FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Visualization and Annotation of Space Surveillance and Tracking Images

Rodrigo José de Castro Pinheiro

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Paulo Garcia

Second Supervisor: Prof. Fernando Cassola

July 14, 2026

© Rodrigo José de Castro Pinheiro, 2026

Visualization and Annotation of Space Surveillance and Tracking Images

Rodrigo José de Castro Pinheiro

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Ricardo Cruz
Examiner: Prof. Paulo Gordo
Member: Prof. Paulo Garcia

July 14, 2026

Resumo

A monitorização de objetos em órbita terrestre, em particular dos detritos espaciais, tornou-se um desafio urgente. Por isso, a análise eficaz das imagens obtidas por telescópios, no contexto da vigilância e rastreamento espacial, reveste-se de grande importância para melhorar o conhecimento situacional espacial ([2]). O principal problema decorre da falta de uma solução prática que permita o processamento, a identificação e a visualização integrados de imagens provenientes de sensores e telescópios, tais como as captadas pelo Observatório Óptico do Pico do Areeiro.

Para enfrentar este desafio, foi desenvolvido um painel interativo capaz de processar imagens de telescópio. Ferramentas externas, como aquelas disponibilizados pela Agência Espacial Europeia, foram utilizadas para realização de um sistema robusto e interoperável. Deste modo, um algoritmo externo, o ASTRiDE, foi integrado no painel e incorporou-se também o catálogo de correspondência cruzada Gaia DR3. O primeiro permite uma para deteção automática de rastos no campo de visão de cada imagem carregada, e o segundo permite a correspondência cruzada de estrelas, apoiando a validação astrométrica e a discriminação de objetos. Os resultados são apresentados ao utilizador num ambiente dinâmico e interativo. A abordagem incorpora princípios de engenharia de painéis, cadeias de processamento assíncronas e técnicas de visualização orientadas para a interação.

Os resultados mostram que o protótipo, que constitui o produto mínimo viável desta dissertação, melhora a eficiência do fluxo de trabalho de vigilância e rastreio espacial. Ao reduzir a carga de trabalho manual, apoiar a tomada de decisões e melhorar a eficiência da análise de dados de imagens provenientes de telescópio, o painel desenvolvido é uma ferramenta útil e potencialmente escalável para que os operadores especializados na área possam desempenhar melhor as suas tarefas. A avaliação heurística, os testes de desempenho e os estudos de usabilidade orientados para o utilizador confirmam a usabilidade e a utilidade do sistema. Conclui-se que a solução proposta constitui uma contribuição significativa para a modernização das ferramentas de monitorização de objetos em órbita, abrindo caminho para futuras extensões e integrações operacionais.

Abstract

Monitoring objects in Earth orbit, particularly space debris, has become an urgent challenge. Therefore, the effective analysis of images obtained by telescopes, in the context of space surveillance and tracking, is of great importance for improving space situational awareness ([2]). The main problem stems from the lack of a practical solution that allows for the integrated processing, identification, and visualization of images from sensors and telescopes, such as those captured by the Pico do Areeiro Optical Observatory.

To face this challenge, an interactive dashboard capable of processing telescope images was developed. External tools, such as those provided by the European Space Agency, were used to create a robust and interoperable system. Consequently, an external algorithm, ASTRiDE, was integrated into the dashboard, and the Gaia DR3 cross-catalog was also incorporated. The first enables the automatic detection of streaks within the field of view of each uploaded image, while the second allows for the cross-referencing of stars, supporting astrometric validation and object discrimination. The results are presented to the user in a dynamic and interactive environment. The approach incorporates dashboard engineering principles, asynchronous processing pipelines, and interaction-oriented visualization techniques.

The results show that the prototype, which constitutes the minimum viable product of this dissertation, improves the efficiency of the space surveillance and tracking workflow. By reducing manual workload, supporting decision-making, and improving the efficiency of analyzing image data from telescopes, the developed dashboard is a useful and potentially scalable tool that enables specialized operators in the field to perform their tasks more effectively. Heuristic evaluation, performance tests, and user-centered usability studies confirm the system's usability and usefulness. It is concluded that the proposed solution represents a significant contribution to the modernization of tools for monitoring objects in orbit, paving the way for future extensions and operational integrations.

UN Sustainable Development Goals

The United Nations Sustainable Development Goals (SDGs) constitute an international framework focused on building a more equitable, inclusive, and environmentally responsible future. This set consists of 17 goals[1] that aim to address major global challenges, ranging from poverty eradication and reducing inequalities to climate action, environmental protection, the promotion of peace and social justice.

The focus of this dissertation is the development of an interactive and dynamic dashboard that allows the analysis of space debris in Earth orbit in telescope images uploaded by the users, as well as the visualization of common data regarding stars visible in the field of view of each uploaded image. The work developed aims to contribute to several United Nations Sustainable Development Goals (SDGs), specifically those related to scientific infrastructure, technological innovation, responsible management of orbital environments, and international cooperation in Space Surveillance and Tracking. The SDGs defined in Table 1 identify targets and indicators that were selected following the implementation of the prototype for this dissertation, as well as their contribution to the area and relevance to the sustainable and safe development of the orbital environment.

The specific Sustainable Development Goals mentioned have the following names and descriptions:

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 information, see: https://sdgs.un.org/goals

SDG	Target	Contribution	Performance Indicators and Metrics
4	4.4	Promotes the development of scientific and digital skills, offering users an interactive, visual tool that enables them to analyze telescope images, understand catalog cross-matching, and interact with SST data. In addition to experts, the prototype can be used by anyone who wants to learn more and explore.	Number of students or researchers using the dashboard in academic or non-academic contexts. Feedback on learning success. User testing workshops.
9	9.1	Improves the performance of SST specialists by offering an interactive, digital tool for analyzing telescope images and carrying out scientific monitoring activities.	Usage of the dashboard in SST operations. Frequency of use by operators. Qualitative feedback on reliability and usability.
9	9.5	Integrates external algorithms such as ASTRiDE and the GAIA DR3 catalog, resulting in an engineering dashboard that is linked to image and data analysis, providing support for decision-making and scientific research.	Usage of the dashboard in scientific research projects. Feedback collection from operators and researchers. Integration into pipelines and related contexts.
17	17.6	Promotes international scientific collaboration through shared catalogs such as Gaia DR3, encouraging interoperability and collective analysis.	Number of teams working together using the same dashboard. Feedback received from outside partners. Participation in shared SST activities.
17	17.16	Supports multiple partnerships, providing a tool that enables image analysis and the export of results for later incorporation in other workflows, in the field of SST, within a collaborative and transparent environment.	Use of the exported results in subsequent work. Reproducibility of results across different teams. Participation in SST activities and workshops.

Table 1: SDG goals and targets considered in this dissertation.

Acknowledgements

I dedicate this chapter to expressing my gratitude to everyone who has supported me during this phase of my life:

To my family, especially my mom, who watched me grow up and were always there for me whenever I needed them, helping me to learn, question, experiment, discover, and dream over the course of more than twenty years, also providing me with love.

To my pets, who have always comforted me and have shown me a special bond and unconditional love.

To my friends, who made me laugh, but also supported me during the difficult times.

To my supervisor and co-supervisor, Professor Paulo Garcia and Professor Fernando Cassola, who provided me with guidance, help, and constant support, and who believed in me throughout the development of this dissertation.

To the STARPORT team, who welcomed me with such warmth and were always ready to help with whatever I needed.

To Professor José Sobrinho (Universidade da Madeira), José Lino and Observatório Óptico do Pico do Areeiro for providing telescope images used for dashboard features testing, user and performance testing and validation.

To all of the above, thank you for being a part of my life and for helping me find meaning in it.

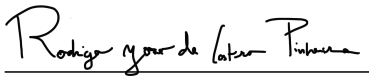
Rodrigo Pinheiro

Declaração de Honra

Declaro que a presente dissertação é de minha autoria e não foi previamente apresentada e avaliada nesta ou em outra instituição. As referências a outros autores (afirmações, ideias, pensamentos), assim como a utilização de trabalhos publicados respeitam escrupulosamente as regras da atribuição, e encontram-se devidamente indicadas no texto e nas referências bibliográficas, de acordo corn as normas de referenciação. Tenho consciência de que a prática de plágio ou de autoplágio constituem ilícitos académicos, conforme estabelecido no Código de Ética da U.Porto.

Declaro, ainda, que utilizei ferramentas de Inteligência Artificial para a realização de parte(s) da presente dissertação, e essa utilização e os seus resultados estão devidamente assinalados e foram objeto de cuidada verificação para deteção de erros, imprecisões, atribuições infundadas ou outras formas de enviesamento de conteúdos e argumentos, bem como de determinação de autoria.

<Rodrigo José de Castro Pinheiro>



Author Notes

The DeepL software [54] was utilized to improve the English in some parts of the dissertation originally written in English by the author. The result was scrutinized by the author to ensure scientific reliability.
In addition, the GPT-5.3-Codex model [41] was used as an agentic coding assistant to support the development of this dissertation's dashboard prototype. The author acted as the primary architect and supervisor, meticulously reviewing, testing, and validating all the results to ensure technical accuracy, architectural integrity, and scientific reliability.
Artificial Intelligence was also used to broaden the author's understanding of specific concepts, particularly those related to space surveillance and tracking, dashboard engineering, and visualization techniques. These resources were used exclusively as supplementary tools for clarification and exploration; all interpretations, decisions, and final formulations were the sole responsibility of the author.

Contents

Acknowledgements v

1 Introduction 1
- 1.1 Context . . . 1
- 1.2 Motivation . . . 1
- 1.3 Problem . . . 2
- 1.4 Research Questions . . . 2

2 Literature Review 4
- 2.1 Type of literature review adopted . . . 4
- 2.2 Why Web of Science? . . . 4
- 2.3 Keywords . . . 5
- 2.4 Research Procedure . . . 5

3 State of the Art 7
- 3.1 Fundamentals of data visualization . . . 7
- 3.2 Dashboards: definitions, challenges, and typologies . . . 9
- 3.3 Dashboard engineering . . . 11
- 3.4 Dashboards and visual pipelines in space and SST . . . 16
- 3.5 State of the art conclusions . . . 22

4 Methodology 24
- 4.1 Methodology overview . . . 24
- 4.2 Software engineering requirements and dashboard design . . . 25
- 4.3 Experimental process and evaluation design . . . 26
- 4.4 Expected results and relevance . . . 27

5 Requirements Elicitation 28
- 5.1 Elicitation process . . . 28
- 5.2 Requirements definition and organization . . . 29
- 5.3 Use cases . . . 32
- 5.4 Summary . . . 33

6 Architecture and Technologies 35
- 6.1 System architecture . . . 35
- 6.2 System technologies . . . 42
- 6.3 Summary . . . 46

7 Prototype Implementation **47**
7.1 Implementation strategy . . . 47
7.2 Backend implementation . . . 48
7.3 Frontend implementation . . . 50
7.4 Technical decisions, trade-offs, and challenges . . . 51
7.5 MVP and the requirements . . . 52
7.6 Illustrations . . . 53
7.7 Summary . . . 56

8 Evaluation **58**
8.1 Objectives . . . 58
8.2 Evaluation methodology . . . 58
8.3 Performance evaluation . . . 60
8.4 Task-based usability assessment . . . 62
8.5 User testing results . . . 63
8.6 Requirements verification . . . 68
8.7 Results discussion . . . 69
8.8 Limitations and future testing . . . 69
8.9 Summary . . . 70

9 Conclusions **71**
9.1 Limitations of the current prototype . . . 71
9.2 Conclusion . . . 71
9.3 Future Work . . . 72

References **74**

A Metadata structure for .fits files **78**

List of Figures

3.1 Gaia Archive Visualisation Service (Dashboard Case Study Example) 18
3.2 Example of a .fits image applied to SST, which will be used later in the development of this dissertation (mapa1.fits). 21

6.1 Logical View of the SpotDebris dashboard architecture 36
6.2 SpotDebris Architecture Layers 37
6.3 SpotDebris Architecture with Communication Flows 38
6.4 SpotDebris entry view, with the drag and drop or browse files features to upload .fits images and a check button to auto-run pipeline 44
6.5 The upload partial view, with a .fits file selected 45
6.6 SpotDebris main view - the image analysis interface 45
6.7 The history view, featuring some arbitrary objects and their confirmed or rejected debris status. From here it is possible to export a report 46

7.1 SpotDebris entry view, with the drag and drop or browse files features to upload .fits images and a question which, if hovered up, shows a small description of what the external pipelines consist of 53
7.2 The main view displaying the view-port of an uploaded, processed image, with no ASTRiDE streaks detected, but countless Gaia stars cataloged, featuring diverse astrometric properties and interactive tools 54
7.3 The candidates partial view, featuring the detected and manually validated candidates; in this specific image, there are no detections available 54
7.4 The summary partial view, displaying the summarized version of the analysis . . 55
7.5 The processing, informative messages, telling the user that the pipeline is running, displaying the images (results) in a brief moment later 55
7.6 Another processed image, in this case, two images uploaded and processed, and the selected one in the view-port contains a potential space debris streak, which was manually confirmed, as well as modified visual parameters by the user 56
7.7 The history page, featuring two images uploaded and analyzed by the user, one of them with a potential space debris streak confirmed by the operator, and another image with no visible streaks 56

List of Tables

1 SDG goals and targets considered in this dissertation. iv

2.1 Keywords used in the literature review search 5

3.1 The Seven Categories of Interaction According to User Intent ([62]) 8
3.2 Multiple-view layout patterns in dashboards (adapted from [6]) 9
3.3 Main approaches to dashboard configuration and adaptation. 11
3.4 Classification of System Usability Scale values according to [4]. 15

4.1 Initial dashboard requirements definition and evaluation tasks 26

8.1 Results of the heuristic review applied to the SpotDebris prototype. 60
8.2 Observed upload→preview times for 20 .fits files (headless test). 61
8.3 Workload evaluation questionnaire items. 63
8.4 Raw NASA-TLX scores (0–100) for the six workload dimensions across all participants. 63
8.5 System Usability Scale (SUS) questionnaire items. 65
8.6 Individual SUS responses (Q1–Q10) and corresponding SUS scores for all the ten participants. Here participants are not in numerical form for better visual distinction. 65
8.7 Technology Acceptance (TAM/UTAUT) questionnaire items. 66
8.8 Individual TAM/UTAUT responses (Q1–Q9) and corresponding TAM scores for all the ten participants. Here, also, participants are not in numerical form for better visual distinction. 66
8.9 Individual UEQ-S responses (Q1–Q8) and corresponding UEQ-S scores for all the ten participants. Participants are not in numerical form for better visual distinction. 67
8.10 Requirements compliance chart 68

A.1 Core .fits header keywords utilized by the Spot Debris backend. 78

List of Acronyms

ESA European Space Agency

MVP Minimum Viable Product

SSA Space Situational Awareness

SST Space Surveillance and Tracking

Chapter 1

Introduction

This dissertation explores the possibility of processing and analyzing telescope images to detect space debris by cross-referencing existing catalogs. The results are presented in an interactive dashboard to help operators improve monitoring efficiency and decision-making. The following sections cover the context, motivation, problem definition, and research questions.

1.1 Context

The exponential increase in the congestion of certain orbits around Earth, due to a growing number of operational satellites and debris, has intensified the need for accurate and reliable Space Situational Awareness (SSA) approaches. Space situational awareness involves monitoring orbital objects, space weather, and natural debris fluxes. This demands heterogeneous sensors and advanced data-fusion methodologies ([28]). Within this sector, Space Surveillance and Tracking (SST) plays an important role, dedicated to detecting, monitoring, identifying and predicting the movement of active orbital satellites, as well as orbital defunct objects (space debris) and the risk of collisions ([17, 31]).

SST capabilities provide a fundamental aspect in collision prevention and successful mission planning, ensuring the viability of sustainable space activities. The primary source of data acquisition for the above-mentioned process is mainly through hardware-based sensors, such as optical telescopes, tracking and surveillance radars. The development of an interactive data visualization dashboard that accepts an image as input, detects sources through existing software tools, and labels the outputs using catalogs is of significant importance and constitutes a valuable contribution to the SST ecosystem.

1.2 Motivation

The motivation for this dissertation arises from the increase in telescope data collected for space surveillance and the challenge this poses for efficient analysis and interpretation. Although there are classification and analysis systems for in-orbit objects, they are often fragmented and still

require a considerable manual intervention, which can be time-consuming and lead to workflow inefficiency. In this scenario, it is urgent to adopt faster and more automated data processing pipelines.

This dissertation addresses these challenges by developing an interactive and dynamic system, a dashboard for viewing telescope images of celestial bodies such as stars, but also satellites and other objects in Earth's orbit. In addition to the improved operational efficiency expected from utilizing the developed dashboard, automatic object detection based on catalog cross-referencing is integrated. With this project, greater reliability, optimized data interpretation, and support for more confident and robust decisions are expected. The goal is better informed and calculated decision-making in the SST area.

1.3 Problem

Spatial monitoring within the European SST context is based on the collection and evaluation of telescope images obtained from ground-based observatories, such as the Pico do Areeiro Optical Observatory, in Madeira, Portugal. These stations are essential for identifying and tracking active satellites and space debris. However, despite the wide array of observed data available, there is still no practical solution capable of processing, identifying, and displaying the information contained in the images.

Within the scope of this dissertation, Pico do Areeiro Optical Observatory is the primary data source for the collection of images to support the development and validation of the proposed system, contributing considerably to the European SST.

The major challenge consists of extracting and presenting useful information in a dynamic and interactive way, in order to correctly catalog the detected sources. In addition, there is the challenge of employing systems engineering methods to combine image processing tools, online catalog queries, and dashboard technologies into a fully functional system. By tackling these challenges, this dissertation promotes a practical and effective solution for SST operators to improve the performance of their functions.

1.4 Research Questions

The following research questions have been carefully designed and selected to meet the need for an interactive system. This platform is intended for cataloging and viewing telescope images originating from the Pico do Areeiro in Madeira, with particular emphasis on the labeling of space debris. These questions refer to the specific gap that a dashboard must bridge in the context of SST, ranging from data integration to interaction design and operational impact.

RQ1 What are key data sources to be fused in a dashboard to label the Pico do Areeiro Optical Observatory SST images?

RQ2 What requirements (interface, interactivity) should a dashboard addressing the Pico do Areeiro SST follow?

RQ3 How impactful will the contribution of a proposed visualization and annotation tool be for decision-making and work efficiency within the SST workflow?

Chapter 2

Literature Review

This chapter details the literature review conducted within the scope of this dissertation. The following subsections cover the methodology employed and its associated elements.

2.1 Type of literature review adopted

In the prepared literature review, a semi-systematic article search strategy was adopted, that is, a structured but flexible exploratory plan. This method is intended to map the state of the art regarding data visualization in dashboards, with possible applications in the context of space surveillance and tracking.

The search for information concerning the scope of interest was carried out within the Web of Science scientific knowledge database, carefully selecting and planning a list of keywords used to guide the referred research.

2.2 Why Web of Science?

Web of Science, IEEE XPlore and Google Scholar knowledge databases were initially considered to carry the literature review of this dissertation. However, Web of Science was selected because this digital library provides comprehensive coverage of multidisciplinary research, as well as robust filtering and search capabilities based on impact, such as category quartile, number of citations and references, and journal rank. These qualities allow for a refined and objective selection of high-quality publications, ensuring that the research is based on influential and reputable sources.

