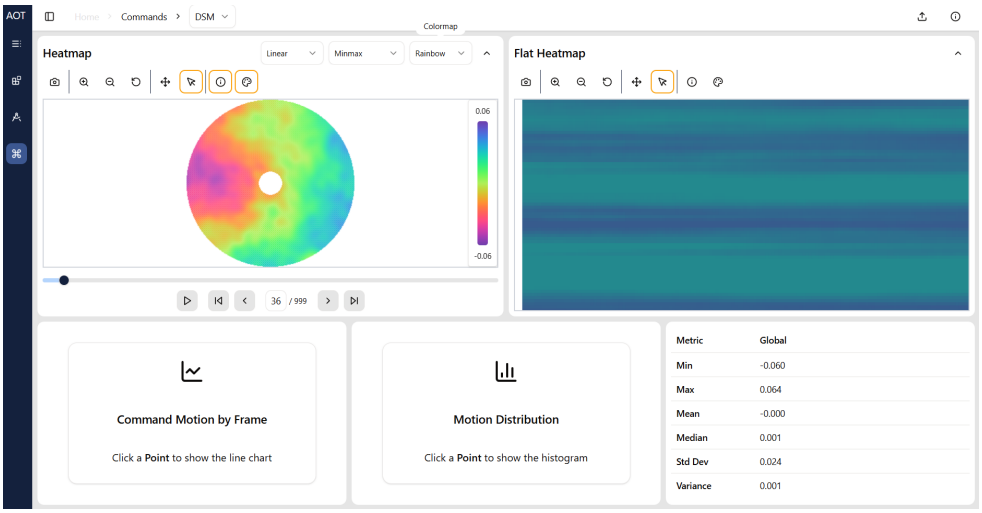


Figure 5.4: Finalized Commands Page

5.5 Summary

This chapter outlined the implementation of the AOTrack Dashboard, focusing on the key challenges encountered and the technical decisions made to address them. Rather than following a rigid development model, the implementation phase evolved iteratively, adapting to performance constraints, user feedback, and technical experimentation.

A central concern throughout was the handling and rendering of large telemetry datasets. The system avoided complex database schemas by adopting a session-based, file-on-disk approach that minimized overhead for short-term exploration. Frontend memory and performance constraints led to the development of a moving frame buffer and tile-based rendering strategy, both of which ensured efficient, responsive visualizations. While WebSockets and WebGL-based solutions like PixiJS were explored, these were deferred due to complexity and time limitations.

Canvas was selected as the rendering technology due to its performance benefits over SVG, despite the added burden of manually implementing interactivity. These efforts were necessary to support the dashboard's requirement for cross-chart synchronization and exploratory interaction.

Finally, the chapter presented the resulting system and its core components, including the Landing, Overview, Pixel, Measurement, and Command pages. These pages collectively support detailed analysis of adaptive optics telemetry data, backed by optimized data flow, robust interactivity, and user-driven design decisions. The next chapter will evaluate the effectiveness of the system against its original requirements and stakeholder expectations.

Chapter 6

Evaluation

After implementing the AOTrack Dashboard, it became necessary to assess how well the system meets its technical objectives and user-centered goals. This chapter provides a preliminary evaluation of the dashboard's functionality, performance, and usability. The purpose is not only to verify whether the system behaves as expected but also to reflect on its current limitations and propose pathways for future testing and improvement.

The evaluation is structured along two key dimensions: performance and user experience. The performance evaluation examines the system's behavior under data-intensive conditions, including rendering and responsiveness. Meanwhile, the user evaluation investigates how domain experts interact with the tool, focusing on usability and the intuitiveness of the visual design.

Given the constraints on time, hardware, and participant availability, the evaluations presented here are exploratory rather than exhaustive. Nonetheless, this chapter offers valuable insights and a structured roadmap for future validation efforts. It concludes with a requirements compliance matrix and a discussion of how well the system aligns with the functional and non-functional goals outlined earlier in the project.

6.1 Objectives

The primary goal of this evaluation chapter is to assess the AOTrack Dashboard from both a technical and user-centered perspective. Two evaluation dimensions were originally planned:

- **Performance Evaluation**, focused on system responsiveness, rendering efficiency, and data handling under large load scenarios.
- **User Evaluation**, focused on usability, interface intuitiveness, and overall user satisfaction during interactive data exploration tasks.

While constraints on available time and resources limited the scope and depth of the evaluations, both dimensions are discussed in this chapter. Furthermore, possible future evaluation methodologies are outlined to guide any subsequent testing and system validation efforts.

6.2 Methods

6.2.1 Testing Environment

All available tests were conducted using a laptop with the following specifications:

- CPU: Intel(R) Core(TM) i5-10300H
- RAM: 8 GB
- GPU: Intel(R) UHD Graphics
- Operating System: Windows 11
- Browser: Google Chrome (Version 134.x)

The backend server ran locally using FastAPI and was served via Uvicorn in development mode. Frontend testing was performed against a locally hosted build of the React application.

6.2.2 User Testing Approach

Due to the limited availability of qualified participants with relevant domain knowledge, a formal usability study with a statistically significant sample size was not feasible. Instead, an informal think-aloud strategy was adopted. Similar to how the requirements elicitation was performed, users were asked to verbalize their thoughts, impressions, and difficulties, but this time as they explored the dashboard. Feedback was collected through a recorder by the evaluator.