The option also aligns with the recommendations of the supervisor professor, with particular emphasis on the ability to filter articles based on relevance and scientific impact.

2.3 Keywords

With the knowledge of the dissertation proposal in mind, the context, motivation and research questions, the main keywords worked on and utilized for the literature review procedure are explicit in the table 2.1. These keywords were also manipulated into a combination of them and slightly paraphrased.

Keywords
dashboard
data source
data visualization
space debris detection
optical telescope tracking
astronomical catalog matching
SST telescope image

Table 2.1: Keywords used in the literature review search

2.4 Research Procedure

The research procedure began with a meeting with the supervisor, during which the research objective was defined and useful tools for conducting the research were discussed; in addition to the aforementioned database and how to best utilize it, tools such as Zotero, Mendeley or NotebookLM for collecting and archiving papers relevant to the study were also included ([19, 32, 63]).

The research started with the proper keywords, and papers were filtered based on the following criteria:
-Publication year (prioritize articles between 2018–2025);
-Prioritize journals ranked Q1/Q2;
-Papers with high number of citations, compared to the year published;

During the screening process, the titles and abstracts of papers were read to determine if they were of interest. Then, the selected papers were exported to the Zotero application to organize them. This tool was selected over the other two due to the author's prior familiarity with it. This enabled a smoother integration into the research process.

A new meeting was organized with the supervisor, in which the list of papers arranged in the application was demonstrated using PowerPoint slides. Each slide contained a screenshot of the abstract and title of the interesting papers, as well as the date of publication and other meaningful information. With the supervisor's help, the papers were further screened and filtered, with the irrelevant ones discarded, and the important ones given more attention.

The current list of papers is referenced using the BibTeX feature on the References page and is formatted according to the Chicago Manual of Style 18th edition (author-date).

Chapter 3

State of the Art

In the course of the chapter, ideas and concepts are organized into subchapters that address the fundamentals of visualization, dashboard design, dashboard engineering, and finally, applications/work that employ dashboards or visual pipelines in SST.

3.1 Fundamentals of data visualization

As advocated by [35], data visualization is an increasingly important form of scientific communication, although many scientists are still unaware of all its principles for better narrating the story behind their data. It is not a form of decoration; visualization should be understood as a way to convey the message quickly. Specifically, Midway states that “The reality for the vast majority of figures is that you need to make your point in a few seconds.” In addition to the difficulty of adapting or recognizing essential principles for data transmission and its meaning, ineffective methods are used in figures, tables, and means of data dissemination. These may not contain incorrect information, but they still use suboptimal visualization techniques. The author specifically mentioned that "Unfortunately, across scientific disciplines, many figures incorrectly present information or, when not incorrect, still use suboptimal data visualization practices." Figure conception should be carried out deliberately and intentionally; it should not be completely mechanized, as Midway states "[...] the process of displaying information cannot—and perhaps should not—be fully mechanized." Midway listed ten principles to guide authors, stressing the 1st principle, "Diagram First", the need to prioritize the message. Before showing the data, it is crucial to be clear about the message to be conveyed and to design it correctly, so as to make the entire process of data communication useful and conducive to accurate and disciplined scientific knowledge, with fewer margin for error and misinformation. As Midway states, "All scientists seek to share their message as effectively as possible, and a better understanding of figure design and representation is undoubtedly a step toward better information dissemination and fewer errors in interpretation."

In the work undertaken in this dissertation, emphasis is placed on this first principle mentioned in the article by [35], since high-quality visual communication and clear transmission of messages

(scientific data) are important for the development of an interactive dashboard capable of processing and analyzing telescope images, in this particular case. Robust and secure decisions require reliable and clear data.

Along with the importance of proper data visualization, interactivity is a valuable aspect for conveying the desired message to the user. Although interaction is a strong component of information visualization systems, it has not received the attention it deserves, compared to representation. Interaction is vital to transcending the limits of static representation and increasing user cognition. More precisely, [62] state that "While existing research in the area often focuses on representation, we highlight the overshadowed, but very important interaction component and strongly argue that it provides a way to overcome the limits of representation and augment a user's cognition."

Deep knowledge also comes from a "science of interaction." As proposed by [62], seven categories of interaction techniques are discussed, with the purpose of the user's intention on the surface, rather than low-level operations, seeking to link the user's objectives to the interactive techniques that bring them to fruition.

The seven categories mentioned, which serve as a common vocabulary for discussing and evaluating interaction techniques, are shown in Table 3.1.

Table 3.1: The Seven Categories of Interaction According to User Intent ([62])

Category	**User Intent**
Select	Mark something as interesting
Explore	Show me something more
Reconfigure	Show me a different arrangement
Encode	Show me a different representation
Abstract/Elaborate	Show me more or less detail
Filter	Show me something conditionally
Connect	Show me related items

Representation, as mentioned earlier, often receives more attention than interactivity, but it cannot be neglected. Multiple-view visualization is a layout design technique often used to help users see a large number of attributes and data values in a single cohesive representation. By allowing the user to simultaneously view representations of the same data from different perspectives, a well-designed multiple-view layout is a ubiquitous technique in exploratory data visualization. Formally, a multiple-view allows two or more views to be laid out in a view-port, and each view within a multiple-view has perspective attributes, such as the type of view (visually, the mapping of data to bar or line charts, for instance) and the bounding box, in this case, the size and position in the view-port.

Despite the central role of multiple-views in visual data analysis, the absence of concrete design guidelines for composition and spatial placement often forces designers to resort to manual curation through trial and error. As directly stated by [6]. "However, for more complex data and tasks, designers often still need to manually curate the layout of multiple-views through trial and

error. This process can be tedious and time consuming, and sometimes produces results that fail to meet design guidelines." To close this gap and better understand the space of a multiple-view, Chen et al. conducted an empirical study focusing on two key dimensions: Composition (measuring the viewpoints used) and Configuration (describing the spatial arrangement). By performing an analysis of real-world multiple-view designs, they found that designers tend to favor simple multiple-view layouts, with most designers incorporating fewer than five different views (62.2%), as well as a preference for more perceptually accurate view types, such as bar (32.2%) and line charts (32.5%).

Table 3.2 presents a selection of the most relevant concepts related to the organization and description of multiple-views in visualization dashboards.

Table 3.2: Multiple-view layout patterns in dashboards (adapted from [6])

Dimension	**Pattern**	**Description**
Number of views	2–5 views	Most dashboards are based on simple layouts. In fact, 62.2% of multiple-view designs contain five views or less.
Spatial organization	Slice-and-dice	Recursive partitioning of the display area either horizontally or vertically.
Visual hierarchy	Dominant view	One singular view highlighted more than alternative views.
	Small multiples	Multiple occurrences of the identical view type for comparative analysis.
View composition	Heterogeneous	Integration of various chart styles within the same dashboard.
Auxiliary elements	Panel	Textual elements, filters and legends included in approximately 68% of dashboards.
Aspect ratio	Balanced proportions	The majority of views display aspect ratios ranging between 0.5 and 2 to promote readability.

3.2 Dashboards: definitions, challenges, and typologies

Today, with the enormous amount of information available, there is a great dispersion of knowledge and data. Thus, there is a need to organize data sets into databases, as well as to extract them using tools such as visualization dashboards, for their usage and understanding. Since the focus of this dissertation is on dashboards, the very definition of this type of information and data dissemination tool is in flux, evolving from simple monitoring screens to large interactive interfaces with various purposes, namely communication and learning.

The evolution of the definition is due to the dichotomy between the “visual genre,” which, according to [52], consists of “a visual data representation structured as a tiled layout of simple charts and/or large numbers” and the “functional genre,” which, from the perspective of the

same authors, is precisely “an interactive display that enables real-time monitoring of dynamically updating data”. In the scope of telescope image visualization (in the field of SST), this duality is vital. The importance of this distinction is deserved because the dashboard should not only be able to represent captured images (visual), but also allow for the functional and operational monitoring of space objects (functional).

In typological terms, dashboards are characterized by the decision support they offer. This support can be strategic (i.e., focused on long-term goals), analytical/tactical (i.e., a summary of evaluation metrics or intermediate analysis), or operational (i.e., for monitoring immediate or quantifiable activities). Given the context of this dissertation, the literature highlights the need for customized solutions, i.e., tailored dashboards (related to more domain complex systems), capable of varying their appearance and functionality in response to the evolution of more specific user requirements and context. The design of this type of tools or solutions is not trivial, which implies the need for tailored displays focused on “specific requirements, goals, user roles, situations, domains.” ([58]) This is mainly due to the increasing popularity of these solutions, as pointed out by the authors.

However, the effective development of robust and consistent dashboards encounters some critical challenges, notably visual overload and a lack of visual literacy on the part of users, i.e., the ability of a certain type of user to effectively understand what they are viewing on the dashboard. “One-size-fits-all” is a common concept used to describe a standard that can serve a large group of users, with default settings. The problem with this approach is that it is often pointed out as utopian; “not every user is driven by the same goals, not every user is interested in the same data, not every user has the same visualization literacy", as [58] mention. In order to prevent inconsistency and failure in decision support, it is imperative to integrate customization (explicit approach), personalization (implicit approach), and adaptation (real-time) capabilities, to allow users to adapt the layout and filters to their analysis needs. Furthermore, in the case of dashboards where data visualization is performed in real-time with constant changes in content, accuracy, and responsiveness are essential. These adaptive capabilities become the difference between effective detection of momentary information and loss of critical information due to visual overload. The three main approaches mentioned above are shown in the Table 3.3.

Table 3.3: Main approaches to dashboard configuration and adaptation.

Approach	Description
Customization	The user directly controls changes through: • **Component arrangement**: Organize the layout and position elements through direct interaction. • **Selection of widgets and data sources**: Choose predefined metrics and specific data streams for visualization. • **Indicator creation**: Building custom metric indicators. • **Visual mapping**: Wizards that help determine the best type of chart or graph for the selected data structure.
Personalization	The system infers the ideal configuration based on: • **Roles and objectives**: Automatic display generation based on user role (e.g., manager vs. technician) and business goals. • **Accessibility**: Adjustment of the visual design according to the user's physical capabilities, such as vision impairments or tremors, detected via initial questionnaires.
Adaptation	Ability of the dashboard to dynamically restructure itself during usage, based on analysis of the user's interaction history and behavior.

3.3 Dashboard engineering

3.3.1 Engineering etymology and the connection with dashboards

The word engineering, from Latin *ingenium*, "ingenuity," "natural ability," or "ability to create a solution," evolved into the now Old French *engin*, meaning an engine, device, or mechanical contraption. The word engineer, in Old French *engigneor*, is one who practices engineering, "ingenuity", was commonly associated with builders of war machines in the Middle Ages. A semantic expansion allowed the meaning of the word engineering to be broadened to something like "applying technical knowledge to solve problems."

Dashboards, as mentioned earlier in this chapter, are "artifacts" capable of provoking knowledge, transmitting information, and supporting decisions, just like bridges, engines, or mechanical systems. A dashboard engineer is one who designs, structures, and optimizes dynamic and visual information flows.

For optimal dashboard execution, methodological rigor is required, for instance: requirements definition, use cases, architecture, prototyping and design, validation, and iteration; in other words, a set of guidelines to follow. The requirements, use cases, and conformance to standards like ISO

and ECSS represent, from an engineering point of view, core engineering artifacts, which constrain, shape, and validate the system. Furthermore, usability and interaction are engineering constraints rather than aesthetic concerns, thereby strengthening the idea that dashboards are engineered systems designed to support decision-making processes. The subsequent sub-chapters will address proper dashboard engineering.

3.3.2 Dashboard design patterns

According to the existing literature, there are several dashboard design patterns, providing a bridge between engineering and the visual component. In a study conducted by [3], the authors observed a total of 144 existing dashboards and concluded that there are 42 design patterns. Unified Dashboard Design Space is the organization that gives the study its name, where, in addition to the number of patterns mentioned above, two major groups are distinguished: Content Patterns and Composition Patterns. Content Patterns define what is presented, i.e., from levels of information abstraction (single values, derived values, and aggregated datasets) to ways of visual representation (pictograms, miniature charts, tables, or detailed visualizations).

Composition Patterns serve to specify how these elements are structured, encompassing page layout decisions (stratified, tabulated, or grouped), greenspace management (screenfit, overflow, detail-on-demand), and interaction mechanisms (filters, drilldown, and navigation). By revealing the existence of inevitable trade-offs, these patterns also demonstrate that dashboard design is limited to the available space, requiring compensation in other areas, such as greater abstraction or interactivity, in order to balance the communicational effectiveness of the system. Showcasing the most vital information in the foreground is absolutely fundamental, characterizing the gold standard in dashboard design.

The authors also introduce the classification of “dashboard genres,” which include analytic, magazine, and infographic, for instance; revealing that different combinations of patterns are required to serve different specific purposes, ranging from more detailed technical exploration to journalistic narrative. The tactical application of a color scheme in a dashboard is also a vital composition pattern, encompassing semantic coding associated with positive or negative states, good or bad, that is, shared and cross-cutting color schemes that promote visual consistency across multiple graphics and views (multiple-views concept in 3.1).

In summary, and as [3] stated the core message of their work: "[...] abstraction, screenspace, number of pages, and interactivity. In a design process, the goal is to minimize each of these parameters. For example, to fit as much information as possible (low abstraction) into the least amount of screenspace, with no interaction, and to only show a single page. Such a solution would possibly be the gold standard in dashboard design: showing all important information at a glance, without the need for costly interaction."

In addition to the conceptual decisions captured by dashboard design patterns, the technical implementation of these patterns encounters some underlying challenges, which are mentioned and presented by [61] in their findings, more precisely as: "We report observations and reflections from five large multidisciplinary visualization design projects and highlight six data-specific

visualization challenges for design specification and handoff. These challenges include adapting to changing data, anticipating edge cases in data, understanding technical challenges, articulating data-dependent interactions, communicating data mappings, and preserving the integrity of data mappings across iterations".

The authors explain that the handoff (transfer) stage between initial design and actual development is critical because data input is a catalyst for breaking an existing dashboard layout design. Even when carefully designed, layouts can undergo significant changes due to minor alterations in datasets, thus posing challenge no. 1 (C1). A further issue would be that changes to the dataset could expose edge cases that were not anticipated during the design phase, such as filters related to scales or displaying/hiding visual elements (C2). Whenever composition patterns (mentioned earlier, such as screenfit, overflow, or detail-on-demand) depend on assumptions about data distribution, these challenges become even more critical. Often, communication errors stem from flaws in data mappings (C5), and [61] warn that, for example, minute details such as mapping values to area rather than to the diameter of a circle make all the difference and can undermine the desired visual semantics.

Therefore, dashboard engineering is much more than just applying design standards. It goes beyond, ensuring that patterns remain solid in the face of dynamic datasets, multi-faceted interactions, and multidisciplinary teams, reaffirming the need for thorough documentation, constantly evolving prototypes, and consequent recurring validation throughout the development cycle.

3.3.3 Heuristics and methods for dashboard evaluation

The vast majority of guidelines focus on the individual user, but since the purpose of developing a dashboard to analyze SST images is to be used by a team of specialists in the field, the dashboard must be understood as a “boundary object,” that is, it must serve as a meeting point for different specialists. According to [55], the most relevant out of the 39 heuristics they propose, focused on communication, are:

1. **Initiation**
   (a) Provide clear initial context.
   (b) Define key concepts and metrics.
   (c) Show clear entry points for exploration.
2. **Grounding**
   (a) Provide sufficient context to interpret data.
   (b) Maintain visual and semantic consistency.
   (c) Communicate uncertainties and limitations.
3. **Turn-taking**
   (a) React clearly to user actions.

   (b) Provide immediate and visible feedback.

   (c) Support multiple interaction paths.

4. **Repair & Refinement**

   (a) Allow actions to be easily undone.

   (b) Make ambiguity visible and resolvable.

   (c) Explain why changes occur in the dashboard.

5. **Close**

   (a) Provide clear summaries of the analysis.

   (b) Highlight key insights.

   (c) Allow exporting or sharing results.

In addition to the collaborative conversational heuristics proposed by the authors mentioned above, [10] focus on the traditional validation practice of dashboards, reinterpreting Nielsen's 10 Heuristics, originally designed for more generic interfaces. This adaptation can be considered pertinent in more technical contexts, such as the development of a dashboard focused on SST, where transparency and alignment with the domain are paramount.

[10] highlight, above all:

1. **Visibility of System Status**: In the context of dashboards, this means clearly communicating information on the freshness of the data (e.g. the date of the last update) and the current status of any applied filters.

2. **Match between System and the Real World**: The visual and textual language must correspond to the technical jargon and expectations of experts in the field.

3. **Consistency and Standards**: Colors, symbols, and visual conventions should maintain stable meanings across all dashboard views.

Thus, while [55] provide heuristics geared toward communication and collaboration, [10] offer a complementary framework focused on classic usability, allowing for a more complete and rigorous evaluation of the dashboard.

In addition to the heuristics of previous authors analyzed in this subchapter, it is equally important to learn about standardized metrics that allow for quantitative assessment of the perceived usability of a dashboard. This is where the concept of System Usability Scale comes in. Developed by Brooke and later studied in detail by [4], the System Usability Scale is one of the most reliable and widely used tools among software engineers and interactive system designers for measuring the degree of satisfaction and perceived usability of these solutions. Over a period of ten years, the authors identified interpretable usability ranges on the System Usability Scale from α ranging between 0 and 100, based on the 2,300 responses obtained in their large-scale analysis.

- The System Usability Scale demonstrated **high reliability** ($\alpha \approx 0.91$).
- Applicable to various types of systems: *software*, websites, hardware, and dashboards.
- Identification of **interpretable intervals** on a scale of 0–100.
- Introduction of qualitative labels: **marginal**, **acceptable**, **excellent**.
- Strong correlation between System Usability Scale scores and **actual perception of usability**.

These usability ranges are summarized in Table 3.4.

Table 3.4: Classification of System Usability Scale values according to [4].

System Usability Scale Range	Rating	Interpretation
0–50	Marginal (Low)	Poor usability; serious problems.
50–70	Marginal (High)	Acceptable usability, but needs improvement.
70–80	Acceptable	Good level of usability.
80–90	Excellent	Very high usability.
90–100	Best imaginable	Maximum level of satisfaction.

This metric will later be used to assess the quantitative user acceptance and satisfaction of the proposed dashboard for this dissertation.

In summary, the contributions of these three frameworks discussed in this subchapter offer complementary perspectives: communication, measurement, and usability. Together, it is possible to support robust dashboard evaluations.