6.3 Performance Evaluation

Although a full-scale performance evaluation was not possible within the project timeline, performance tests were planned and now serve as a recommendation for future iterations. To evaluate rendering performance and system scalability, the following test scenarios were created:

- **Rendering Load Tests:** Measure frame rate and memory usage while rendering heatmaps of increasing size (e.g., 1,000x100, 10,000x500, 50,000x1,000 datasets).
- **Interaction Responsiveness:** Record time delays between user actions (changing frames, zooming, and panning) and corresponding frontend updates.
- **Backend Response Time:** Benchmark backend API response times for typical requests under varying file sizes and number of concurrent sessions.
- **Network Throughput:** Monitor average payload sizes and total data transferred per user session, especially for larger datasets.

Such tests would help quantify the system's performance envelope and identify future optimization opportunities.

6.4 User Evaluation

As previously mentioned, no formal user evaluation was conducted, but in this section, both the informal sessions and the plans for future evaluation will be discussed.

6.4.1 Apparatus and Tasks

The apparatus for the informal testing consisted of the same setup described in Section 6.2.1, with one observer, the evaluator, recording feedback and user comments during the session. Participants were given an open-ended task: *"Please explore the dataset using the AOTrack Dashboard as you would during a typical session. Try using all available pages and interact with the charts to examine the data."*.

6.4.2 Participant Profile

The two participants involved in the user evaluation were the same individuals who participated in the earlier requirements elicitation phase (see Chapter 3). Although this number is insufficient for formal statistical usability analysis, their background made them well-suited to provide relevant feedback, even within the informal context. The qualitative feedback gathered provided valuable insights into the system's strengths and weaknesses. The think-aloud sessions, as well as the analysis and reflection on them, yielded the feedback and respective improvements present in Table 6.1.

Table 6.1: Identified Issues and Solutions

Identified Issue	**Proposed Solution**	**Status**
Lack of right-click panning on charts (user expected it).	Add right-click support for panning charts.	Future Work
Users were unable to discover existing chart navigation controls.	Add an on-screen help or tutorial section describing navigation controls.	Future Work
Download only captures the visible chart area, but the user also expects full chart export.	Explore full-chart export options, considering large dataset sizes and resolution.	Future Work
Some charts lacked units and axis labels.	Add unit and axis labels to all relevant charts.	Implemented
The default data shown in line and histogram plots was not meaningful and could confuse users.	Show blank chart with instructional watermark prompting users to select a point.	Implemented

Identified Issue	Proposed Solution	Status
Miscalculation when converting between row/column and index in certain graphs.	Fix conversion logic to ensure correct mapping between index and (row, column).	Implemented
Some data was vertically inverted since the image starts at the bottom-left and not the top-left, as assumed.	Apply vertical flip during visualization to match expected orientation.	Implemented
The percentile dropdown limits users to predefined values (no custom percentile selection).	Add input option for custom percentile selection.	Future Work
Users could click on points in areas outside valid measurement and command regions of the reconstructed charts.	Improve restrictions on interaction and visualization for invalid regions.	Implemented
Legend values showed excessive decimal places.	Apply value clipping or rounding for cleaner legend display.	Implemented
Legends are static; users suggested clicking labels to toggle line visibility in charts.	Implement interactive legends allowing users to show/hide individual data series.	Future Work
Some unfinished features (e.g., ESO Science Archive query button) remained visible.	Remove or complete unfinished features to avoid confusion.	Future Work
The selected point highlight color did not match the color used in the corresponding charts.	Synchronize selection highlight color with the corresponding chart line color.	Implemented
Clicking a point in the 3D array chart did not update the corresponding frame in other charts (lack of bidirectional sync).	Implement bidirectional frame synchronization between charts.	Implemented
Charts only show frame numbers; users suggested showing time (e.g., milliseconds) instead.	Add option to toggle between frame number and time-based x-axis labels.	Future Work

Identified Issue	Proposed Solution	Status
Observed lag and slow back-end response during testing (due to hardware limitations of the test machine).	Conduct future testing on higher-performance hardware or deploy on an external server.	Future Work

These insights will serve as valuable guidance for future refinements and enhancements.

6.4.3 Planned Formal User Evaluation

The apparatus would, of course, depend on the evaluator, but it should resemble the average computer capabilities of a standard laptop, such as the one described in Section 6.2.1.

For future versions of the system, a more structured evaluation is recommended, following established usability testing protocols. The following methodology is proposed:

- **Apparatus:** Hosted deployment of the AOTrack Dashboard, screen recording software, and a standard machine such as the one described in Section 6.2.1.
- **Participants:** A sample size of at least 8-12 domain experts to ensure a reasonable level of statistical power.
- **Tasks:** A mix of predefined goal-oriented tasks (e.g., "Find the frame with the highest pixel intensity for pixel X", "Navigate to the Commands Page and inspect frame 500") and open-ended exploratory tasks. Some task examples include:
  - "Upload the file ao.fits and identify the Strehl ratio of the recording."
  - "Find the frame with the highest pixel intensity for pixel (0,0)."
  - "Navigate to the Commands Page and inspect frame 500."
  - "Identify possible anomalies in the measurements using the provided heatmaps."
- **Metrics:**
  - Task completion time.
  - Error rates and observed usability issues.
  - Participant-reported satisfaction using standardized instruments like the System Usability Scale (SUS) [2].
  - Qualitative feedback from post-task interviews.

A structured approach like this would provide quantifiable insights into usability and user satisfaction. A User Testing Report should be the output of the evaluation and should contain all details of said evaluation process, statistical results obtained, and a discussion of said results.

6.5 Requirements Verification

The Requirements Compliance Matrix presented in Table 6.2 provides a structured summary of how each functional and non-functional requirement outlined in the Software Requirements Specification (SRS) has been addressed in the implementation of the AOTrack Dashboard. This table is essential for tracking project progress and identifying areas where work remains. Each requirement is categorized by its functional group and annotated with its implementation status, which indicates whether it has been accepted by demonstration, still requires verification, or remains unimplemented. This systematic approach facilitates transparency and aligns with standard software engineering practices as recommended by the ECSS-E-ST-40C [6] guidelines.

A significant portion of the functional requirements has been accepted by demonstration, particularly those related to data handling, overview display, measurements, pixel analysis, and performance-critical visualizations. This reflects the maturity and usability of the implemented features, which are central to the dashboard's core purpose of visualizing and interacting with large-scale adaptive optics telemetry data. These demonstrations serve as credible evidence of functional completeness in the absence of formal certification processes.

However, the table also highlights functional areas that are either pending validation (e.g., many command-related features) or not yet implemented, such as the sandbox customization and reporting features. Additionally, several non-functional requirements related to performance and browser compatibility are marked as needing acceptance by testing. These entries flag important work that remains and serve as a guide for future development or testing cycles. Including such distinctions in the compliance matrix ensures that limitations are acknowledged and provides a roadmap for continued improvement, aligning with the iterative and feedback-driven nature of the project.

Table 6.2: Requirements Compliance Matrix

Requirement ID	**Implementation Status**
FR-DAT-01	Accepted by Demonstration
FR-DAT-02	Accepted by Demonstration
FR-DAT-03	Accepted by Demonstration
FR-DAT-04	Accepted by Demonstration
FR-DAT-05	Not Implemented
FR-OVR-01	Accepted by Demonstration
FR-OVR-02	Accepted by Demonstration
FR-OVR-03	Accepted by Demonstration
FR-OVR-04	Accepted by Demonstration
FR-OVR-05	Accepted by Demonstration
FR-OVR-06	Accepted by Demonstration
FR-OVR-07	Accepted by Demonstration
FR-OVR-08	Accepted by Demonstration

Requirement ID	**Implementation Status**
FR-OVR-09	Accepted by Demonstration
FR-OVR-10	Accepted by Demonstration
FR-OVR-11	Accepted by Demonstration
FR-OVR-12	Accepted by Demonstration
FR-OVR-13	Accepted by Demonstration
FR-MSR-01	Accepted by Demonstration
FR-MSR-02	Accepted by Demonstration
FR-MSR-03	Accepted by Demonstration
FR-MSR-04	Accepted by Demonstration
FR-MSR-05	Accepted by Demonstration
FR-MSR-06	Accepted by Demonstration
FR-MSR-07	Accepted by Demonstration
FR-MSR-08	Accepted by Demonstration
FR-MSR-09	Accepted by Demonstration
FR-MSR-10	Accepted by Demonstration
FR-MSR-11	Accepted by Demonstration
FR-MSR-12	Accepted by Demonstration
FR-MSR-13	Accepted by Demonstration
FR-MSR-14	Accepted by Demonstration
FR-MSR-15	Accepted by Demonstration
FR-MSR-16	Accepted by Demonstration
FR-MSR-17	Accepted by Demonstration
FR-MSR-18	Accepted by Demonstration
FR-MSR-19	Accepted by Demonstration
FR-PXL-01	Accepted by Demonstration
FR-PXL-02	Accepted by Demonstration
FR-PXL-03	Accepted by Demonstration
FR-PXL-04	Accepted by Demonstration
FR-PXL-05	Accepted by Demonstration
FR-PXL-06	Accepted by Demonstration
FR-PXL-07	Accepted by Demonstration
FR-PXL-08	Accepted by Demonstration
FR-PXL-09	Accepted by Demonstration
FR-PXL-10	Accepted by Demonstration
FR-PXL-11	Accepted by Demonstration
FR-PXL-12	Accepted by Demonstration
FR-PXL-13	Accepted by Demonstration
FR-PXL-14	Accepted by Demonstration
FR-PXL-15	Accepted by Demonstration

Requirement ID	Implementation Status
FR-PXL-16	Accepted by Demonstration
FR-PXL-17	Accepted by Demonstration
FR-PXL-18	Accepted by Demonstration
FR-PXL-19	Accepted by Demonstration
FR-PXL-20	Accepted by Demonstration
FR-PXL-21	Accepted by Demonstration
FR-CMD-01	Accepted by Demonstration
FR-CMD-02	Needs Acceptance by Demonstration
FR-CMD-03	Needs Acceptance by Demonstration
FR-CMD-04	Needs Acceptance by Demonstration
FR-CMD-05	Needs Acceptance by Demonstration
FR-CMD-06	Needs Acceptance by Demonstration
FR-CMD-07	Needs Acceptance by Demonstration
FR-CMD-08	Needs Acceptance by Demonstration
FR-CMD-09	Needs Acceptance by Demonstration
FR-CMD-10	Needs Acceptance by Demonstration
FR-CMD-11	Needs Acceptance by Demonstration
FR-CMD-12	Needs Acceptance by Demonstration
FR-CMD-14	Needs Acceptance by Demonstration
FR-CMD-16	Needs Acceptance by Demonstration
FR-CMD-17	Needs Acceptance by Demonstration
FR-CMD-18	Needs Acceptance by Demonstration
FR-CMD-19	Needs Acceptance by Demonstration
FR-SBX-01	Not Implemented
FR-SBX-02	Not Implemented
FR-SBX-03	Not Implemented
FR-SBX-04	Not Implemented
FR-SBX-05	Not Implemented
FR-SBX-06	Not Implemented
FR-SBX-07	Not Implemented
FR-SEM-01	Accepted by Demonstration
FR-SEM-02	Not Implemented
NFR-PRF-01	Accepted by Demonstration
NFR-PRF-02	Needs Acceptance by Testing
NFR-PRF-03	Needs Acceptance by Testing
NFR-PRF-04	Needs Acceptance by Testing
NFR-PRF-05	Needs Acceptance by Testing
NFR-PRF-06	Needs Acceptance by Testing
NFR-CST-01	Accepted by Review