3.3.4 Dashboard usability evaluation and the link to interaction

As already mentioned, in an engineered dashboard, usability and interaction are not decorative additions, but core constraints that reveal whether the system can truly support human operators in real time for critical decision-making. Dashboards are the epitome of human-computer interaction, especially in contexts of information overload, and their effectiveness depends on how efficiently their users can perceive, interpret, and act on the information displayed. Beyond the System Usability Scale heuristics studied previously, to evaluate the usability of a system (dashboard), [1] explicitly address usability evaluation through a systematic review of surveys conducted on healthcare dashboards. The authors concluded, that in their studies, the preferred tools for usability evaluation are the System Usability Scale, Technology Acceptance Model (TAM), Questionnaire for User Interface Satisfaction (QUIS), or Information Technology Usability Evaluation Scale (Health-ITUES), but that they cannot capture all the particularities of a dashboard system because they were not originally designed for dashboard assessment. Based on the 29 studies carried out by the authors, it was concluded that the following metrics are useful for properly evaluating

dashboards: usefulness, operability, learnability, ease of use, task suitability, situational awareness, satisfaction, user interface, content, and system features.

More specifically, as the authors say: "When selecting the usability evaluation criteria for dashboards, it is important to pay attention to the evaluation objectives, dashboard features and capabilities, and context of use." ([1])

In operational contexts, a dashboard is not just a visual interface, but a dynamic, interactive platform that requires mental focus and situational awareness. Usability, in this case, goes beyond the traditional “ease of use”: it involves cognitive effort, processing of data-derived information, and the ability to act under time pressure. Therefore, evaluating a dashboard involves not only ensuring that the interface is intuitive but also understanding how interaction design directly influences human performance in tasks where, for example, attention to detail is of paramount importance.

To complement “What to measure?” from the perspective of [1], [39] introduce the perspective “Why measure?”, and their study focuses on how interactive analytical features affect human performance and situation awareness in a decision involving operational support. In a test conducted with analytical dashboards in a controlled laboratory, where the purpose of these dashboards is production planning, the authors revealed that adding a “what-if” analysis can substantially improve task performance, but at the same time reduce operators’ situation awareness and foster “out-of-the-loop” effects. The results obtained by the authors demonstrate an important trade-off: interaction can increase decision quality and efficiency, but poorly balanced automation and opacity can reduce the operator’s understanding of the situation in front of them. According to the authors, "[...] by introducing an interactive analytical feature, operational decision-makers can interact with the optimization system without knowing the optimization model or understanding how the optimization procedure behind the results operates. Such behavior can harm an individual’s comprehension of the current situation, nourishing an out-of-the-loop problem—a situation in which a human being lacks sufficient knowledge of particular issues." ([39])

In conclusion, usability and interaction emerge as essential engineering requirements to ensure a robust and useful dashboard for human operators. Thus, the literature shows that the effectiveness of a dashboard depends both on its technical architecture and its ability to preserve situational awareness and human performance in operational scenarios.

3.4 Dashboards and visual pipelines in space and SST

It has been learned that, in order to design effective and useful dashboards to support important decisions, it is necessary to consider the concepts of dashboard engineering as well as the user(s) in mind. In the field of space, more precisely in the area of study of this dissertation, SST, the development of a proper dashboard for analyzing previously acquired and processed telescope images is highly beneficial to operating scientists monitoring objects in Earth orbit, namely space debris. In this way, dashboards in the field of space are a visual analysis pipeline, helping operators explore, filter, and validate observational data.

This subchapter reviews existing visualization systems used in the space domain, including both general and SST specific tools, identifying their engineering limitations and highlighting the problems that motivate the creation of an interactive dashboard tailored to SST.

3.4.1 Existing visualization systems in space: insights and a case study

Considering the enormous volatility of emerging data in the context of space, SST operators often have to deal with images, catalogs, telemetry, and detections in real-time. This requires them to filter, explore, validate, and interpret emerging data quickly and effectively. In this context, dashboards are a fundamental pipeline, not just interfaces for entertainment (far from it); they consist of scientific tools for operational analysis.

Due to the considerable heterogeneity, volume, and criticality of the data relating to the field of study in question, dashboards providing support in this context face particular requirements. For instance, monitoring space debris requires systems capable of integrating measurements from multiple sensors (in this case, telescope images) and representing complex orbital flows, given that "ground-based radars, optical telescopes and laser systems are employed to detect, track and catalogue objects". ([38])

The detection and processing of (telescope) images requires effective dashboards that can display images on screen with the results of previously performed processing algorithms (including streaks, tracklets, or even altitude estimates), enabling dashboard operators to quickly interpret the results generated by automatic methods. In a context such as SST, this capability is essential for validating debris detections, monitoring uncategorized objects, and supporting expert operators' decisions in real-time. According to [17], "Optical space debris observation is performed by detecting faint and small moving objects in a background of stars."

Everything is integrated into a closed visual environment, enabling easy image analysis and, for example, the evolution of space debris orbits. Finally, according to [38], effective SST operations depend on systems capable of integrating heterogeneous measurements from multiple sensors and presenting them through visual interfaces that support tracking, validation, and decision-making. ESA's ISOC Suite exemplifies this need by providing an integrated platform for processing and correlating measurements from different SST sensors, supporting orbit determination and validation of tracked objects ([14, 37]).

A concrete and real example of a scientific dashboard in use, which responds to many of the challenges identified and meets the requirements mentioned in the literature review carried out previously, is the Gaia Archive.

This is the official European Space Agency (ESA) portal that allows users to explore data from the Gaia mission (astrometry space telescope, launched in 2013). The Gaia Archive (fig. 3.1) is a large-scale analytical dashboard capable of handling the volume and complexity of astronomical data, as well as its great heterogeneity ([36]). This dashboard integrates multiple-view (tables, celestial maps, HR (Hertzsprung-Russell) diagrams, which are crucial graphs in astronomy that relate the luminosity (brightness) of a star to its temperature, showing its evolutionary phase and stellar mass. It also incorporates interactive filtering tools and an ADQL (Astronomical Data

Query Language) query engine, which allows data to be extracted, validated, and interpreted in near real time. This dashboard incorporates these tools, which allow for the inspection of individual sources and the comparison of statistics from catalogs with over a billion objects.

Given its architecture and functionality, this is a brilliant case study for scientific visualization dashboards, where interaction, visualization, and analysis are perfectly combined to support complex, informed, and effective decisions, supporting operators, and inspiring best practices applicable to the detection, monitoring, and analysis of space objects.

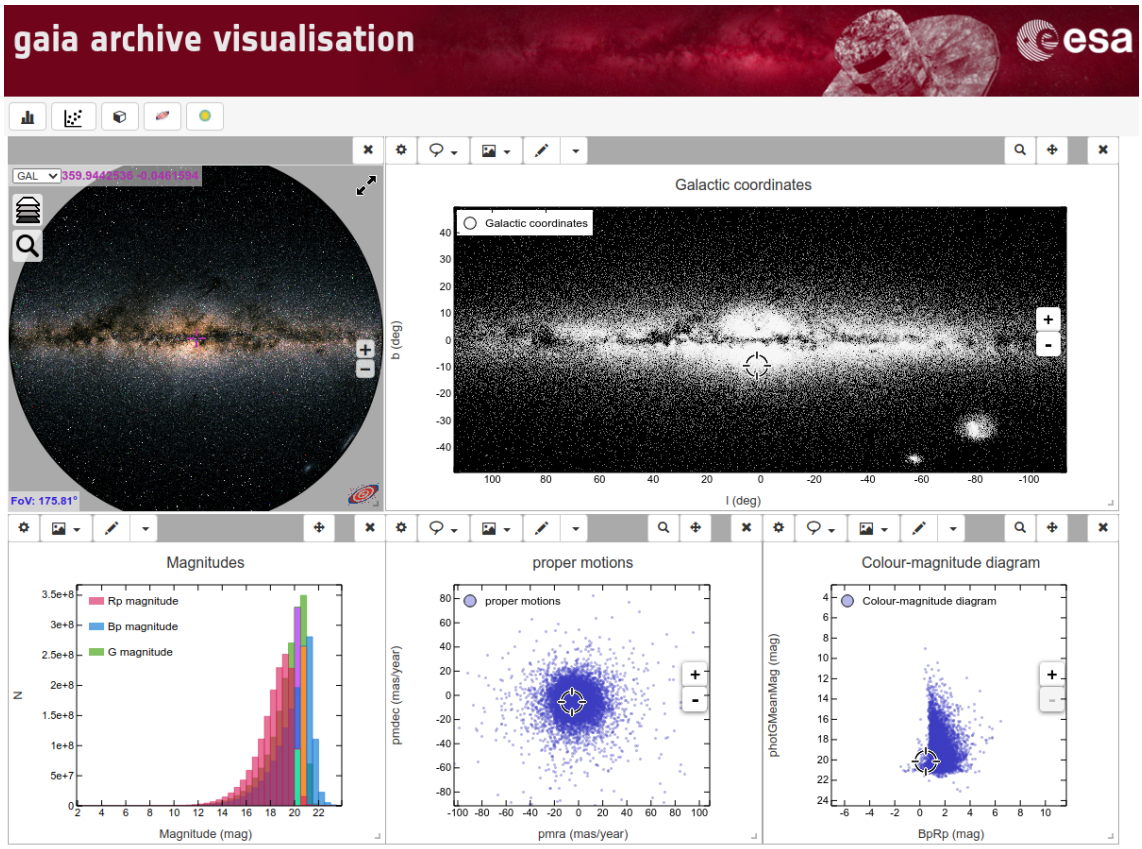


Figure 3.1: Gaia Archive Visualisation Service (Dashboard Case Study Example)

This dashboard case study fulfills the “Diagram First” principle from [35]: It features a celestial map that conveys the message clearly and concisely. In addition, various categories of interaction, such as selection, exploration, filtering, etc., are integrated according to [62]. According to [6], the concept of multiple-views is important. This dashboard meets the concept in that it presents various views, a variety of charts, a dominant view, and balanced proportions. Furthermore, as [3] points out, this tool features a well-organized layout and efficient use of space, as well as attention to detail. The heuristics from [55] are also followed: immediate feedback, consistency, and clear entry points. Also, The adapted heuristics from [10], such as domain terminology, visual consistency, and visibility of applied filters, are built into the tool. However, the dashboard has limitations identified by [1, 39]: a steep learning curve, cognitive complexity, and a lack of mechanisms to support real-time operational situations.

3.4.2 Challenges from a dashboard engineering perspective

This section addresses a list of general limitations of dashboard engineering applied to the context's domain.

1. **Visual and cognitive scalability**
    - Scientific dashboards host enormous catalogs, with millions of objects.
    - Human limitations in interpreting extensive maps, tables, and densities, even with advanced tools.
    - Dashboards should avoid overwhelming the user with excessive information.
2. **Integration of heterogeneous data**
    - SST data includes images, catalogs, telemetry, detections, uncertainties, and algorithmic outputs.
    - Heterogeneity can cause inconsistencies in visualization.
    - Normalizing data formats may not be trivial.
    - Integrating heterogeneous data into a single visual environment requires complex pipelines.
3. **Latency and near real-time updating**
    - Engineering must balance accuracy and speed.
    - In scientific research, real-time detections and alerts can be crucial.
    - Dependence on data collection and processing pipelines may introduce delays that affect rapid decision-making.
4. **Limitations of interaction in multiple views**
    - Scientific dashboards typically require multiple coordinated views (MCV).
    - Simultaneous interaction with several views can become tedious and complex for operators.
    - Synchronizing tables, maps, and graphics may degrade system performance.
5. **Steep learning curve**
    - Scientific dashboards require domain knowledge and familiarity with parameters, concepts, and filters.
    - Engineering must balance usability with analytical specificity.
6. **Limitations in representing uncertainty**
    - The field depends heavily on errors, uncertainties, and probabilities.
    - Representing these elements visually is a challenge in dashboard engineering.

- The lack of universal visual standards can lead to inconsistency.

7. **Dependence on external pipelines**

   - Dashboards may depend on detection algorithms, processing pipelines, catalogs, and external services.
   - Any upstream failure can significantly affect the operator's experience.

3.4.3 SST images and their implications to dashboards

It is important to note the nature of the images used in SST workflows. Unlike generic astronomical images, these observations typically consist of optical frames sequenced over time, in which stars appear as point sources and fast-moving objects (e.g., debris or satellites) appear as streaks. These images have their own metadata, such as celestial coordinates and image field of view, for instance. All of these metadata are essential for validating the different objects present in the images. Since these images serve a main operational purpose, the dashboards must allow for rapid inspection, comparison, and interpretation of these visual clues.

To contextualize the type of data used in SST, Figure 3.2 is an example of a .fits image, the standard astronomical data format defined in the Flexible Image Transport System specification by [45]. This image format is widely used in these space surveillance and tracking workflows. The image illustrates the visual characteristics typically encountered during optical observations of space objects. The common structural and astrometric metadata of this type of telescope image files are detailed in Appendix A (Table A.1).

Figure 3.2: Example of a .fits image applied to SST, which will be used later in the development of this dissertation (mapa1.fits).

However, it should be noted that dashboards that feature images of this type must adhere to specific standards. Below is a comprehensive list of implications for SST Image-Centric Dashboards, which emerge from the operational characteristics described in the typical SST and astronomical workflows.

1. **Need for real-time visual feedback**

    - Dashboards must support real-time visualization of processed images, enabling rapid validation of detected objects, namely space debris in-orbit.

2. **Integration of algorithmic outputs into the visual workflow**

- Detection, classification, and orbit determination pipelines must constantly feed information into the dashboard, ensuring that operators can interpret results without needing external tools.

3. **Support for heterogeneous data overlays**
   - Integrating images, catalogs, uncertainties, and other data requires interfaces that can interleave multiple layers of information without degrading readability.
4. **Scalable interaction with large image sequences**
   - SST uses temporal image sequences, so dashboards must allow users to navigate, compare, and filter multiple frames without compromising system performance.
5. **Visual encoding of uncertainty and confidence levels**
   - Dashboards must represent uncertainties, probabilities, and errors intuitively, because detections can be ambiguous and there is always the possibility of human error.
6. **Human-in-the-loop validation**
   - Human validation is crucial because, despite increasing automation, there is always a need for a human mind to confirm what appears, and dashboards must facilitate manual inspection, annotation, and validation of detections.
7. **Operational robustness and fault tolerance**
   - Since dashboards in this area depend on external pipelines, they must be able to handle upstream failures, latency, or data loss, mitigating the impact on the user experience.

3.5 State of the art conclusions

In conclusion, the literature reviewed reveals the construction of effective visualization and decision support dashboards, taking into account the combination of the principles of data visualization, software engineering, and human interaction. However, when these principles are discussed in the context of SST, several limitations have to be taken into account. Existing dashboards provide valuable information on interaction techniques, multiple-view coordination, uncertainty communication, and analytical exploration, yet they do not fully address the operational characteristics of telescope-image, space-debris monitoring and star metadata for distinction between debris and stars.

The State of the Art reveals clear gaps when compared with the research questions defined in Chapter 1. These include, for instance, the absence of an interactive, focused on engineering oriented, image based dashboard tailored to sequential .fits files inspection for debris annotation and star-catalog cross-matching; and limited integration between detection pipelines and human-in-the-loop validation.

Therefore, the absence of a dedicated dashboard for the previously mentioned operational needs of SST motivates the development of the prototype proposed in this dissertation. At the end of this literature review, the conceptual and technical foundations that guide the development of an interactive dashboard are established. The following chapters, namely Chapter 4 address the engineering approach, as well as, the requirements, architecture of the system, the implementation strategy to address these limitations, results and their respective evaluation.

Chapter 4

Methodology

This chapter focuses on the methodology followed in this dissertation to design, implement and evaluate an interactive dashboard for SST image monitoring practices. By applying the knowledge of the previous state of the art, Chapter 3, following a software engineering approach, which means, translating an initial problem into requirements, designing a solution, and validating that solution to ensure it meets those requirements using specific, domain-related evaluation methods.

4.1 Methodology overview

The methodology for the development of an interactive, SST specific dashboard combines its core engineering process and an image analysis workflow.

This dashboard design process is achieved through a series of specific tasks and evaluation criteria, which serve to assist in choices and decision-making regarding the visualization and interaction of elements on the dashboard; in the main view (image, position of elements, results) in contrast to other secondary views, and the exploration mechanisms (filtering, validation, exporting). Both the system architecture and the choice of software engineering technologies are driven by non-functional requirements, such as the target for expert usability and low latency of use.

The preliminary image analysis pipeline can be summarized as follows:

(i) import an observed telescope image

(ii) detect and profile sources

(iii) validate detections by cross-referencing them with external catalogs

(iv) showcase results in a dynamic interface which allows visualizing, filtering, and exporting the findings

This schematic reflects the duality of dashboards described in the literature: a strong *visual component*, followed by a robust *interactive component* ([52]).

The methodology is organized in the following phases:

(a) *Requirements workshop & foundation*: establish user needs, system requirements, and main evaluation objectives.

(b) *Architecture & prototyping*: define the dashboard structure, technologies, main view and secondary views, and the flow of interaction through iterative prototyping.

(c) *Implementation (core workflow)*: implement the image analysis pipeline and integrate it with the dashboard.

(d) *Interfacing & validation*: validate integration of external catalogs and ensure performance, latency, and annotation export.

(e) *Evaluation (heuristics & tasks)*: evaluate usability and task performance through heuristic reviews, task-based tests, and System Usability Scale.

4.2 Software engineering requirements and dashboard design

Requirements engineering is the driving engine for the conception of a dashboard, following the principles studied in the literature review, the *diagram first* is an important concept, underlining the importance of learning the core message before designing the visuals ([35]).

4.2.1 Initial requirements

Based on a meeting with SST experts, there will be a discussion of what is considered essential for an interactive visualization platform for the domain in question. The expected result of this meeting is a list of requirements, in priority order as agreed by experts for implementation, as well as a small set of evaluation tasks, respecting the needs of dashboards guided by role, goals, and context ([58]).

Before this meeting takes place, a set of functional and non-functional requirements is gathered in a table. The former define what the system does, while the latter determine how the system works. This table of preliminary requirements derives from a preconceived idea of systems suitable for visualization and exploration in the area, and also serves to guide potential future discussion at the meeting.

Table 4.1 summarizes the initial requirements and their evaluation link.

Table 4.1: Initial dashboard requirements definition and evaluation tasks

Type	**Requirement**	**Indicator**
Functional	Import an image	Time to first render (s)
Functional	Cross-reference an imported image with existing catalogs	Expert acceptance rate (%)
Functional	Display an image with basic control features	Task completion rate (%)
Functional	Export results (annotated image)	Export success rate (%)
Non-functional	Low interactive latency	Measured latency (ms)
Non-functional	Learnable and usable for domain experts	System Usability Scale ([4])
Non-functional	Consistent and transparent overlays/mappings	Heuristic review ([10])

This dissertation follows a systems engineering approach consistent with the *NASA Systems Engineering Handbook* ([40]) and ECSS standards ([12]). It is adopted namely through explicit requirements definition, identification of system interfaces, architectural decomposition, iterative prototyping, and validation activities aligned with SST operational needs.

4.3 Experimental process and evaluation design

The experimental process is divided into two phases. The first consists of a heuristic review, applying dashboard focused heuristics and collaborative-analysis principles to identify preliminary usability issues. The second phase evaluates user task completion through standardized usability metrics, focusing on representative SST image-analysis tasks to assess performance, workload, perceived usability, acceptance, and overall user experience.