Requirement ID	Implementation Status
NFR-CST-02	Accepted by Review

6.6 Discussion of Results

Given the informal nature of the conducted user evaluation and the absence of full performance benchmarks, the results presented in this chapter should be interpreted as preliminary and indicative rather than conclusive.

Nevertheless, the informal feedback sessions confirmed that the system is usable and performant enough for exploratory analysis on typical consumer hardware, especially when working with moderately sized AOT files. Participants consistently commented on the smoothness of frame navigation and the responsiveness of the interface, validating key implementation decisions such as session-based data handling and tile-based rendering.

However, both performance and usability remain areas where more rigorous evaluation is needed. The proposed testing methodologies outlined in this chapter provide a foundation for future work to conduct more detailed and statistically significant studies, which would further validate the dashboard's scalability, efficiency, and usability across a broader range of use cases and user profiles.

6.7 Summary

This chapter presented an initial evaluation of the AOTrack Dashboard, examining both performance and usability within the limitations of available resources. The performance evaluation identified key test scenarios that can be used in future benchmarking efforts, while informal user testing with domain experts provided qualitative feedback that helped address usability issues and confirm implementation strengths.

The think-aloud protocol enabled quick iteration and refinement based on real user expectations, leading to several implemented fixes and a set of clearly defined future improvements. Participants found the dashboard generally intuitive and responsive, validating major design decisions such as session-based file handling, moving frame buffers, and tile-based rendering.

A proposed methodology for formal user evaluation was also outlined, including concrete task examples and measurable usability metrics such as SUS scores and task completion times. Additionally, the Requirements Compliance Matrix demonstrated broad alignment between the implemented system and its original goals, while also highlighting pending features and gaps to be addressed in future versions.

While not definitive, this evaluation confirms that the dashboard is functional, usable, and performant in its current state. It lays the groundwork for more rigorous future assessments that can further validate its role as a reliable tool for adaptive optics telemetry data exploration.

Chapter 7

Conclusion

What began as an initial exploration of visualization challenges in astronomical data evolved into a focused and technically demanding project centered on adaptive optics. The scope shift from a broad space-related data visualization to the particular and highly complex field of Adaptive Optics visualization shaped every stage of this dissertation, from framing the problem to implementing a solution.

By identifying the unique challenges inherent to AO telemetry visualization and reviewing existing tools in the astronomical visualization domain, it was clear that, while valuable, current solutions did not meet the specific needs of AO telemetry exploration. The recent standardization effort around the AOT file format also reinforced the opportunity and necessity for new visualization solutions.

Although limited in participant count, the requirement elicitation process involved key stakeholders from both the data generation and scientific analysis sides of the workflow, ensuring that the developed solution reflected real-world needs and constraints of the adaptive optics community. These requirements drove the subsequent system architecture and visual design choices, ranging from technology stack selection to frontend-backend interactions.

When implementing, the AO data challenges became even more noticeable. The practical steps taken to bring the dashboard to life demanded backend data handling strategies and frontend rendering optimizations necessary for managing files ranging from thousands of megabytes to several gigabytes. Various decisions in the process needed to be made, and careful consideration and risk analysis were needed before moving forward.

The evaluation, while not as complete as originally planned due to several constraints, still outlined the approaches taken for requirement verification and gave suggestions for future formal evaluation, ensuring that the dashboard can remain open for future refinement and testing.

The result of this dissertation was a functional dashboard prototype that balances the technical demands of the data with the more user-centered aspects of the problem. This dashboard still leaves much space for further optimizations and formal evaluation, as AO is an area that continues to gain interest. Future steps outlined throughout the dissertation that could significantly improve the dashboard include further performance tuning of the data handling process, exploring full

WebGL-based rendering with tools like Pixi.js, and conducting broader user testing with a larger and more diverse participant pool.

Ultimately, while this work was developed with AO in mind, the project's task of balancing performance with usability and designing for exploratory scientific analysis may prove valuable beyond the adaptive optics lens.

References

[1] Benjamin Bach et al. “Dashboard Design Patterns”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (Jan. 2023). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 342–352. ISSN: 1941-0506. DOI: 10.1109/TVCG.2022.3209448. URL: https://ieeexplore.ieee.org/document/9903550 (visited on 11/03/2024).

[2] Aaron Bangor, Philip T. Kortum, and James T. Miller. “An Empirical Evaluation of the System Usability Scale”. In: *International Journal of Human–Computer Interaction* 24.6 (July 2008). Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10447310802205776, pp. 574–594. ISSN: 1044-7318. DOI: 10.1080/10447310802205776. URL: https://doi.org/10.1080/10447310802205776 (visited on 12/15/2024).

[3] F. Bonnarel et al. “The ALADIN interactive sky atlas - A reference tool for identification of astronomical sources”. en. In: *Astronomy and Astrophysics Supplement Series* 143.1 (Apr. 2000). Number: 1 Publisher: EDP Sciences, pp. 33–40. ISSN: 0365-0138, 1286-4846. DOI: 10.1051/aas:2000331. URL: https://aas.aanda.org/articles/aas/abs/2000/07/ds1785/ds1785.html (visited on 07/27/2025).

[4] Claire Max. *Introduction to Adaptive Optics and its History*. Sept. 2001. URL: https://www.ucolick.org/~max/History_AO_Max.htm (visited on 07/31/2025).

[5] Dawn Dowding and Jacqueline A. Merrill. “The Development of Heuristics for Evaluation of Dashboard Visualizations”. en. In: *Applied Clinical Informatics* 09 (July 2018). Publisher: Georg Thieme Verlag KG, pp. 511–518. DOI: 10.1055/s-0038-1666842. URL: https://www.thieme-connect.de/products/ejournals/abstract/10.1055/s-0038-1666842 (visited on 07/27/2025).

[6] ECSS. *ECSS-E-ST-40C Rev.1 – Software*. Apr. 2025. URL: https://ecss.nl/standard/ecss-e-st-40c-rev-1-software-30-april-2025/ (visited on 06/30/2025).

[7] ESO. *E-ELT Construction Proposal*. en-gb. Dec. 2011. URL: https://www.eso.org/public/products/books/book_0046/ (visited on 07/27/2025).

[8] ESO. *The E-ELT compared to the Padrão dos Descobrimentos in Lisbon, Portugal*. en-gb. May 2016. URL: https://www.eso.org/public/images/eso1617t/ (visited on 07/27/2025).

[9] Pierre Fernique. *Aladin User Manual*. Sofware Manual. Centre de Données astronomiques de Strasbourg, June 2022. URL: https://aladin.cds.unistra.fr/java/AladinManual.pdf (visited on 12/18/2024).

[10] Stephen Few. *Information Dashboard Design*. O’Reilly, Jan. 2006. ISBN: 0-596-10016-7.

[11] Stephen Few. *What Ordinary People Need Most from Information Visualization Today*. en. Aug. 2008. URL: https://www.perceptualedge.com/articles/visual_business_intelligence/what_people_need_from_infovis.pdf.

[12] Tiago Gomes et al. “Adaptive optics telemetry standard - Design and specification of a novel data exchange format”. en. In: *Astronomy & Astrophysics* 686 (June 2024). Publisher: EDP Sciences, A7. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/202348486. URL: https://www.aanda.org/articles/aa/abs/2024/06/aa48486-23/aa48486-23.html (visited on 07/27/2025).

[13] Jeffrey Heer and George Robertson. “Animated Transitions in Statistical Data Graphics”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 1240–1247. ISSN: 1941-0506. DOI: 10.1109/TVCG.2007.70539. URL: https://ieeexplore.ieee.org/document/4376146 (visited on 12/19/2024).

[14] ISO. *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) - Product quality model*. Nov. 2023. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-2:v1:en (visited on 12/15/2024).

[15] P.B. Kruchten. “The 4+1 View Model of architecture”. In: *IEEE Software* 12.6 (Nov. 1995), pp. 42–50. ISSN: 1937-4194. DOI: 10.1109/52.469759. URL: https://ieeexplore.ieee.org/document/469759 (visited on 07/29/2025).

[16] Stephen R. Midway. “Principles of Effective Data Visualization”. In: *Patterns* 1.9 (Dec. 2020), p. 100141. ISSN: 2666-3899. DOI: 10.1016/j.patter.2020.100141. URL: https://www.sciencedirect.com/science/article/pii/S2666389920301896 (visited on 11/03/2024).

[17] A. Moitinho. *Gaia Archive Visualisation Service (GAVS)*. June 2022. URL: https://www.cosmos.esa.int/web/gaia-users/archive/visualisation (visited on 12/17/2024).

[18] A. Moitinho et al. “Gaia Data Release 1 - The archive visualisation service”. en. In: *Astronomy & Astrophysics* 605 (Sept. 2017). Publisher: EDP Sciences, A52. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/201731059. URL: https://www.aanda.org/articles/aa/abs/2017/09/aa31059-17/aa31059-17.html (visited on 11/26/2024).

[19] Mario Nadj, Alexander Maedche, and Christian Schieder. “The effect of interactive analytical dashboard features on situation awareness and task performance”. In: *Decision Support Systems* 135 (Aug. 2020), p. 113322. ISSN: 0167-9236. DOI: 10.1016/j.dss.2020.113322. URL: https://www.sciencedirect.com/science/article/pii/S0167923620300774 (visited on 12/10/2024).

[20] J. Nielsen. *10 Usability Heuristics for User Interface Design*. Jan. 2024. URL: nngroup.com/articles/ten-usability-heuristics (visited on 12/15/2024).

[21] Paolo Padovani and Michele Cirasuolo. “The Extremely Large Telescope”. In: *Contemporary Physics* 64.1 (Jan. 2023), pp. 47–64. ISSN: 0010-7514. DOI: 10.1080/00107514.2023.2266921. (Visited on 10/10/2024).

[22] Paolo Padovani et al. “The ESO’s Extremely Large Telescope Working Groups”. In: *The Messenger* 189 (Jan. 2023). ISSN: 0722-6691 Publisher: European Southern Observatory (ESO), pp. 23–30. DOI: 10.18727/0722-6691/5286. URL: https://doi.eso.org/10.18727/0722-6691/5286 (visited on 07/27/2025).