4.3.1 Heuristic review

It is important to assess the potential for initial problems through heuristics and usability testing in an initial evaluation cycle before launching the final product to end-users. From classic heuristics applied to dashboards (e.g. visibility of system status, match with domain language, consistency) [10], to heuristics dedicated to collaborative efforts in analytical systems (e.g. grounding, turn-taking and repair/refinement) ([55]), it is expected to achieve an outcome where a set of must-have changes are implemented, to potentially improve future expert's work performance.

4.3.2 Task-based usability assessment

After the early usability assessment, it is crucial to evaluate the dashboard using a task completion protocol in which a set of users test the dashboard through the completion of specific domain tasks (e.g., validate detection with proper catalog match; export a result: annotated image). After that, they complete a series of questionnaires, compiled in an online survey, regarding their perceived usability, overall performance and satisfaction with the dashboard. More details on these questionnaires are discussed in Chapter 8.

4.3.3 Additional evaluation metrics

The evaluation design also includes pipeline performance tests, measuring upload-to-preview latency and variability across various .fits files. A requirements verification was also included, which is to confirm that the developed prototype aligns with the previously stated requirements. Finally, comments and qualitative feedback from users were collected.

4.4 Expected results and relevance

The expected results are fundamentally, a dynamic, interactive dashboard, capable of providing ease of use for operators in the area, and also, above all, a valuable tool for performing their tasks more efficiently.

The prototype developed is of significant importance, as it is expected to help improve the interpretation of results by operators, reduce mental effort, and ensure engineering precision through traceable requirements and subsequent validation, for potential future scalability.

Chapter 5

Requirements Elicitation

This chapter explores the fundamentals of a robust software system designed to meet user needs and operational objectives. A dashboard developed to the analysis of telescope images for the detection of potential space debris requires a solid and technically viable design framework that is supported by recognized scientific practices.

A meeting was organized with domain experts, and the fundamental requirements for developing this software prototype were established, thereby guiding design priorities as well as the choice of technology and system architecture, which will be further discussed later.

5.1 Elicitation process

Understanding the business goals of stakeholders is key to understanding and designing new software that can meet the needs and context of the problem to be solved for users. An incorrect or incomplete comprehension of what future users of the developed software intend to see implemented is a sure way to solve non-existent and irrelevant problems, resulting in a loss of time and resources.

As mentioned earlier, the context of this dissertation consists of a functional prototype which is far from an advanced software system in active production. Therefore, few users are expected to use this prototype, which limits, to a certain extent, the techniques used to collect requirements and also limits iterative development. Workshops or in-depth surveys were dismissed for this elicitation of requirements given the small number of participants and schedule limitations, and a single meeting with domain experts was held.

During the meeting with domain experts, a semi-structured interview was conducted, and an initial mock-up (which had been developed based on knowledge and information previously acquired about the context) was presented, providing a unique view of what could be the core of the dashboard. The stakeholders involved provided appropriate feedback on the visual mock-up, revealing the aspects they considered most important and which they would prefer to highlight. They also suggested changes and outlined features and attributes that would be beneficial to include in the future prototype.

Despite the small number of participants, they represent a fundamental cornerstone for establishing the first version of the requirements. Notes were taken during the meeting to ensure that no details were lost before the requirements were finally formalized.

5.2 Requirements definition and organization

In software engineering, requirements validation and documentation is a vital step. This stage involves sharing a common understanding among stakeholders about the scope of the project and the expected features to be implemented. A formal requirements document serves as a structured guide to help the software development process, as well as aligning the expectations of users, who in this case are Space Surveillance and Tracking operators, with the project's objectives. With requirements engineering, ambiguity during implementation and subsequent verification and validation is minimized.

For this dissertation, the documentation of requirements is guided by the best practices of **ECSS-E-ST-40C Rev.1** standard (Space Engineering - Software General Requirements by [12]). This standard is applicable to space-focused systems and applications, providing a working foundation for defining clear, verifiable, and retrievable software requirements.

Adhering to the methodology proposed by the [12], this document distinguishes between functional and non-functional requirements. In order to ensure clarity and rigor in the implementation of requirements, precise linguistic conventions are utilized.

- **"Shall"**: Defines a mandatory requirement. These features are critical to the system to ensure a minimum viable product (MVP) and must be implemented for the system to be considered successful.
- **"Should" / "May"**: Defines a requirement that is optional and can be implemented later in development. These features make the system richer, but they are not needed for it to function properly.

By following this division and linguistic conventions, it is possible to distinguish core functionalities such as image ingestion and catalog cross-referencing, allowing for the solid development of the dashboard.

5.2.1 Functional requirements

The functional requirements refer to the proper functioning of the system and the specific behavior that the system must perform. They derive directly from the session held with experts in the field and are in line with the objectives of telescope image analysis for the detection of space debris.

For the purpose of their definition, they are divided into two categories:

5.2.1.1 Backend & data processing functional requirements

Regarding the computational functions processed on the server side.

- **FR-IMG-01: Image import & handling**
  - **FR-IMG-01.1**: The system shall import telescope images in FITS (Flexible Image Transport System) format.
- **FR-ALG-01: Source detection**
  - **FR-ALG-01.1**: The system shall integrate with the ASTRiDE (Automated Streak Detection for Astronomical Images) pipeline to automatically detect streaks in telescope images ([29]).
  - **FR-ALG-01.2**: The system should filter background noise by enabling users to configure a signal-to-noise ratio threshold to the pipeline output.
- **FR-CAT-01: Catalog validation**
  - **FR-CAT-01.1**: The system shall integrate with the Gaia DR3 catalog to identify known stars within the image field of view (FOV) ([57]).
- **FR-CALC-01: Attribute calculation**
  - **FR-CALC-01.1**: The system shall calculate identifying attributes for each potential debris candidate, including: right ascension (RA)[1], declination (DEC)[2], and apparent magnitude[3].
- **FR-EXP-01: Reporting**
  - **FR-EXP-01.1**: The system shall export a structured report (e.g., CSV/JSON) containing the list of validated debris candidates and their unique attributes.

5.2.1.2 Dashboard & user interaction functional requirements

Regarding the range of dashboard functions available to users (operators) on the front end of the application.

- **FR-VIS-01: Data visualization**
  - **FR-VIS-01.1**: The dashboard shall render the ingested telescope image(s) with interactive geometric overlays: orange squares/rectangles for potential debris candidates and yellow circles for identified Gaia stars.
  - **FR-VIS-01.2**: The dashboard should provide layer toggle controls, allowing the user to hide specific overlay types (e.g., "Hide Stars") to reduce visual clutter.
- **FR-INT-01: Interactive Analysis**

[1] Right ascension (RA) is the celestial equivalent of longitude, expressing an object's position east–west on the sky.
[2] Declination (DEC) is the celestial equivalent of latitude, expressing an object's position north–south on the sky.
[3] Apparent magnitude is a measure of how bright an object appears from Earth; lower values correspond to brighter objects.

  - **FR-INT-01.1**: Upon selecting a candidate from the list, the system must automatically center and zoom the view on the object (Region of Interest crop).
  - **FR-INT-01.2**: The system shall calculate, for the selected candidate, attributes such as right ascension (RA), declination (DEC), and apparent magnitude.

- **FR-SYS-01: System Feedback**
  - **FR-SYS-01.1**: The dashboard must display global status indicators (e.g., "Processing...", "Uploading...") in the header to ensure visibility of the system state during long-running operations.

5.2.2 Non-functional requirements

Non-functional requirements refer to the perceptible qualities of the system, such as performance, usability, and constraints. They are crucial in helping users accept the system, as they guarantee the platform's effectiveness in the context of use.

- **NFR-INT-01: Interactive Analysis**
  - **NFR-INT-01.1**: The system should, upon selecting a candidate from the results list, automatically center the view on the object and apply a zoom (region of Interest crop); according to data visibility patterns ([22]).
  - **NFR-INT-01.2**: The dashboard shall display a detailed information panel for the selected candidate, presenting the attributes calculated by the backend (RA, DEC, Magnitude).
  - **NFR-INT-01.3**: The dashboard should provide a low-latency interactive experience to support the smooth execution of actions.

- **NFR-USE-01: Usability**
  - **NFR-USE-01.1**: The system's graphical user interface (GUI) shall follow a dark mode design to enable users' eyes to adapt under low-light image analysis conditions.
  - **NFR-USE-01.2**: The system shall promote accessibility by adopting a color-blind friendly color scheme when selecting/distinguishing between different objects (debris vs. star).
  - **NFR-USE-01.3**: The system shall display a domain specific scientific vocabulary, for better expert learning and adaptation to the system capabilities.

- **NFR-INS-01: Installation**
  - **NFR-INS-01.1**: The system should be distributable as a containerized application (Docker), promoting an intuitive installation in common operating systems (e.g. Windows/Linux).

- **NFR-ARQ-01: Architecture**
  - **NFR-ARQ-01.1**: The system architecture shall separate between front-end and back-end features, enabling background source detection and catalog validation without visual interference.

5.3 Use cases

After the definition of functional and non-functional requirements, it is equally important to describe common use cases to the SpotDebris dashboard. These define interaction scenarios between the user (operator) and the system. The use cases defined in this section follow the guidelines provided by the **ISO/IEC/IEEE 29148:2018** standard ([27]), identifying actors, preconditions, post-conditions, and the main interaction flows to fulfill the system's goal.

5.3.1 UC-01: Automated star identification

Description: The user initiates automatic star cataloging of the uploaded image to distinguish between stars and potential debris.

- **Actors:** User/Operator (primary), Backend Pipeline (secondary).
- **Preconditions:** A single .fits image or multiple .fits images are loaded in the entry view page.
- **Post-conditions:** All identified stars are visually highlighted with yellow overlays, for instance.
- **Main Interactive Flow:**
  1. The Operator uploads the image(s).
  2. The Dashboard automatically starts processing the uploaded image(s) using external catalogs (ASTRiDE and Gaia DR3) and Python ([50]) libraries.
  3. The Dashboard may display a success status message: "Gaia pipeline catalog processing".
  4. The Backend extracts point sources using pre-configured parameters.
  5. The System receives the coordinate list.
  6. The Dashboard renders circular (yellow) overlays on identified sources (stars).
  7. The Dashboard may display a success status message: "Star match complete: X stars found".
  8. The System updates the main image analysis view-port with the identified stars.

5.3.2 UC-02: Debris candidate verification

Description: The user manually validates a potential space debris streak detected by the system.

- **Actors:** Operator.
- **Preconditions:** The ASTRiDE streak detection pipeline has executed, and candidates are listed in the sidebar.
- **Post-conditions:** The candidate is classified as "Confirmed debris" and recorded in the database.
- **Main Interactive Flow:**
  1. The Operator selects a candidate from the "Detections list".
  2. The Dashboard zooms into the region of interest (ROI).
  3. The Operator visually inspects the streak pattern (checking for pixel linearity).
  4. The Operator clicks the "Confirm Debris" button.
  5. The System updates the object style to a solid Red border and changes the list tag to "Confirmed".

5.3.3 UC-03: Export observation report

Description: Generating a report of verified detections for external catalogs.

- **Actors:** Data Scientist / Operator.
- **Main Interactive Flow:**
  1. The User navigates to the "History" view.
  2. The User may apply filters (e.g., Date Range: Last 24h, Status: Confirmed).
  3. The User clicks "Export Data".
  4. The System may prompt for format selection (CSV and JSON).
  5. The System generates the file including timestamps, RA/DEC coordinates, and classification.
  6. The Browser starts the file download.

5.4 Summary

With the elicitation of requirements now established, as this process may be subject to change depending on the implementation of mock-ups and subsequent code development, the essential functionalities of the prototype have been defined, as well as the optional qualities that can enhance the robustness of the system.

The next stages involve defining the most appropriate technologies for development, designing the system architecture, and creating the first visual mock-ups of the prototype, which will guide the implementation of the interface in a subsequent stage.

Chapter 6

Architecture and Technologies

This chapter details the architectural principles and technological choices that support the SpotDebris dashboard. In addition to a structured description of the system, the various components that integrate it, as well as the interaction between them, according to the previously defined requirements, are explicitly described in this chapter.

6.1 System architecture

6.1.1 Overview

The system was designed to be a web application, following a client-server approach. This solution was preferred over a desktop application due to its ease of deployment and cross-platform compatibility. As the computational load at the cataloging algorithm level is run by the server, the client-side dashboard is ultimately more lightweight and responsive, regardless of the user's hardware specifications.

In order to ensure consistency between the development and production environments, the developed SpotDebris dashboard app is containerized using Docker ([9]). The isolation of the environment and the grouping of all libraries necessary for proper functioning between frontend and backend resolve version conflicts and ensure software reproducibility.

Figure 6.1 illustrates the high-level architecture of the dashboard, adopting the logical view of the "4+1 View Model" by [30]. This perspective of dividing the system into working modules ensures a clear separation of concerns. The User Interface module is the starting point of the system, managing what the user sees and interacts with visually. This module communicates with the Debris Analysis Engine, which orchestrates the detection workflow and is the main processing unit, i.e. the application's computational core. This core module is supported by the Application State and Metadata module, which ensures consistency between requests (e.g., observation parameters, image coordinates). The dashboard also makes use of external pipelines, such as ASTRiDE ([29]) and Gaia DR3 ([57]), to perform streak detection and astrometric data validation. Ultimately, the processed results will be formatted within the Data Ingestion and Transformation module before being visually displayed to the user.

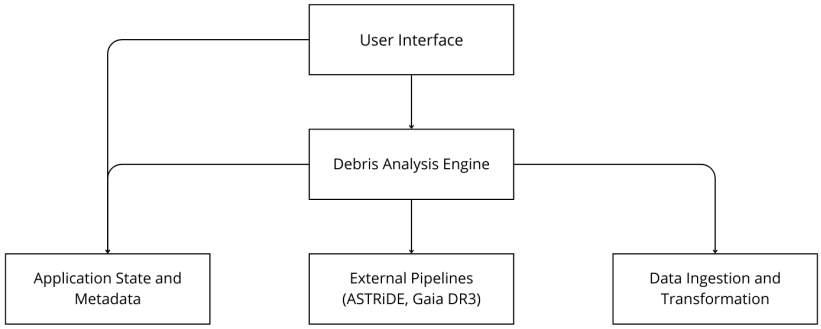


Figure 6.1: Logical View of the SpotDebris dashboard architecture

6.1.2 Main components

As seen in the previous diagram 6.1, the system architecture includes the diverse operating modules, to separate concerns. This diagram can be expanded to include the three primary architectural layers of the system, as seen in figure 6.2.

- The Presentation Layer (Frontend): The visual part of the interactive application (dashboard), which is what the user sees and interacts with, ensuring a consistent presentation of visual elements and data and communicating the visually hidden part of the system.
- The Application Layer (Backend): Serves as the core logic unit. It uses a REST API protocol (which means, an Application Programming Interface that follows the design principles of the Representational State Transfer architectural style) to communicate with the frontend and commands the execution of algorithms and external pipelines for space debris detection.
- The Infrastructure Layer (Containerization): Encapsulates the entire application in a Docker container to ensure that the Python environment (for processing) and the Node.js environment (for serving the UI) operate seamlessly together.

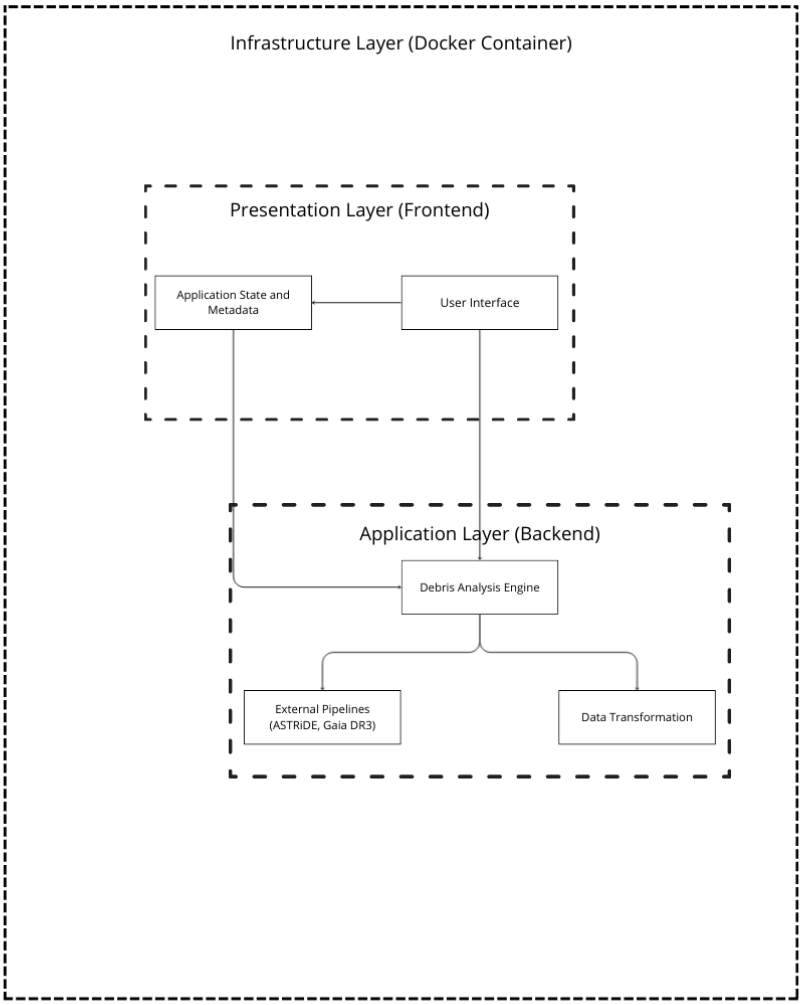


Figure 6.2: SpotDebris Architecture Layers

6.1.3 Communication flows

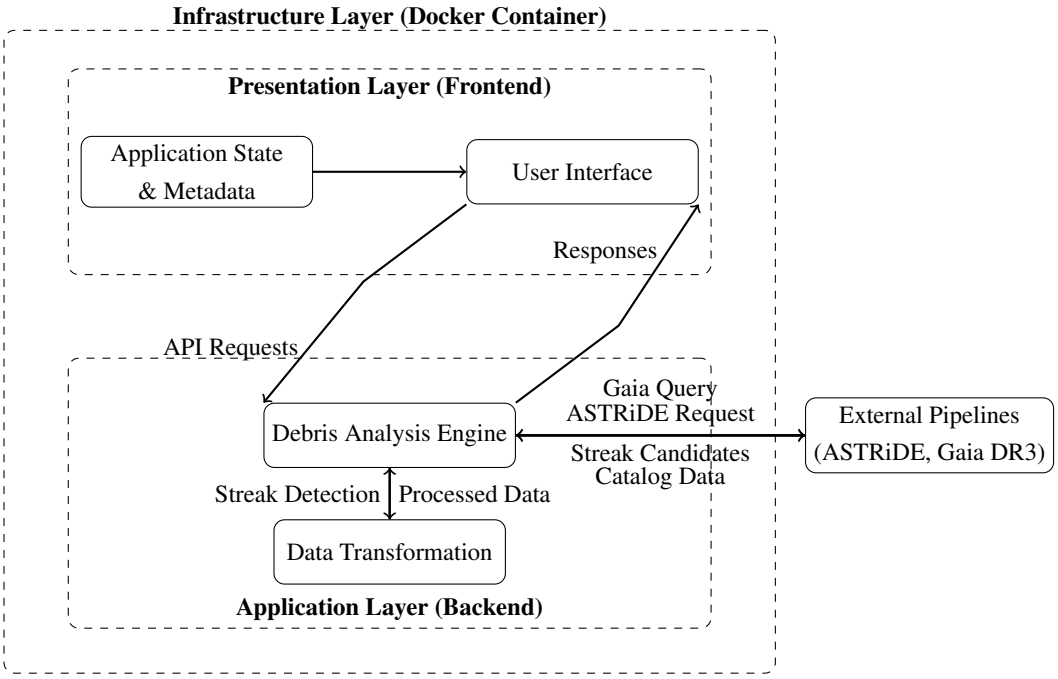


Figure 6.3: SpotDebris Architecture with Communication Flows

Figure 6.3 shows an updated version of the previous architectural diagram, Figure 6.2, now supplemented with communication flows between layers and interactions with external pipelines. A thorough understanding of the data flow within the system is crucial to assessing its responsiveness, given the computational load of the image processing algorithms involved. The system operates on a stateless request-response model over HTTP, incorporating asynchronous background processing for the respective algorithms.