[23] Freddy Paz and Jose Pow-Sang. "A systematic mapping review of usability evaluation methods for software development process". In: *International Journal of Software Engineering and Its Applications* 10 (Jan. 2016), pp. 165–178. DOI: 10.14257/ijseia.2016.10.1.16.

[24] R Pushpakumar et al. "Human-Computer Interaction: Enhancing User Experience in Interactive Systems". In: *E3S Web of Conferences* 399 (July 2023). DOI: 10.1051/e3sconf/202339904037.

[25] Alper Sarikaya et al. "What Do We Talk About When We Talk About Dashboards?" In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (Jan. 2019). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 682–692. ISSN: 1941-0506. DOI: 10.1109/TVCG.2018.2864903. URL: https://ieeexplore.ieee.org/document/8443395 (visited on 10/10/2024).

[26] Gayane Sedrakyan, Erik Mannens, and Katrien Verbert. "Guiding the choice of learning dashboard visualizations: Linking dashboard design and data visualization concepts". In: *Journal of Computer Languages* 50 (Feb. 2019), pp. 19–38. ISSN: 2590-1184. DOI: 10.1016/j.jvlc.2018.11.002. URL: https://www.sciencedirect.com/science/article/pii/S1045926X18301009 (visited on 11/03/2024).

[27] Danielle Albers Szafir. "Modeling Color Difference for Visualization Design". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (Jan. 2018). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 392–401. ISSN: 1941-0506. DOI: 10.1109/TVCG.2017.2744359. URL: https://ieeexplore.ieee.org/document/8017604 (visited on 12/09/2024).

[28] Edward Tufte. *The Visual Display of Quantitative Information*. 2nd. GRAPHICS PRESS LLC, 2001.

[29] Andrea Vázquez-Ingelmo, Francísco J. Garcia-Peñalvo, and Roberto Therón. "Information Dashboards and Tailoring Capabilities - A Systematic Literature Review". In: *IEEE Access* 7 (2019). Conference Name: IEEE Access, pp. 109673–109688. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2933472. URL: https://ieeexplore.ieee.org/document/8789402 (visited on 11/03/2024).

[30] Milena Vuckovic, Dawid Wolosiuk, and Johanna Schmidt. "Designing Interactive Analytics Dashboards for Diverse Target Groups: The Process and Decision-Making". In: *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech)*. June 2024, pp. 1–4. DOI: 10.23919/SpliTech61897.2024.10612622. URL: https://ieeexplore.ieee.org/document/10612622 (visited on 11/03/2024).

[31] W3C World Wide Web Consortium. *Web Content Accessibility Guidelines*. Tech. rep. Recommendation 06 May 2025. URL: https://www.w3.org/TR/WCAG21/ (visited on 07/29/2025).

[32] Jagoda Walny et al. "Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff". In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (Jan. 2020). Conference Name: IEEE Transactions on Visualization and Computer Graphics, pp. 12–22. ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2934538. URL: https://ieeexplore.ieee.org/document/8816695 (visited on 10/10/2024).

[33] Steve Wexler, Jeffrey Shaffer, and Andy Cotgreave. *The Big Book of Dashboards: Visualizing Your Data Using Real-World Business Scenarios*. 2017. ISBN: 978-1-119-28308-9. URL: 10.1002/9781119283089.

[34] Ji Soo Yi et al. “Toward a Deeper Understanding of the Role of Interaction in Information Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (Nov. 2007), pp. 1224–1231. ISSN: 1941-0506. DOI: 10.1109/TVCG.2007.70515. URL: https://ieeexplore.ieee.org/document/4376144 (visited on 07/27/2025).

[35] Nuowen Zhang et al. “The Effects of Layout Order on Interface Complexity: An Eye-Tracking Study for Dashboard Design”. In: *Sensors* 24 (Sept. 2024), p. 5966. DOI: 10.3390/s24185966. URL: https://ui.adsabs.harvard.edu/abs/2024Senso..24.5966Z (visited on 11/04/2024).

[36] Jonathan Zong et al. “Animated Vega-Lite: Unifying Animation with a Grammar of Interactive Graphics”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (Jan. 2023), pp. 149–159. ISSN: 1941-0506. DOI: 10.1109/TVCG.2022.3209369. URL: https://ieeexplore.ieee.org/document/9914804 (visited on 07/27/2025).

Appendix A

SRS document

The following pages contain the full Software Requirements Specification (SRS) document developed for this dissertation.

Software Requirements Specification
for the
AOTrack Dashboard

Prepared by:

Ana Rita Baptista de Oliveira

Faculdade de Engenharia da Universidade do Porto
Mestrado em Engenharia Informática e Computação

Contents

Chapter 1

Introduction

The purpose of this document is to define the software requirements for the **AOTrack Dashboard**. The dashboard is designed to support astronomers, engineers, and researchers by providing an intuitive and efficient interface for monitoring, analyzing, and visualizing adaptive optics data, specifically with the AOT format. The improved dashboard will address current usability issues, enhance performance, and introduce new features based on user feedback.

Chapter 2

Applicable and reference documents

ID	Document
R-01	ECSS-E-ST-40C-Rev.1 [1]
R-02	Adaptive optics telemetry standard - Design and specification of a novel data exchange format [2]

Chapter 3

Terms, definitions and abbreviated terms

AO Adaptive Optics

AOT Adaptive Optics Telemetry

ELT Extremely Large Telescope

ESO European Southern Observatory

FITS Flexible Image Transport System

UI User Interface

UX User Experience

WFC Wavefront Corrector

WFS Wavefront Sensor

Chapter 4

Software context and overview

The AOTrack Dashboard is a planned web-based software system intended to support the visualization and analysis of telemetry data generated by Adaptive Optics (AO) systems. Its primary goal is to provide the stakeholders identified in table 4.1 with tools to explore, inspect, and interpret large-scale telemetry datasets produced during telescope operations.

AO telemetry data is typically characterized by its large volume and complexity, with individual sessions often producing files containing millions of rows and reaching sizes of several gigabytes. The dashboard will address this challenge by implementing efficient data handling mechanisms on the backend and interactive, performance-optimized visualizations on the frontend.

The system will support essential user tasks such as loading telemetry sessions, visualizing parameter time series, filtering data, and comparing different telemetry variables. The user interface will be designed to accommodate both technical users familiar with AO concepts and less technical stakeholders who may require simpler exploration tools.

This document defines the functional and non-functional requirements for the development of the Adaptive Optics Telemetry Dashboard, outlining its intended scope, operating environment, user characteristics, and key system constraints.

Table 4.1: Identified system stakeholders.

Stakeholder	**Description**	**Interest / Role in the System**
AO System Developers	Engineers responsible for the creation and management of the Adaptive Optics Telemetry data.	Ensure that the dashboard can correctly interpret and visualize data exported from AO systems, following the AOT format specifications.
Dashboard Developers	Software developers in charge of building and maintaining the dashboard application.	Responsible for implementing system functionality, ensuring performance, and maintaining usability and scalability.
AO Researchers	Scientists and engineers who analyze AO telemetry data for performance evaluation and research purposes.	Need interactive tools for exploring, filtering, and visualizing large-scale AO telemetry datasets.

Chapter 5

Functional Requirements

This section outlines the functional requirements for the ELT Dashboard, structured by functional areas and grouped logically for clarity.

5.1 Data Handling

ID	Requirement
FR-DAT-01	The system shall support the upload of AO FITS files (.fits).
FR-DAT-02	The system shall reject files that don't conform to the AOT format.
FR-DAT-03	The system shall display a descriptive error message when files are rejected.
FR-DAT-04	The system shall display a loading indicator while the file is being uploaded.
FR-DAT-05	The system may support direct upload of AOT files using the ESO Science Archive API.

5.2 Overview details

ID	Requirement
FR-OVR-01	The system shall have an overview section displaying informative data on the dataset.
FR-OVR-02	The system shall display the main telescope name.
FR-OVR-03	The system shall display the system name.
FR-OVR-04	The system shall display the telescope mode.
FR-OVR-05	The system shall display the recording start date.
FR-OVR-06	The system shall display the recording end date.
FR-OVR-07	The system shall display the Strehl ratio.
FR-OVR-08	The system shall display the telescope configuration.
FR-OVR-09	The system shall display the atmospheric parameters at time of recording.
FR-OVR-10	The system shall display the light sources.
FR-OVR-11	The system shall display the wavefront sensors.
FR-OVR-12	The system shall display the wavefront correctors.
FR-OVR-13	The system shall display the loops.

5.3 Measurement details

ID	Requirement
FR-MSR-01	The system shall have a measurements section displaying data on the measurements.
FR-MSR-02	The system shall allow the selection of measurement types.
FR-MSR-03	The system shall display the name.
FR-MSR-04	The system shall display the type.
FR-MSR-05	The system shall display the wavelength.
FR-MSR-06	The system shall display the centroiding algorithm.
FR-MSR-07	The system shall display statistical values of the measurements (min, max, mean, standard deviation).
FR-MSR-08	The system shall display statistical values of the X measurements (min, max, mean, standard deviation).
FR-MSR-09	The system shall display statistical values of the Y measurements (min, max, mean, standard deviation).
FR-MSR-10	The system shall display a heatmap of the X measurement values for a selected frame.
FR-MSR-11	The system shall display a heatmap of the Y measurement values for a selected frame.
FR-MSR-12	The system shall a line chart showing the average intensity of the X and Y measurements across frames
FR-MSR-13	The system shall display a heatmap of the X measurement values per index by frame.
FR-MSR-14	The system shall display a heatmap of the Y measurement values per index by frame.
FR-MSR-15	The system shall allow the user to choose the scale applied to the data.
FR-MSR-16	The system shall allow the user to choose the color map applied to the data.
FR-MSR-17	The system shall allow the user to choose the interval applied to the data.
FR-MSR-18	The system shall allow the user to choose the selected frame.
FR-MSR-19	When the user clicks a point on any chart, the system shall display statistical values for that data point.

5.4 Pixel details

ID	Requirement
FR-PXL-01	The system shall have a pixel section displaying data on the pixels.
FR-PXL-02	The system shall allow the selection of WFS.
FR-PXL-03	The system shall display the selected WFS name.
FR-PXL-04	The system shall display the shutter type.
FR-PXL-05	The system shall display the sub-aperture size.
FR-PXL-06	The system shall display the mask offset.
FR-PXL-07	The system shall display the frame rate.
FR-PXL-08	The system shall display the gain.
FR-PXL-09	The system shall display the integration time.
FR-PXL-10	The system shall display the pixel scale.
FR-PXL-11	The system shall display the quantum efficiency.
FR-PXL-12	The system shall display the readout noise.
FR-PXL-13	The system shall display the readout rate.
FR-PXL-14	The system shall display the sampling technique.
FR-PXL-15	The system shall display statistical values of the pixels (min, max, mean, standard deviation).
FR-PXL-16	The system shall display a heatmap of the pixel intensities for a selected frame.
FR-PXL-17	The system shall a line chart showing the average intensity of the pixels across frames
FR-PXL-18	The system shall display a heatmap of the pixel intensities per index by frame.
FR-PXL-19	The system shall allow the user to choose the scale applied to the data.
FR-PXL-20	The system shall allow the user to choose the color map applied to the data.
FR-PXL-21	The system shall allow the user to choose the interval applied to the data.
FR-PXL-21	The system shall allow the user to choose the selected frame.
FR-PXL-22	When the user clicks a point on any chart, the system shall display statistical values for that data point.