The system's interaction flow can be interpreted as follows:

1. **Data ingestion:** The operator uploads a .fits file via the frontend interface. Once the file extension has been successfully validated, an HTTP POST request is sent to the backend.

2. **Asynchronous processing:** After receiving the file, the backend initiates the associated processing pipeline. In this situation, integration with ASTRiDE, asynchronously. A "Processing" message may be displayed to the user on the dashboard interface.

3. **Computation & cataloging:** The backend runs the streak detection algorithm and, at the same time or shortly thereafter, requests Gaia DR3 catalog data to identify stars in the image's field of view (FOV) using a world coordinate system.

4. **Payload Response:** A JSON response is generated after backend processing, containing:

    - The path to the normalized image in the view-port (converted from .fits to .jpg for web display).

- A list of detected candidates with their specific bounding boxes and attributes (right ascension, declination, etc.).
- A list of cataloged stars in the image FOV.

5. **Client-Side Rendering:** The React frontend updates the application state upon receiving the payload. It renders the image following common HTML practices and draws the geometric overlays (green squares for streaks, yellow circles for stars) based on each of the coordinates received.

6.1.4 The dashboard as a subsystem

The prototype, which is discussed in greater detail in the Chapter 7, is based on a web subsystem architecture divided into backend and frontend components, as well as local persistence, as discussed earlier in this chapter.

This approach is consistent with the Logical View of the 4+1 Model mentioned earlier in this chapter, in which the visual layer of the dashboard (frontend) interacts with the underlying, non-visible components (backend) through clearly defined responsibilities, data flows, and interfaces that, combined, support the overall behavior of the system.

At a superficial level, the pipeline can be characterized as follows:

- The user uploads one or more .fits files, which is or are validated and associated with a new run;
- The backend processes the file(s) asynchronously, extracting metadata, generating a preview, querying the Gaia catalog and running the ASTRiDE algorithm;
- Results and intermediate states are persisted to enable history tracking and recovery;
- The frontend queries the corresponding endpoints to update the interface, display errors, and enable manual classification and report export.

The dashboard makes use of the following external interfaces: .fits files, .json and .csv structures, and standard HTTP/REST endpoints. The Interface Control Document (ICD), which defines how different software systems, hardware components, or sub-systems communicate with each other.) describes these interfaces. In detail explanation of how these interfaces interact with each other is left for the next Chapter, 7.

6.1.5 Data Architecture

The data architecture worked on in the backend can be understood as follows:

- Each file upload is stored in `data/uploads` with a `run_id` prefix, and the uploaded files are retained for later metadata extraction.

- The object persisted in `index.json` stores `status`, `started_at`, `finished_at`, `error`, and `result`. The `result` field aggregates the data produced by the pipeline.
- A .png preview image is generated in `data/previews` by the backend, and the frontend uses this file to render the dashboard viewport.
- The .json or .csv exported via the history tab includes information such as `shape`, `has_wcs`, `center_ra_deg`, `center_dec_deg`, `fov_width_deg`, `fov_height_deg`, WCS warnings, and sensor/telescope identification when available.
- The query to Gaia DR3 extracts a set of stars with `source_id`, coordinates `ra_deg`/`dec_deg`, G magnitude, and pixel position, which the frontend uses to draw the yellow overlays on the rendered image. The pipeline ends with the labeling of potential candidates `candidate_id`, `bbox_px`, `classification`, and associated parameters, cataloged by ASTRiDE, for subsequent manual analysis of the rectangular orange overlays.

This data organization also ensures that intermediate pipeline artifacts are stored in human-readable formats (such as .json and .png), making it easier to troubleshoot issues during development.

In this way, it is clarified what the system stores and returns. It is a data structure designed for the workflow of uploading, processing, catalog queries, and visualization.

6.1.6 Interfaces and ICD contracts

As mentioned above, ICD establishes contracts between subsystems [12]. The interfaces that have been implemented in the prototype are:

- Frontend <-> Backend: HTTP/REST requests in .json format via `/api/v1/uploads/fits`, `/api/v1/runs`, `/api/v1/runs/{run_id}`, `/api/v1/runs/{run_id}/result` and `/api/v1/runs/{run_id}/report`.
- Backend <-> Preview: preview generation in .png and pixel-to-celestial coordinate conversion in `/api/v1/runs/{run_id}/preview` and `/api/v1/runs/{run_id}/cursor-coordinates`.
- Backend <-> Gaia DR3: query the catalog via `astroquery.gaia`, on endpoint `/api/v1/runs/{run_id}/gaia-stars`.
- Backend <-> human annotation: candidate status update via `/api/v1/runs/{run_id}/candidates/{candidate_id}` with states `pending`, `confirmed` and `rejected`.

The ICD specifies, among other elements, the mandatory fields for each interface: run identifiers and metadata (e.g., run_id, started_at, finished_at); the format and units of the scientific astrometric parameters: RA/DEC in decimal degrees, magnitudes in Gaia G system, timestamps

in ISO8601 UTC [26], the .fits file format, the minimum fields for candidates and stars, and examples of error and success payloads.

A minimal example, for illustration purposes, of an error payload in .json format could be:

"code": 400, "message": "Bad Request", "details": "missing field run_id".

In this dissertation, the ICD is not defined as a standalone document. Instead, its logical structure is explicit through the definition of the interfaces implemented in the prototype, as detailed above. These interface specifications are the practical representation of the ICD, allowing verification of required fields, units, formats, and payload conventions. The prototype corresponds to a minimum viable product, with interfaces that may continue to evolve. According to the [12] standards, a formal definition of a ICD is more appropriate when interfaces are stable and integrated into an environment with multiple subsystems. Therefore, the interfaces were iteratively adjusted during the design process, and no external subsystem was dependent on rigid contracts. In this way, it was decided to provide only a logical description of the ICD to ensure clarity and traceability, reserving the development of a formal ICD for future iteration phases.

6.1.7 Design Science Research and Standards Compliance

The dashboard was developed with the mindset that it is an instrument that solves a problem, and its usefulness and usability were evaluated retrospectively using Design Science Research principles ([24, 44]). The problem definition, prototype implementation, and proper demonstration of the upload/processing/visualization workflow were conducted, along with an evaluation of the system's behavior using real data and perceived usability. More precisely, the problem was identified as the lack of an interactive, image-focused dashboard capable of supporting SST operations for inspecting telescope images, such as the analysis of space debris streaks and information on cataloged stars. With this problem in mind, the methodology for addressing this challenge followed these main steps: setting up a workflow for uploading .fits images, running streak detection (ASTRiDE), integrating the star identification catalog (Gaia DR3), and, finally, visualizing the candidates and identified stars in an interactive interface. According to the Design Science Research process, the implemented dashboard was demonstrated through the execution of the full upload–processing–visualization workflow. Telescope .fits images were used and evaluated through functional validation and a usability assessment. These evaluations confirmed that the prototype effectively addresses the identified problem.

Furthermore, traceability, interface management, and a clear description of the system's contracts are supported by the recommendations of the [12] standards. The reference to the ICD comes naturally in this context as the document that brings together the contracts between subsystems, without the need to explore it in greater depth beyond what was explained in the previous subsection.

6.2 System technologies

This section discusses the choice of technologies for the development of SpotDebris dashboard, considering its scientific and technological scope, in order to deliver a good and dynamic user experience.

6.2.1 Selection criteria

The choice of technology stack to be adopted was mainly determined by the requirements elicitation process described previously.

1. The backend must integrate with pipelines properly connected to the scope of the space, namely ASTRiDE and the GAIA DR3 catalog, as previously mentioned, which are primarily written in Python.

2. The dashboard must handle interaction capabilities without constant page refresh, for instance, toggling visibility of star layers, selecting individual debris candidates, and adjusting visualization thresholds.

3. The need to build a solid system capable of handling backend computing tasks while delivering a seamless user experience, with an eye toward potential future scalability of features.

4. The capacity to consistently replicate the scientific environment on various devices.

At the start of the system architecture design phase, consideration was given to using certain rapid prototyping frameworks for Python developers ([50]), such as Streamlit ([25]) or Plotly Dash ([47]). However, since these frameworks are more geared toward server centralization, and since the model adopted in this dashboard requires that all interactions (e.g., zooming and filtering) provide instant feedback to the user (which, in this case, would involve a client-server roundtrip), these frameworks were discarded. In a telescope image processing and analysis dashboard, this type of constant round-trip to the server-centralized model produces considerable latency, making the system a constant source of friction during use and degrading its overall performance. Consequently, a fully decoupled architecture using a dedicated JavaScript frontend was chosen to ensure a fluid and smooth user experience with the dashboard.

6.2.2 Backend technologies

Given the reliance on the algorithms mentioned earlier for streak detection and star cataloging, Python is undoubtedly the language of election for the system's backend environment.

FastAPI ([15]) was chosen for the Web API framework, in contrast to other traditional alternatives such as Django ([8]) or Flask ([43]).

The reasons behind this decision are as follows:

- FastAPI ([15]) uses an ASGI (Asynchronous Server Gateway Interface) standard, which means that it makes it possible to use concurrent requests efficiently. This is important when multiple .fits files are being uploaded and processed simultaneously, ensuring that the server remains responsive.
- The framework also makes use of Pydantic ([49]) for data validation. Despite the simplicity of the communication interface, the more complex space related data requires that invalid data must be rejected early in the pipeline.
- FastAPI is more efficient performance wise, compared to NodeJS ([42]) and Go ([20]), outperforming Flask ([43]) in input or output operations.
- FastAPI is also served using Uvicorn ([11]), a lean ASGI server that allows effective handling of asynchronous requests. This ensures responsiveness when multiple .fits files are uploaded and processed simultaneously.

6.2.3 Frontend technologies

React ([33]) is a Javascript library and was selected for its virtual DOM (Document Object Model), where an ideal, or "virtual" representation of a user interface is kept in memory and synced with the "real" DOM capabilities.

Other reasons for the choice of React for the frontend include:

- React allows the logical separation previously defined in the requirements, in the way that it enables the definition of isolated, reusable UI components/elements.
- React allows common interactions in the dashboard, for example, filtering and toggling the view of cataloged stars instantly in the browser memory, which provides feedback to the user without refreshing the page, unlike the rejected server-side rendering solutions mentioned earlier.

Vite ([60]) was chosen as the frontend build tool. It is a way to provide a robust modern web development environment with Hot Module Replacement (HMR). This enables fast iteration cycles. Visualization of the processed .fits images and the drawing of overlay boxes are done using HTML5 Canvas/SVG (Scalable Vector Graphics) manipulation, handled by the React state.

6.2.4 Support tools

In order to ensure a more agile software development lifecycle and greater robustness in the delivered prototype, the following tools were included:

- **Docker:** Container used to encapsulate the dependencies of both the Python backend, Node.js frontend and other libraries. This helps maintain the system's reproducibility, guaranteeing that the pipeline runs seamlessly across multiple machines ([9]).

- **GitHub:** Used for source code management, facilitating the tracking of changes ([18]).
- **Figma:** Used to design high-fidelity mockups, creating a purely visual representation of the various dashboard views, thus assisting with future code implementation and page design ([16]).

6.2.5 Dashboard mockups

As mentioned in the previous subsection, the Figma design tool was utilized to create visual prototypes of the main views of the SpotDebris dashboard. [16]

Although, in the state of the art of this dissertation, the logic of multiple-views in a data visualization dashboard was studied and emphasized, at this time, for experimental purposes and prior to the actual code development, a single view is adopted for each of the dashboard's interactive pages.

Nevertheless, the design of the mockups was guided by the visualization principles established in the state of the art (Chapter 3).

Below are the mockups designed for this iteration phase (Figures 6.4 to 6.7).

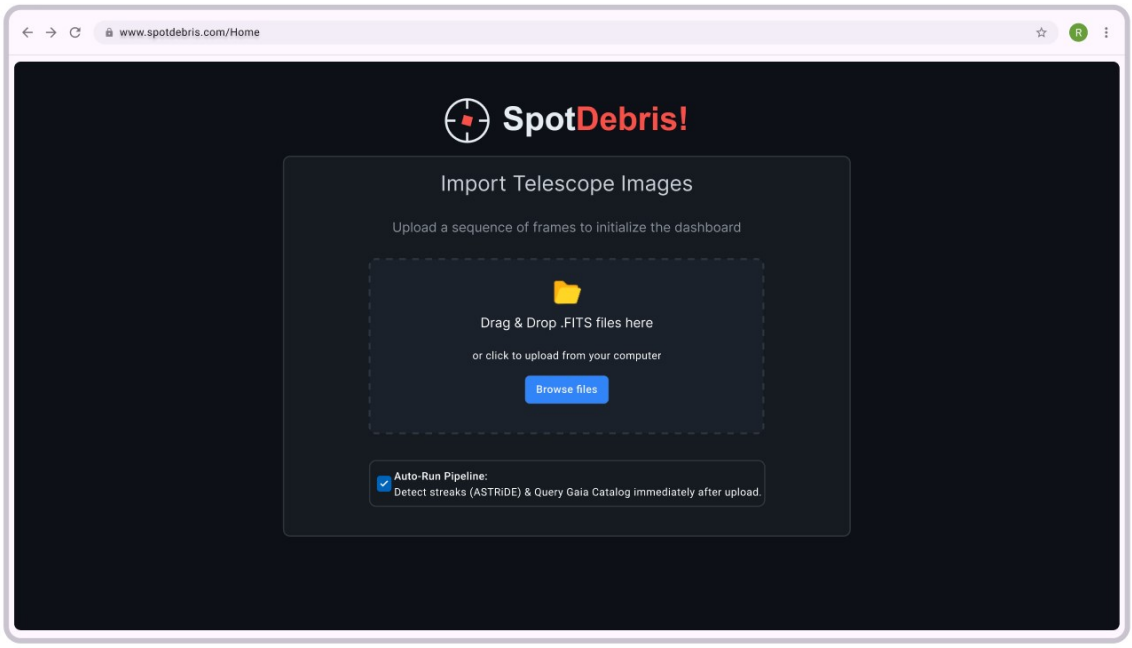


Figure 6.4: SpotDebris entry view, with the drag and drop or browse files features to upload .fits images and a check button to auto-run pipeline