5.5 Command details

ID	Requirement
FR-CMD-01	The system shall have a command section displaying data on the commands.
FR-CMD-02	The system shall allow the selection of WFC.
FR-CMD-03	The system shall display the selected WFC name.
FR-CMD-04	The system shall display the loop type.
FR-CMD-05	The system shall display the reference commands.
FR-CMD-06	The system shall display the residual commands.
FR-CMD-07	The system shall display the modal coefficients.
FR-CMD-08	The system shall display the valid actuators.
FR-CMD-09	The system shall display the `tfz_num` value.
FR-CMD-10	The system shall display the `tfz_den` value.
FR-CMD-11	The system shall display a heatmap of the actuator motion for a selected frame.
FR-CMD-12	The system shall display a line chart showing the average actuator motion across frames.
FR-CMD-14	The system shall allow the user to choose the scale applied to the actuator diagram.
FR-CMD-16	The system shall allow the user to choose the colormap applied to the actuator diagram.
FR-CMD-17	The system shall allow the user to choose the interval applied to the actuator diagram.
FR-CMD-18	The system shall allow the user to choose the selected frame.
FR-CMD-19	When the user clicks a point on any chart, the system shall display statistical values for that data point.

5.6 Sandbox

ID	Requirement
FR-SBX-01	The system may have a sandbox section where the user can customize the displayed visualizations.
FR-SBX-02	The system may allow users to import visualizations from the pixel, command, and measurements sections.
FR-SBX-03	The system may allow users to arrange visualizations freely within the visualization area.
FR-SBX-04	The system may allow users to resize visualizations freely within the visualization area.
FR-SBX-05	The system may allow users to reposition visualizations freely within the visualization area.
FR-SBX-06	The system may allow users to save custom visualization layouts.
FR-SBX-07	The system may allow users to load custom visualization layouts.

5.7 Session Management

ID	Requirement
FR-SEM-01	The system shall not retain data between sessions.
FR-SEM-02	The system may have a "Print Report" button that generates and downloads a PDF of the current state of the dashboard.

Chapter 6

Non-Functional Requirements

6.1 Performance

ID	Requirement
NFR-PRF-01	The system shall complete parsing of any file up to 4 GB in <30 seconds over a sustained 100 Mbps connection.
NFR-PRF-02	The system shall be compatible with all major modern browsers (Chrome, Firefox, Opera, and Safari).
NFR-PRF-03	The redraw latency for all charts shall not exceed 200 ms (p95) when visualizing the reference dataset.
NFR-PRF-04	The system shall support at least 100 concurrent sessions with CPU utilization <80% on target hardware.

6.2 Reliability

ID	Requirement
NFR-PRF-01	The system shall have a mean time between failure of at least 100 hours under normal usage conditions.
NFR-PRF-02	The system shall have a mean time to recovery of <2 minutes after an unexpected crash or service interruption.

6.3 Design and Implementation Constraints

ID	Requirement
NFR-CST-01	The system shall rely exclusively on open-source tools and libraries.
NFR-CST-02	The system shall use the `aotpy` library for reading and processing AOT data.

Chapter 7

Requirement Compliance

Requirement compliance refers to the process of ensuring that the implemented system meets the functional and non-functional requirements defined in this document. Given the nature of this project and its academic scope, different verification and validation strategies may be applied to confirm compliance with each requirement.

The approaches for requirement verification are:

- **Testing**: Executing the system under controlled conditions to verify that specific functional requirements behave as expected.
- **Demonstration**: Running the system and showing that certain features or behaviors occur as described, without requiring formal test cases.
- **Inspection or Review**: Manually checking code, documentation, or system outputs to ensure compliance with non-functional requirements or standards.
- **User Feedback and Acceptance**: Engaging representative users to interact with the system and provide feedback on whether their needs and expectations have been met.

In the context of this project, some requirements are best validated through demonstration and user acceptance testing. Others, such as functional feature implementation, can be confirmed through direct testing or code inspection.

A formal requirement compliance matrix detailing the status of each requirement is maintained separately from this document as part of the dissertation.

Bibliography

[1] European Cooperation for Space Standardization. *ECSS-E-ST-40C Rev.1 – Software*. Version Rev.1. Apr. 30, 2025. URL: https://ecss.nl/standard/ecss-e-st-40c-rev-1-software-30-april-2025/ (visited on 06/30/2025).

[2] Tiago Gomes et al. "Adaptive optics telemetry standard - Design and specification of a novel data exchange format". en. In: *A&A* 686 (June 2024). Publisher: EDP Sciences, A7. ISSN: 0004-6361, 1432-0746. DOI: 10.1051/0004-6361/202348486. URL: https://www.aanda.org/articles/aa/abs/2024/06/aa48486-23/aa48486-23.html (visited on 06/30/2025).