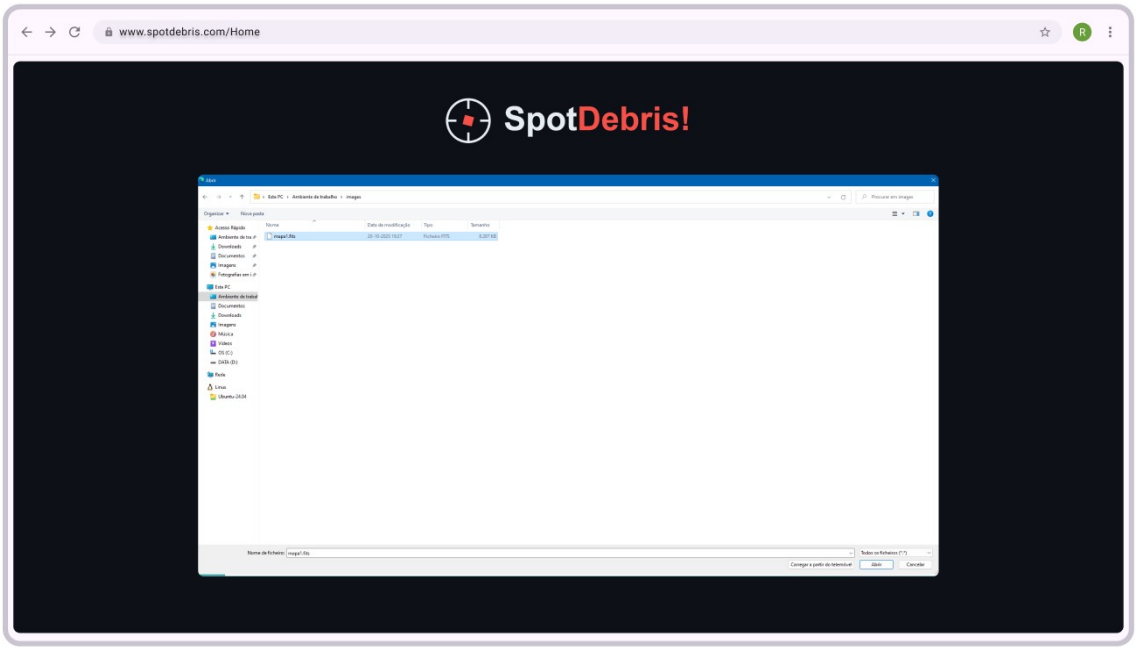


Figure 6.5: The upload partial view, with a .fits file selected

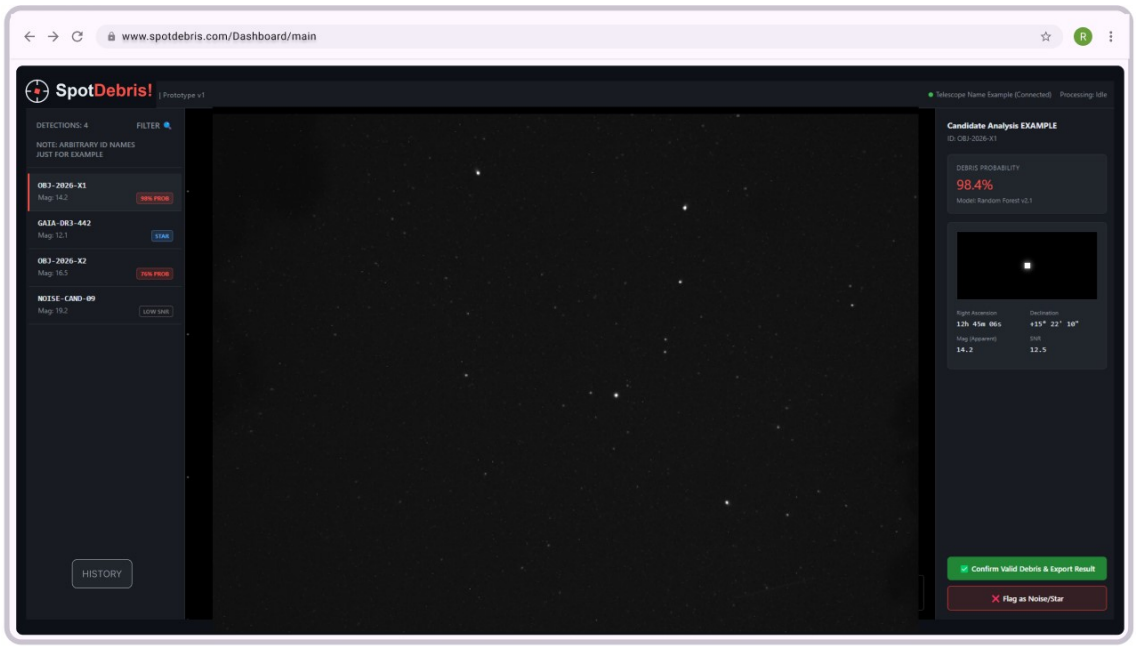


Figure 6.6: SpotDebris main view - the image analysis interface

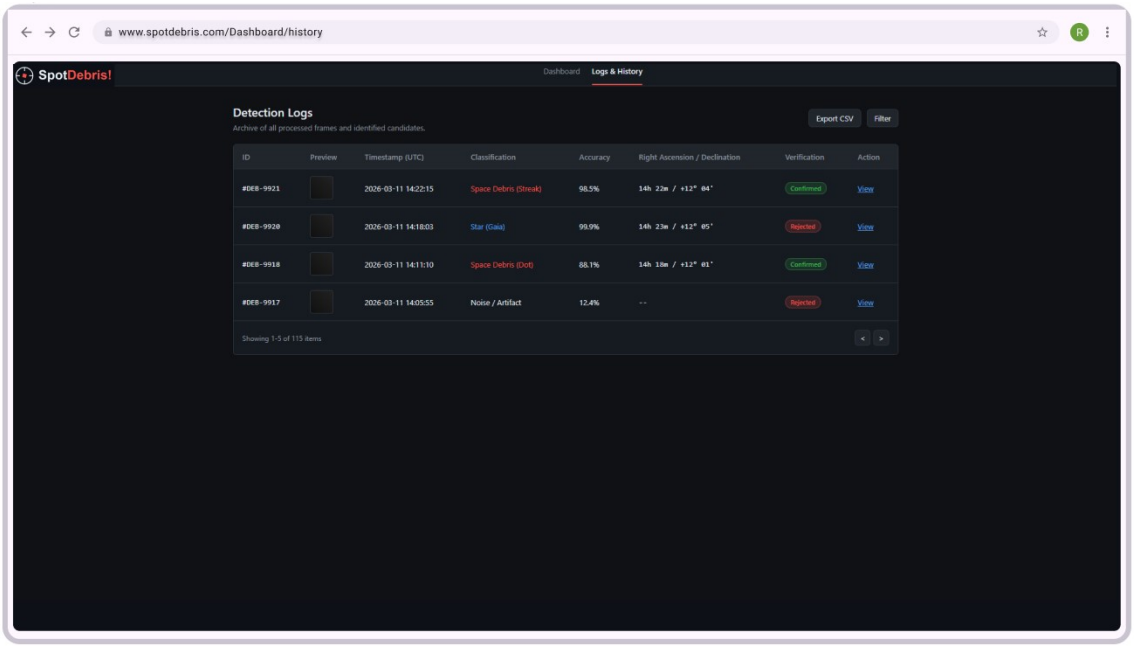


Figure 6.7: The history view, featuring some arbitrary objects and their confirmed or rejected debris status. From here it is possible to export a report

6.3 Summary

Now that the decision to build a web application with client-server communication, containerized in Docker, has been finalized; the system architecture and the relevant technologies have also been established, paving the way for the actual implementation in the next chapter.

Chapter 7

Prototype Implementation

This chapter discusses the implementation of the SpotDebris interactive dashboard prototype, which is the Minimum Viable Product (MVP) of this dissertation, as well as the decisions made and the paths followed to bring this application to reality.

The implementation of this prototype was done iteratively, with the product being refined through progressive adjustments. These refinements were based on the need to prioritize low latency and operational performance in a potential real-world scenario, technical validation of external pipelines, and the scientific accuracy of the data presented.

7.1 Implementation strategy

The strategy adopted to develop the Minimum Viable Product was based on translating requirements into short cycles of iteration and validation. Four main qualities were considered during the iterative process:

1. Ability to load and process .fits files.
2. Availability of results relevant to the user.
3. Human-Computer Interaction.
4. Export and analysis of results.

This set of four primary qualities aligns with the state of the art, particularly with regard to dashboard engineering, the analytical process combined with decision support, status transparency, feedback, and traceability (Chapter 3). In addition, the objectives defined for the implementation in Chapter 4 were followed and the use cases for this prototype described in Chapter 5 were also respected.

7.2 Backend implementation

7.2.1 Run process and data model

Each process is assigned a unique identifier. This includes life-cycle timestamps and a status, which can be one of the following: queued, running, completed, or failed. Since the developed prototype is a MVP, the decision was made not to use a relational database. Instead, lightweight data persistence on disk was opted, using a JSON index file and dedicated directories for uploads, previews, and results. The decision to proceed in this regard was based on the following reasons, given that this is a prototype: Simplify actions and reduce infrastructure complexity; enable analysis of objects generated during testing; and maintain the dissertation's focus: processing data related to telescope images and visual interaction.

The developed solution is designed for fast and precise interactions and is intended for use by a single operator; it can be scaled in the future to a more transactional backend system, where multiple operators can interact simultaneously.

7.2.2 Pipeline processing

Once the .fits file(s) upload has been validated, the pipeline is scheduled to execute in the background. It consists of four main blocks:

7.2.2.1 Metadata extraction and World Coordinate System

Accurate astrometric mapping and interface population require precise metadata extraction. Extracting such metadata from .fits files includes image shape, (availability of) celestial coordinates, approximate field center, angular field-of-view measurement, and metadata consistency alerts. The libraries `astropy.io.fits` for .fits file parsing and `astropy.wcs` for World Coordinate System calculations are utilized in this pipeline stage.

7.2.2.2 Web preview

Since a .fits file cannot be rendered natively in a standard browser, the uploaded image(s) are converted to .png, an image format commonly rendered by browsers, using parameters such as zscale normalization, optional stretch, and colormap. The workflow used for this pipeline includes `numpy` for pixel-level array manipulation, `astropy.visualization.ZScaleInterval` for the empirical calculation of the display range, similar to the DS9 software ([56]), which is an application frequently used for the visual interpretation of .fits files, and `PIL (Pillow)` for converting these files into the .png format ([46]). The process includes normalizing pixel values using the zscale method and, when necessary, applying additional techniques such as histogram equalization or gamma correction. The result is then exported as a grayscale or RGB image, limited to 8 bits. This option allows for immediate visualization in the frontend without transferring any part of the scientific logic to the client, while preserving the representation conventions commonly used in space.

7.2.2.3 Gaia DR3 Query

An ADQL (Astronomical Data Query Language) query is executed to check whether valid World Coordinate System data exist in the loaded .fits file(s), in order to retrieve stars in the field of view, which are defined by yellow circular overlays, taking into account explicit magnitude and row count limits to ensure dashboard performance. This task includes celestial coordinates, pixel coordinates, and warnings if the results were truncated by the query boundaries. The cone-search radius is determined using the diagonal of the .fits WCS field of view to ensure that the query covers the entire observed region. Magnitude limits are defined using environment variables, which gives operators the flexibility to adjust the stellar overlay density in order to balance visual clarity and system performance.

7.2.2.4 Streak detection with ASTRiDE

Detection candidates (space debris streaks) are initially outlined using orange geometric overlays. As a post-processing step, collinear detections are merged to prevent the same object from being fragmented into multiple segments (in the case of the slightest discontinuity in a streak). This results in more coherent outputs that are easier to validate manually.

Each execution returns a structured set of outputs (.fits data, Gaia matches and ASTRiDE results), providing full clarity on what was successfully processed, what failed and where the method's limitations lie.

7.2.3 API and validation operations

The backend exposes a set of REST API endpoints that define the system's interfaces. These endpoints are grouped by functional domain as follows:

- **Run management**
  (`POST /api/v1/uploads/fits`, `GET /api/v1/runs`,
  `GET /api/v1/runs/{run_id}`, `GET /api/v1/runs/{run_id}/result`)
- **Image preview and visualization**
  (`GET /api/v1/runs/{run_id}/preview`)
- **Pixel-to-sky coordinate conversion (WCS)**
  (`GET /api/v1/runs/{run_id}/cursor-coordinates`)
- **Candidate classification**
  (`PATCH /api/v1/runs/{run_id}/candidates/{candidate_id}`)
- **Report generation**
  (`GET /api/v1/runs/{run_id}/report`)
- **Gaia viewport query endpoint prepared for future interactive exploration**

This layout confirms the "human-in-the-loop" concept defined in the state of the art and use cases: the algorithm makes a proposal, the operator validates it, and the system records it.

7.3 Frontend implementation

7.3.1 Entry page and processing

The homepage allows the user to upload one or more .fits files from their computer; the backend then begins the processing, and the user is redirected to the main view, that is, the core of the application. ASTRiDE and Gaia DR3 pipelines are automatically triggered. Error handling provides feedback in the event of unsupported file formats or network issues.

7.3.2 Main view

According to the state of the art multiple-views concept (Chapter 3), the main view is organized into three functional tabs: Overview, Candidates, and Summary.

The core workflow of the front-end interface is as follows:

1. (Backend) conversion of .fits files to .png, rendered as .svg overlays for interactive image tuning and astrometric overlays.
2. Debris candidates are overlayed as bounding boxes with distinct colors (pending, confirmed, rejected).
3. Gaia star overlays are colored as circles, with magnitude-based sorting.
4. Navigation tools within the field of view, enabling zoom via mouse wheel and panning through middle-button drag.
5. Action controls for candidate selection and subsequent classification.
6. Cursor telemetry with near-real-time WCS transformation.
7. Visualization parameters including stretch adjustment, colormap selection, and gamma tuning.

For the run state, a polling model was maintained in order to reduce complexity in the minimum viable product compared to event streaming. It was also decided at this stage to apply debouncing to WCS coordinate requests triggered by the cursor, thereby preventing successive calls caused by mouse movement, which could saturate the system and hinder responsiveness. Debounce delay was set to 180 ms, balancing responsiveness with backend operations.

The view-port preview features the converted image as well as interactive overlays (with the possibility of switching between the various uploaded and converted images). This separation mechanism helps distinguish between the various visual layers (image, stars, debris), providing a more interactive, dynamic, and fluid user experience.

7.3.3 History and export

The dashboard also includes a history page where users can view runs, space debris candidates, classification, accuracy (provided by ASTRiDE), RA/DEC, status, and the action taken on that candidate.

Users can export a report through the main view, which supports exporting results in .json or .csv formats. This report contains all the data from the uploaded and analyzed files (detected space debris streaks, etc.) , facilitating subsequent integration with external analytics pipelines.

7.4 Technical decisions, trade-offs, and challenges

The implementation of the minimum viable product involved critical decisions about future scalability and the delivery of a scientifically viable prototype.

7.4.1 Relational database vs disk persistence

In the context of a MVP, disk persistence is reasonable, given that the typical use case is a single operator per run, with asynchronous processing designed for queue-based processing. However, despite its simplicity and local traceability, in the context of a larger-scale system, it would be preferable to adopt a relational database, removing the constraints of limited transactions while promoting concurrency control and multi-user usage.

7.4.2 Polling vs. real-time communication

Polling was sufficient for this run-based pipeline and turned out to be a simpler implementation rather than more complex WebSockets or Server-Sent Events. Nevertheless, a streaming approach could provide more detailed progress updates and be a natural evolution for future multi-step pipelines.

7.4.3 Visual processing: server vs. client

Generating previews on the server has an advantage: all the logic related to astrometric data is centralized in a single area, which also means that the image appears consistently in any browser. The downside is the extra load on the backend. However, if this processing took place on the client side, the browser would have to handle greater complexity and cross-platform variations, making debugging much more challenging.

7.4.4 Gaia overlay density

Restricting the number of star overlays and applying magnitude filters improves image readability and enhances rendering performance. Such restrictions, however, may obscure some of the stellar density in denser fields. The current approach prioritizes usability (which also depends on the

system's performance) over complete representation, in line with the dashboard design principles discussed in the state of the art (Chapter 3, section on dashboard design patterns).

7.4.5 ASTRiDE post-processing

Merging collinear segments helps prevent fragmented detections, although this still depends on proper adjustment of the backend parameters, which vary depending on the conditions of the instrument. In the MVP, these values are not available to the user; they were defined based on practical tests, and more thorough validation for different types of instruments is reserved for future work.

7.5 MVP and the requirements

The prototype meets the requirements and use cases set in the Chapter 5, for instance:

7.5.1 Functional domain

- Import and process .fits files (FR-IMG-01).
- Cross-reference with external catalogs to identify potential debris and stars in the field of view (FR-CAT-01).
- Visualize with overlays and interactive tools (FR-INT-01).
- Manual candidate validation with state persistence (UC-02, FR-SYS-01).
- Result export in reporting formats (UC-03).

7.5.2 Non-functional domain

- Display a detailed information panel with the attributes for the selected candidate (NFR-INT-01).
- Learnability and usability for domain experts (NFR-USE-01.3).
- Consistent and transparent overlay mappings between pixel and celestial coordinates (NFR-INT-01.2).
- Accessibility through a color-blind-friendly color scheme for object distinction (NFR-USE-01.2).
- Separated front-end and back-end architecture allowing background processing without visual interference (NFR-ARQ-01.1).

A detailed evaluation of usability and performance metrics will be conducted in the next chapter, using measures such as the System Usability Scale, discussed in the state of the art, Chapter 3.

7.6 Illustrations

Naturally, as it was expected, the final design ended up slightly different from the mockups presented in Chapter 6, subsection Dashboard mockups.

For clarity and ease of reading, all illustrations related to the developed dashboard are grouped in this subsection. This helps avoid breaks in the middle of the text, maintaining the flow of the content, while also providing a complete visual overview of the implemented prototype. Below are images of the SpotDebris dashboard, the minimum viable product of this dissertation (Figures 7.1 to 7.7).

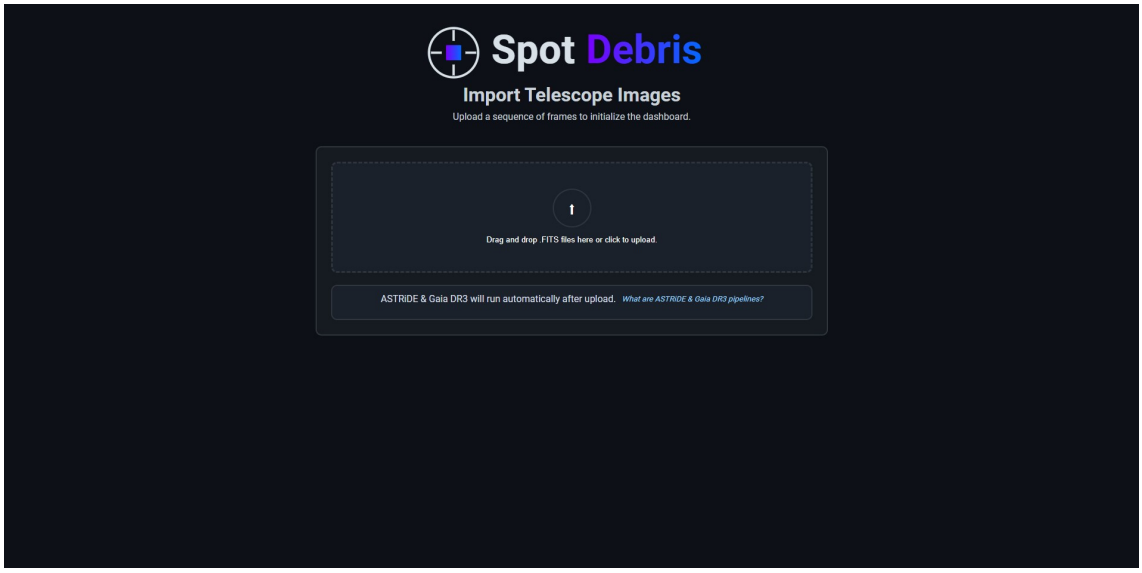


Figure 7.1: SpotDebris entry view, with the drag and drop or browse files features to upload .fits images and a question which, if hovered up, shows a small description of what the external pipelines consist of

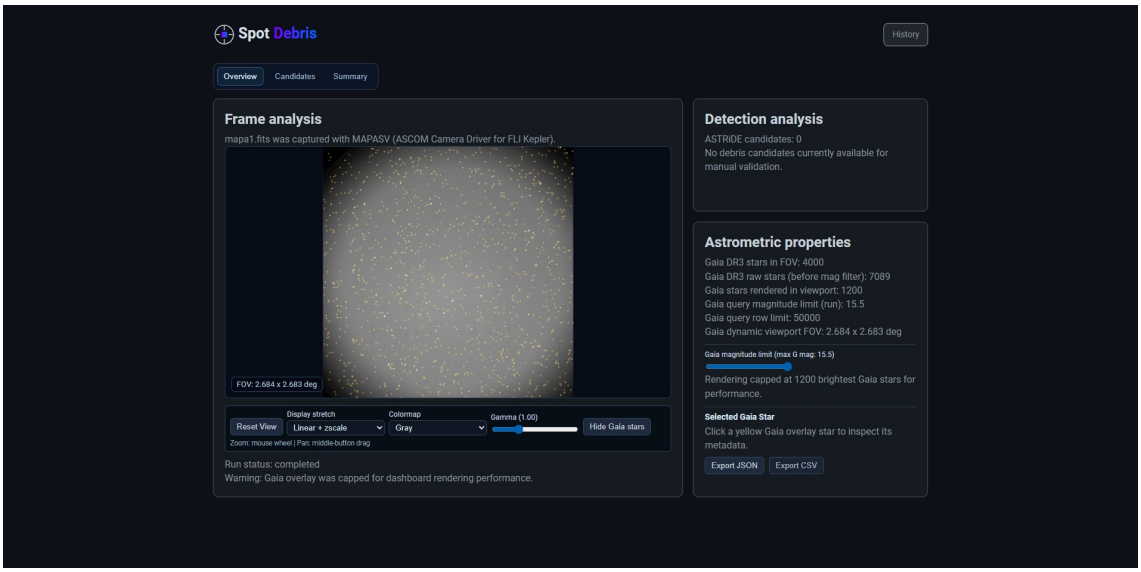


Figure 7.2: The main view displaying the view-port of an uploaded, processed image, with no ASTRiDE streaks detected, but countless Gaia stars cataloged, featuring diverse astrometric properties and interactive tools

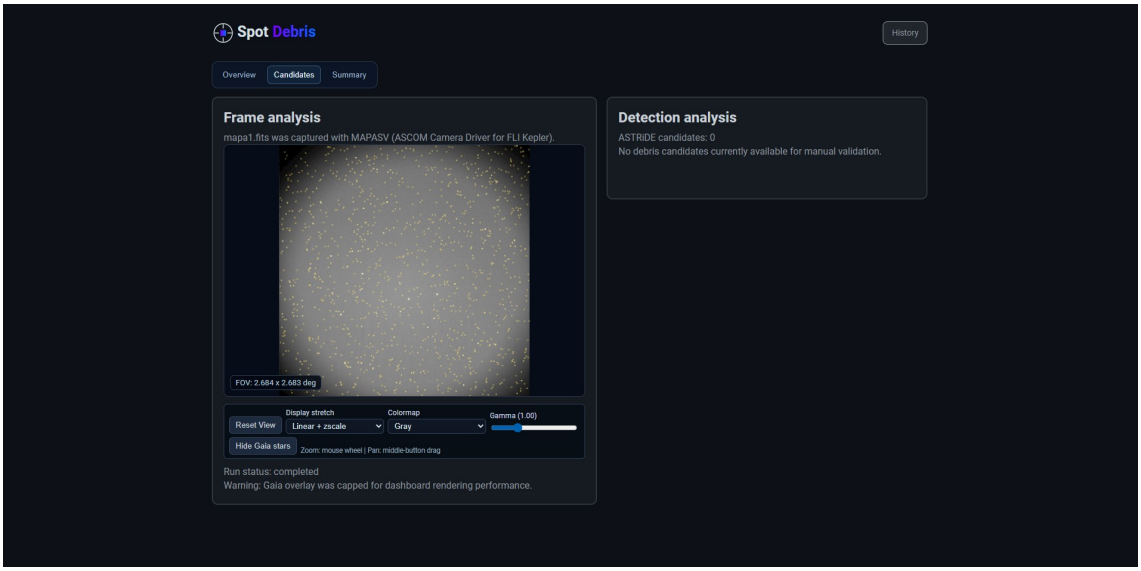


Figure 7.3: The candidates partial view, featuring the detected and manually validated candidates; in this specific image, there are no detections available

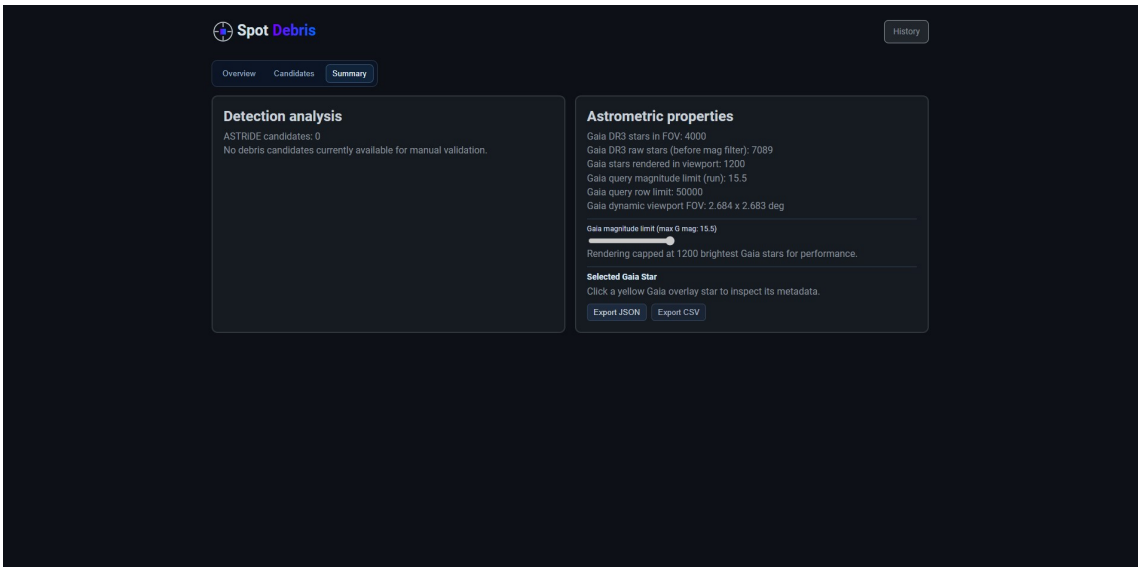


Figure 7.4: The summary partial view, displaying the summarized version of the analysis

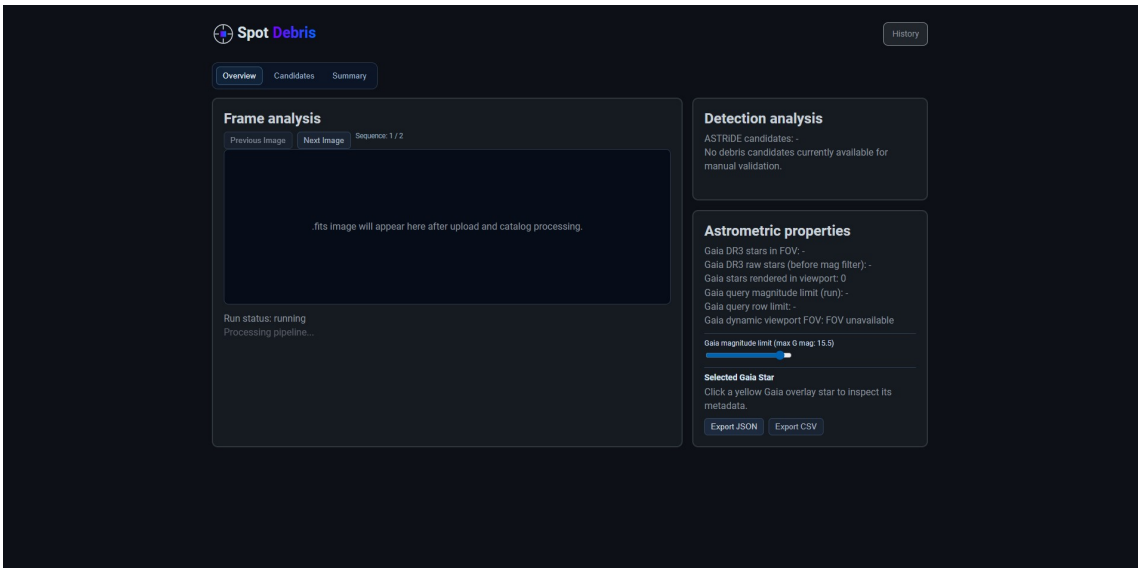


Figure 7.5: The processing, informative messages, telling the user that the pipeline is running, displaying the images (results) in a brief moment later

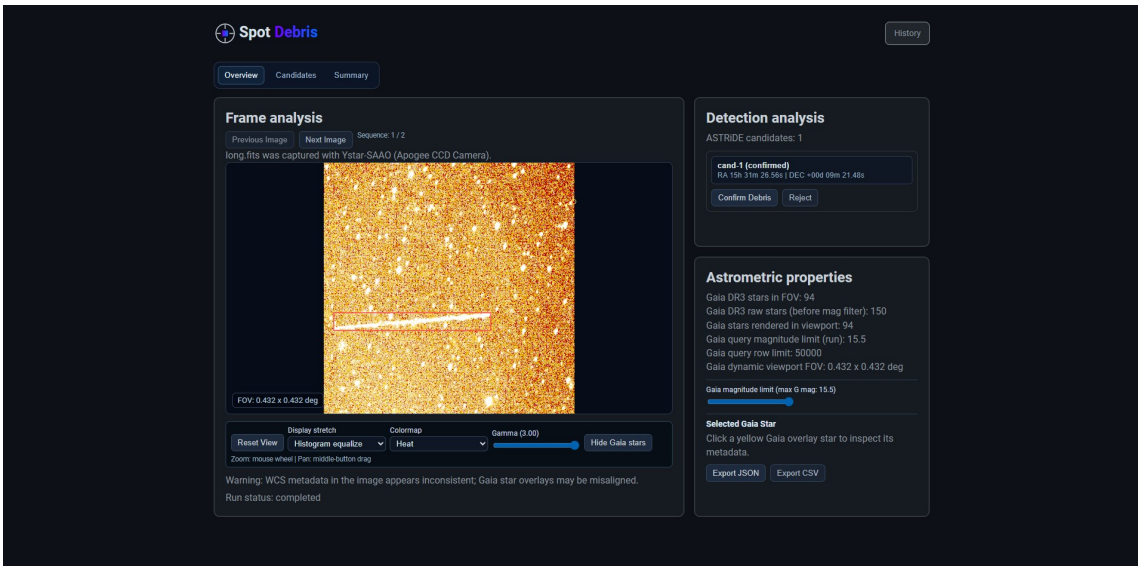


Figure 7.6: Another processed image, in this case, two images uploaded and processed, and the selected one in the view-port contains a potential space debris streak, which was manually confirmed, as well as modified visual parameters by the user

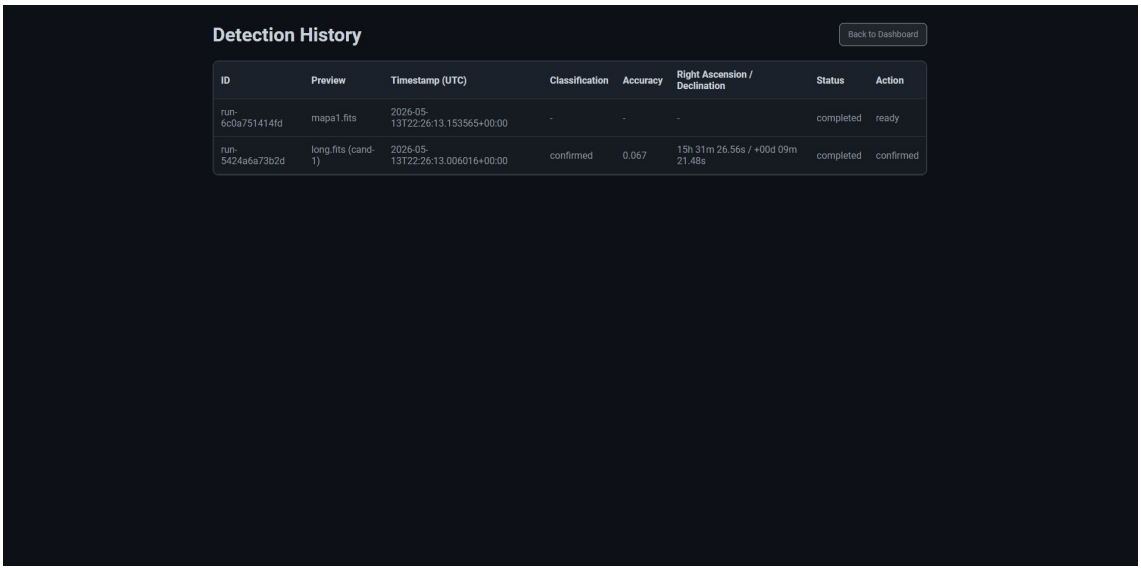


Figure 7.7: The history page, featuring two images uploaded and analyzed by the user, one of them with a potential space debris streak confirmed by the operator, and another image with no visible streaks

7.7 Summary

In conclusion to this chapter, the methodologies for designing the minimum viable product for this dissertation, the SpotDebris dashboard, were thoroughly described, thus bringing this journey to a near completion.

The result is a functional prototype that combines the ingestion and processing of .fits files, the integration of external pipelines, and an interactive, dynamic, and visual environment capable of providing a human user with a comprehensive user experience. It also provides support for decision-making, the export of results, and the maintenance of consistency as defined in the requirements, as well as the engineering of dashboards for Space Surveillance and Tracking.

The next step is to systematically evaluate the system regarding the usability, performance, and operational utility objectives defined in this dissertation.

Chapter 8

Evaluation

Subsequent to the implementation of the minimum viable product: the SpotDebris prototype discussed in Chapter 7, a comprehensive evaluation of the developed system was conducted from both a technical and user test perspective, in order to assess the system's ability to meet the requirements defined in Chapter 5 and also how user-friendly is the prototype, according to other metrics later discussed in this chapter. The evaluation protocol follows the drafted evaluation schematic in Chapter 4.

This chapter presents a systematic review of the prototype's functionality, performance, and usability in relation to Space Surveillance and Tracking image analysis.

8.1 Objectives

The fundamental objectives behind the evaluation of the developed prototype are the following:

1. Validate whether the implemented prototype meets the functional and non-functional requirements defined in Chapter 5, presenting evidence and demonstrating traceability.

2. Evaluate the heuristic review and performance of core tasks such as the ingestion and processing of .fits files, GAIA DR3 query latency, ASTRiDE detection accuracy, and .png preview rendering.

3. Assess how effectively SST experts and non-expert users can use the dashboard to complete debris identification tasks or interpret visual overlays. The metrics include workload, usability, acceptance, user experience, and textual comments ([4, 7, 23, 53, 59]).

8.2 Evaluation methodology

8.2.1 Testing environment

The evaluation tests occurred in a controlled environment with the following specifications:

- **Hardware:** 16 GB RAM, Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, NVIDIA GeForce GTX 1060.
- **Operating System:** Windows 11 with Docker Desktop installed for containerization.
- **Browser:** Microsoft Edge version 148.0.3967.70 for frontend testing.
- **Backend:** FastAPI served via Uvicorn in development mode on localhost:8000.
- **Frontend:** React 18.3 development build with Vite 5.4 on internal port localhost:5173. The user accesses through the localhost:5600 port. These ports are customizable in the ".env" file on the prototype codebase.
- **Data sources:** Real .fits files from the Pico do Areeiro Optical Observatory, with World Coordinate System (WCS) metadata; live queries done to Gaia DR3 through Astroquery (a Python library that provides access to astronomical catalogs and online data services).

8.2.2 Heuristic review

As discussed in the state of the art, Chapter 3, usability heuristics for visual, interactive, and informational dashboards ([10]), as well as collaborative analysis systems ([55]), were applied to a heuristic review conducted by two participants (abridged updated Nielsen's 10 Heuristics):

- **Visibility of system status:** Does the dashboard display pipeline status (e.g. upload pending, processing, and completed)?
- **Match between system and the real world:** Does the terminology (Gaia candidates, debris detection, accuracy score) align with the SST expert knowledge language?
- **User control and freedom:** Can users revert confirmed or rejected detections, re-classify candidates, and modify overlay settings?
- **Error prevention and recovery:** Are invalid files detected? Can users fix upload errors?
- **Recognition rather than recall:** Are the overlays visually distinct? Is the pixel-to-celestial coordinate mapping easy to understand?
- **Efficiency:** Can users navigate between tabs, apply filters, and export results?
- **Domain coherence:** Does the dashboard align with SST workflows, including Gaia overlay hiding, debris annotation, and report export?

8.2.2.1 Heuristic review evaluation

Performed by the author, the heuristic review is described in the table 8.1, which lists the concrete behaviors observed in the prototype during the review.

Heuristic	**Evaluation**	**Observed evidence**
Visibility of system status	Verified	The dashboard displays the various status to the user as the preview is being generated: e.g., uploading, processing, complete.
Match between system and the real world	Verified	The dashboard uses domain-specific terminology (e.g., debris candidates, stars, Gaia overlays, WCS coordinates, magnitude).
User control and freedom	Verified	The dashboard allows users to classify candidates, switch between tabs, adjust image filters, view information on cataloged stars, and more.
Error prevention and recovery	Partially verified	The dashboard identifies when a .fits file is invalid or unsupported and prevents the process from continuing without a warning; however, guidance could be added on how to resume browsing after an error or even return to the upload homepage without having to manually go back to the previous URL, in case the user selects the wrong files to upload.
Recognition rather than recall	Verified	The main actions and results remain visible on the screen. As a result, candidate overlays, labels, and tab navigation reduce the memory load.
Efficiency of use	Verified	Actions such as uploading, analyzing detected streaks, and exporting results are available directly through user-friendly controls.
Domain coherence	Verified	The workflow follows the Space Surveillance and Tracking logic, from the analysis of .fits images and star filters to the review of candidates and the export of results.

Table 8.1: Results of the heuristic review applied to the SpotDebris prototype.

8.3 Performance evaluation

The pipeline processes each .fits file by reading the data and header (WCS), extracting metadata, generating a .png preview, querying the Gaia catalog for astrometric star overlays, and running ASTRiDE to detect streaks.

This section provides a summary of the pipeline performance evaluation for the twenty files used for testing, performed by the author. A headless test was executed using Playwright from the [34] (`scripts/measure_browser_e2e.py`). The script simulates a user operating the dashboard: from the selection and upload of a .fits file, to the actual preview served by the endpoint `/api/v1/runs/{run_id}/preview`.

File	**Upload→Preview (s)**
long.fits	6.3
mapa1.fits	8.7
ALPACA.2022-12-07T01_22_05.230.fits	8.7
ALPACA.2022-12-07T02_03_40.020.fits	10.7
MAPA_SURV_3_155194_220704_113148+442029_16069Bf_01.fits	21.3
MAPA_SURV_3_155268_220704_124149-023642_15017Af_01.fits	6.8
MAPA_SURV_4_085729_220426_113014-114909_scan001_01.fits	6.7
MAPA_SURV_4_095491_220508_134149-041543_scan001_01.fits	6.8
MAPA_SURV_4_100629_220511_093819+114543_scan001_01.fits	7.3
MAPA_SURV_4_124609_220603_153325-091249_scan001_01.fits	6.8
MAPA_SURV_4_125739_220604_103052+152024_rnd0002_01.fits	2.5
MAPA_SURV_4_145304_220616_162409-121306_scan001_01.fits	9.4
MAPA_SURV_4_145926_220619_163418-121438_scan001_01.fits	7.2
MAPA_SURV_4_152496_220624_202272+073139_scan122_01.fits	91.7
MAPA_SURV_4_153782_220705_170444-145134_scan001_01.fits	43.3
MAPA_SURV_4_157584_220707_170444-145134_scan001_01.fits	43.2
MAPA_SURV_4_159758_220707_170444-145134_scan001_01.fits	43.8
MAPA_SURV_4_161950_220708_170444-145134_scan001_01.fits	43.4
MAPA_SURV_4_164136_220709_130114-142121_scan001_01.fits	6.8
MAPA_SURV_4_166350_220710_135158-142548_scan001_01.fits	6.7
Median	**8.0 s**
Mean	**19.0 s**

Table 8.2: Observed upload→preview times for 20 .fits files (headless test).

In the Table 8.2, the first file is an ASTRiDE example, the other ones come from observations done in the Madeira island (Pico do Areeiro Optical Observatory), except for the "Alpaca" files, which were retrieved from the ESO Science Archive Facility ([13]).

The results demonstrate that the prototype shows to the preview of a given image, to the user, relatively fast (typically 6-10 s). However, more complex images, for instance, may exhibit longer execution times due to clouds or more dense star fields. The median (8 s) is therefore a robust indicator of typical performance, whereas the mean (19 s) reflects the natural variability of these images.

Note regarding the variability of observed execution times:

The times obtained in the headless test reflect the behavior observed in the browser, although they may vary between runs due to factors that do not directly depend on the interface, such as fluctuations in backend load, temporary network conditions, latency from external services or even the hardware on which the dashboard container is being run on. Specifically, stages involving queries to Gaia DR3 or ASTRiDE processing tend to exhibit greater variability, as they depend on

external services and their state at the time of execution.

8.4 Task-based usability assessment

Following the heuristic review, a usability evaluation of the system was carried out in accordance with what had been discussed with the domain experts earlier during asynchronous meetings. During the user testing, a set of ten participants was asked to complete a short, anonymous survey on Google Forms and to complete four simple tasks, testing core functionalities of the dashboard ([21]):

- **1st Task – .fits ingestion and metadata extraction:**
  - Load the .fits files (map1.fits and long.fits) from a local folder.
  - Verify that the image and sensor name are displayed correctly in the "Overview" tab.
  - Verify that the WCS (World Coordinate System) coordinates are displayed correctly; that is, the right ascension (RA) and declination (DEC) values on the screen update as you move the mouse.
- **2nd Task – Catalog cross-referencing and star identification:**
  - Go to the "Overview" tab and look at the star overlays from the Gaia DR3 algorithm in the image preview (yellow circles).
  - Adjust the magnitude slider to filter the stars, so that only star overlays with a magnitude greater than 12 are displayed.
  - Check that the overlay updates respond smoothly and quickly.
  - Check that useful star data information appears when you click on a random star overlay.
- **3rd Task – Debris candidate classification:**
  - Go to the "Candidates" tab and view the sequence detected by the ASTRiDE algorithm.
  - Click on the visible space debris streak.
  - Classify the "pending" detection as "confirm debris" or "reject" (as you wish).
  - Verify that the classification changes are retained when switching between tabs.
- **4th Task – Report export:**
  - Go to the history page.
  - Verify that everything is in order based on your acceptance or rejection of the space debris data in long.fits.
  - Return to the previous page by clicking "Back to dashboard".

– Export a report in .json format by clicking “export json”.

Details regarding the questions asked and the corresponding answers from the survey will be revealed later in the next subsection.

8.5 User testing results

After completing the four tasks listed above, the ten participants answered the following evaluation questions, discussed in this subsection.

8.5.1 Workload Assessment

Six workload dimensions were used to assess users’ cognitive and physical effort while completing the tasks. Each dimension was scored on a scale of 0 to 100 points, incorporating six levels (0, 20, 40, 60, 80 and 100 points), with 0 being the lowest (very low) and 100 the highest (very high). These six workload dimensions follow the NASA-TLX framework ([23]).

The six workload dimensions included in the questionnaire are shown in the table below:

#	Workload questionnaire items
1	How mentally demanding were the tasks?
2	How physically demanding were the tasks?
3	Did you feel time pressure while performing the tasks?
4	How satisfied are you with your performance? (0 = Completely dissatisfied, 100 = Completely satisfied)
5	How hard did you have to work to achieve your level of performance?
6	How insecure, discouraged, irritated, or stressed did you feel?

Table 8.3: Workload evaluation questionnaire items.

The participants’ responses, extracted from the survey, are shown below:

Participant	MD	PD	TD	Perf.	Effort	Frust.
1	60	0	0	100	0	0
2	20	0	0	100	40	0
3	20	0	0	80	0	0
4	20	0	0	0	20	0
5	0	0	0	80	20	0
6	0	40	0	80	20	0
7	20	0	0	80	40	0
8	40	0	0	80	60	0
9	40	0	0	60	60	0
10	0	0	0	60	80	0

Table 8.4: Raw NASA-TLX scores (0–100) for the six workload dimensions across all participants.

The average scores for all participants and for each workload dimension are as follows:

- Mental demand: 22
- Physical demand: 4
- Temporal demand: 0
- Performance satisfaction: 72
- Effort: 28
- Frustration: 0

The results show that participants reported a very low level of workload. From both a mental and physical standpoint, the tasks were undemanding, with no time pressure or frustration, revealing a high level of satisfaction with their performance and moderate to low effort required to achieve it.

8.5.2 System Usability Scale (SUS)

The System Usability Scale (SUS) is a standard, 10-question survey that that collects users' subjective opinions on how easy a product or service is to use and how easy it is to learn (perceived usability, [4]).

The score is calculated by assigning a score of -1 to answers to odd-numbered questions and a score of 5 to answers to even-numbered questions, adding all the scores together, and multiplying the total by 2.5, resulting in a value between 0 and 100 that indicates perceived usability.

All the ten participants completed the System Usability Scale questionnaire in the mentioned survey, which includes 10 items and each item a 5-point Likert scale:

- 1 = Strongly Disagree
- 2 = Disagree
- 3 = Neither Agree nor Disagree
- 4 = Agree
- 5 = Strongly Agree

The list of the 10 items covered in the questionnaire is as follows:

#	SUS Questionnaire items
1	I think that I would like to use this system frequently.
2	I found the system unnecessarily complex.
3	I thought the system was easy to use.
4	I think that I would need the support of a technical person to be able to use this system.
5	I found the various functions in this system were well integrated.
6	I thought there was too much inconsistency in this system.
7	I would imagine that most people would learn to use this system very quickly.
8	I found the system very cumbersome to use.
9	I felt very confident using the system.
10	I needed to learn a lot of things before I could get going with this system.

Table 8.5: System Usability Scale (SUS) questionnaire items.

The ten participants' answers to the questionnaire are:

P	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS
a	5	1	5	3	5	1	4	1	5	5	82.5
b	4	3	5	3	4	1	4	1	4	1	77.5
c	2	1	5	2	4	2	5	1	4	1	82.5
d	4	1	5	1	5	1	5	1	5	1	90.0
e	5	2	5	2	4	1	4	1	4	2	85.0
f	4	2	5	4	5	1	4	1	3	1	80.0
g	5	2	5	2	4	1	5	1	4	1	90.0
h	3	2	4	5	4	2	4	1	3	4	60.0
i	3	1	5	3	4	1	4	1	5	4	77.5
j	5	1	4	3	4	1	5	1	3	3	80.0

Table 8.6: Individual SUS responses (Q1–Q10) and corresponding SUS scores for all the ten participants. Here participants are not in numerical form for better visual distinction.

The average questionnaire score for the ten participants was 81.5 points, indicating an excellent perceived usability of the prototype, meaning that users find it easy to use and satisfactory.

8.5.3 Technology Acceptance (TAM/UTAUT)

The Technology Acceptance Model (TAM) and the Unified Theory of Acceptance and Use of Technology (UTAUT) were used to assess how participants perceived the usefulness and ease of use of the dashboard ([7, 59]).

The participants completed the questionnaire, which includes 9 items and uses the 5-point Likert scale (1 = Strongly Disagree, 5 = Strongly Agree).

The list of the 9 items included in the questionnaire is:

#	TAM/UTAUT Questionnaire items
1	Using this system could improve my performance in SST-related tasks.
2	This system is useful for analyzing .fits images.
3	This system increases my productivity.
4	Learning to use this system was easy for me.
5	Interacting with the system is clear and understandable.
6	I find the system easy to master.
7	This system could help me perform important tasks such as debris analysis or star-data visualization.
8	It is easy to learn how to use this system.
9	I have the resources and knowledge necessary to use this system.

Table 8.7: Technology Acceptance (TAM/UTAUT) questionnaire items.

Across all participants, the following results were obtained:

P	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	TAM
a	4	5	3	4	5	5	5	5	3	4.33
b	3	4	4	4	4	4	5	4	4	4.00
c	3	5	5	4	5	5	5	4	5	4.56
d	4	4	5	5	5	5	5	5	5	4.78
e	4	4	4	4	4	4	5	4	4	4.11
f	4	4	3	5	5	5	5	4	2	4.11
g	4	4	4	4	4	4	4	4	3	3.89
h	4	4	4	4	4	4	4	4	3	3.89
i	4	4	4	4	4	4	4	4	3	3.89
j	4	4	3	4	4	4	4	4	4	3.89

Table 8.8: Individual TAM/UTAUT responses (Q1–Q9) and corresponding TAM scores for all the ten participants. Here, also, participants are not in numerical form for better visual distinction.

The average TAM/UTAUT score across all participants was 4.1 points.

These results indicate that participants found the system easy to understand, manipulate, and learn. However, despite the positive score, users still believe that the prototype could be refined in future versions to further strengthen its alignment with SST logic.

8.5.4 User Experience (UEQ-S)

The User Experience Questionnaire (UEQ-S) was used to measure perceived user experience ([53]).

Users responded to the 8 items on a scale ranging from negative (1) to positive (7). The items are:

- Complicated . . . Simple
- Inefficient . . . Efficient

- Confusing ... Clear
- Unpredictable ... Predictable
- Unpleasant ... Pleasant
- Not stimulating ... Stimulating
- Boring ... Interesting
- Conventional ... Creative

The results obtained are, for the ten participants:

P	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	UEQ-S
a	6	7	7	6	7	7	7	7	6.75
b	6	7	6	4	7	7	7	6	6.25
c	7	4	7	6	6	3	4	4	5.13
d	7	7	7	7	7	7	7	5	6.75
e	7	7	7	4	7	6	6	6	6.25
f	7	6	6	6	7	5	6	5	6.00
g	6	6	6	6	6	4	4	5	5.38
h	6	6	6	5	7	5	4	5	5.50
i	5	5	6	5	7	5	7	5	5.63
j	7	7	7	5	7	4	6	4	5.88

Table 8.9: Individual UEQ-S responses (Q1–Q8) and corresponding UEQ-S scores for all the ten participants. Participants are not in numerical form for better visual distinction.

The average score is 5.95, bordering 6 points, which indicates a pleasant, clear, and easy to use interactive dashboard experience. However, in future versions, features could be added to further stimulate users and increase their involvement with the platform.

8.5.5 Qualitative comments

At the end of the survey, participants were invited to leave additional comments, as described below:

- **Positive impressions:** In the course of testing with the ten participants, they frequently praised the dark mode design for image analysis in low-light conditions, overall clean and modern design and the tool's ease of learning and use. They believe that the tool is superior to static image analysis, as it is a dynamic and interactive solution that can help expert operators perform their functions better. Furthermore, one participant noted the value of ASTRiDE's ability to automatically identify potential debris streaks, as well as the importance of manually verifying the algorithm's results.

- **Issues:** The majority of participants did not report any technical issues. However, one participant pointed out a limitation in the current file-loading process: all required files for the given tasks (long.fits and mapa1.fits) must be loaded simultaneously, with no intuitive button to go back to the previous page or update the files progressively (without manually typing the uploads homepage URL).
- **Suggestions for improvement:** The suggestions participants provided for future iterations include adding accessory plugins, providing an input box for magnitude filtering (K-mag), improving spacing on the History page, and enabling navigation back to the initial menu without reloading the entire application. One participant also suggested adding an option to export the entire star catalog data to .csv or .json and repositioning the export button to better align with the detections element, since it is more directed to detections than it is to stars, currently.

8.6 Requirements verification

The chart (8.10 provides a comprehensive review of the prototype's alignment with respect to the functional and non-functional requirements defined in Chapter 5.

ID	Requirement	Status
FR-IMG-01	Image import & handling	Verified
FR-ALG-01	Source detection	Verified
FR-CAT-01	Catalog validation	Verified
FR-CALC-01	Attribute calculation	Verified
FR-EXP-01	Reporting	Verified
FR-VIS-01	Data visualization	Verified
FR-INT-01	Interactive Analysis	Verified
FR-SYS-01	System Feedback	Verified
NFR-USE-01	Usability	Details below
NFR-USE-01.1	Dark mode for low-light analysis	Verified
NFR-USE-01.2	Color-blind friendly color scheme	Needs validation by testing
NFR-USE-01.3	Domain-specific scientific vocabulary	Verified
NFR-INT-01	Interactive Analysis	Verified
NFR-INS-01.1	Installation	Verified
NFR-ARQ-01.1	Architecture	Verified
UC-01	Automated star identification	Verified
UC-02	Debris candidate verification	Verified
UC-03	Export observation report	Verified

Table 8.10: Requirements compliance chart

Through practical demonstrations or tests, all the functional requirements were confirmed, proving that the prototype performs the intended tasks as expected. The non-functional requirements concerning usability, installation, and system architecture were also met. The only pending verification is for NFR-USE-01.2 (color-blind accessibility), which requires further testing, since the ten participants answered in the survey that they are not color-blind. However, initial heuristic

evaluations and the integration of shape-based cues suggest that this requirement is progressing well.

8.7 Results discussion

The evaluation presented throughout this chapter confirms that the SpotDebris dashboard, the minimum viable product of this dissertation, meets the expectations established in Chapter 4:

The prototype performs adequately for interactive use. Processing of .fits files, from upload to the display of an interactive preview in .png format (with .svg overlays), takes between 6 and 10 seconds, including processing by known external pipelines, which is a reasonably fast processing time, given the queries and external API requests done and and the variable nature of internet connection speed, for instance.

The prototype has a good perceived usability. Feedback received from the participants involved in the testing reveals that, overall, the platform is well designed and works properly as intended, with the system's design, clarity, and overall interactivity being on point. The prototype also meets the expectations of future specialists who may come to use it, although there is, naturally, room for improvement.

The requirements compliance chart further confirms the alignment between the implementation and the formal definition of requirements for this dissertation, emphasizing only the need for colorblind accessibility to undergo additional validation, and reinforcing that the proposed prototype fulfills the objective of providing an engineering approach for the visualization and annotation of Space Surveillance and Tracking images.

8.8 Limitations and future testing

While the evaluation provides valuable insights for perfecting the prototype in the future, it is crucial to recognize a few limitations that could affect the interpretation of the results:

1. **Sample size:** User testing was conducted with ten participants. Future studies may include more users for more robust and thorough analyzes.

2. **Gaia stars density:** The stellar density in the field of view of each processed image is limited primarily for performance reasons, as well as to reduce visual clutter. Dense globular clusters containing thousands of expected stars are not included, so testing and scalability in such scenarios may be necessary in the future.

3. **Streaking Realism:** Synthetic streaks used in ASTRiDE's validation do not fully represent the range of debris signatures seen in real observations, especially in cases with non-linear motion or saturation. A detailed assessment using expert-annotated detections from real data is left for future work.

4. **Formal usability metrics:** Although think-aloud protocols provided useful qualitative insights, more quantitative methods (such as eye-tracking or interaction tracking) could provide more accurate measurements of cognitive load and navigation patterns.

8.9 Summary

The evaluation demonstrates that the SpotDebris prototype meets the defined objectives and is in line with the established requirements, offering a functional, interactive platform for processing .fits images. It also allows for querying space related catalogs, assisted identification of space debris, as well as visualizing common star data within the image's field of view. The performance and questionnaire based metrics results confirm the system's suitability for the SST context, although future evaluations with more users, as well as other quantitative usability metrics could further reinforce these conclusions.

Chapter 9

Conclusions

9.1 Limitations of the current prototype

Despite being functional, the minimum viable product presents normal limitations, including:

1. Persistence architecture is not suitable for a multi-user environment or long development cycles.
2. As confirmed with testing, the availability and latency of the Gaia external service may interfere with the normal flow of work.
3. No authentication protocol was implemented.
4. Limited testing, requiring the use of various images (.fits) from additional sensors (telescopes).
5. The total number of stars cataloged by Gaia is limited, so certain images may contain many more stars that could be cataloged by Gaia, but are restricted in this pipeline for reasons of performance and user interaction fluidity.
6. Lack of dedicated task queue for heavy processing at scale.

In a future phase, as an evolution, it would make sense to gradually transition to a queue-based backend (such as Celery ([5]) or equivalent solutions), adopt a transactional persistence system, for instance, PostgreSQL ([48]), introduce caching mechanisms for astrometric results, and further evaluate with other metrics, besides the ones described in Chapter 4 and the criteria in Chapter 5.

9.2 Conclusion

As the need for efficient, interactive tools to identify space debris in Earth orbit has increased, the SpotDebris dashboard was developed to support the analysis of telescope images and address this limitation. Serving as a bridge between Space Surveillance and Tracking and dashboard engineering, the minimum viable product developed in this dissertation integrates, within a single platform,

the Gaia DR3 catalog for star identification, the ASTRiDE algorithm for automatic streak detection, and the SST workflow. This is achieved through the upload of .fits files to the export of results manually validated by the user. By combining the entire technical and scientific workflow into a single tool, the dashboard has filled the gap left by the lack of a unified, interactive tool capable of processing telescope images, with posterior manual validation and analysis.

The initial approach included a review of the state of the art in data visualization, dashboard engineering, and SST. This foundation enabled the formulation of appropriate research questions and provided a concrete direction for the development of this dissertation. Meetings were held with domain experts, resulting in the definition of functional and non-functional requirements aligned with ECSS best practices and ISO standards. These requirements guided the design of the system architecture, following a client-server model, integrating external catalogs and algorithms, and adopting a modular structure.

The resulting prototype demonstrates good integration between automatic streak detection in the field of view of previously loaded images, as well as queries to an external astrometric catalog and visual navigation. It is possible to perform tasks related to SST, such as candidate inspection, star identification, metadata extraction, and report export. The evaluation results confirm that the system meets the previously defined requirements. Performance tests demonstrate acceptable execution times, heuristic reviews validate the interface's alignment with the domain, and usability tests with users reveal high acceptance and a positive user experience.

This dissertation demonstrated that an engineering approach to the design of interactive dashboards can effectively support SST operators in better performing their tasks of interpreting and validating space debris trails in telescope images, such as those from the Pico do Areeiro Optical Observatory, among others. Iterative prototyping, compliance with standards, and rigorous requirements are the pillars of a dashboard that provides a solid foundation for future development, as well as a valuable contribution to the SST field, especially in situations where interactive, transparent, and domain-aligned tools are vital for operational decision-making.

9.3 Future Work

Although the SpotDebris dashboard has proven to be highly practical and useful for interactive analysis of telescope images within the context of the SST, future expansions could further improve the prototype:

1. Since the MVP is designed for a single user and relies on disk-based persistence, if the tool is to be scaled to multi-operator environments, a transactional database (e.g., PostgreSQL) and a distributed task queue (e.g., Celery or Redis Queue, [51]) could be implemented to support concurrent processing and provide long-term data retention.

2. The ASTRiDE algorithm provides a good baseline for streak detection within the field of view. However, future versions of the dashboard could integrate machine-learning-based algorithms (for instance, U-Net, Mask R-CNN or YOLO-style architectures) to improve

robustness, particularly in noisy, very dense star fields and image frames. A wider validation across different instruments and observational conditions would also strengthen the reliability of the detections.

3. While the prototype follows an ICD logical structure, a complete and formal ICD (document) may be developed to support integration with other external systems, ensuring maintainability and adhering to ECSS practices for multi-system environments.

4. The current prototype features color-blind-friendly cues. However, further testing is needed.

5. Although the current solution demonstrates high usability, future studies could involve a larger number of participants, including domain experts, more images from multiple sensors, and quantitative evaluation metrics such as eye-tracking and interaction tracking.

This list of five guidelines for future work once again highlights the potential of the SpotDebris dashboard to reach a higher operational level, enabling it to support better scientific research, as well as more effective operational monitoring and collaborative analysis in increasingly complex space surveillance environments.

References

[1] Sohrab Almasi et al. “Usability Evaluation of Dashboards: A Systematic Literature Review of Tools”. In: *Biomed Research International* 2023.fevereiro (2023), p. 9990933. DOI: 10.1155/2023/9990933. URL: https://doi.org/10.1155/2023/9990933.

[2] Atlantic Centre. *Policy Brief 08*. https://www.defesa.gov.pt/pt/pdefesa/ac/pub/acpubs/Documents/Atlantic-Centre_PB_08.pdf. Accessed 2026-06-24. 2025.

[3] Benjamin Bach, Euan Freeman, Alfie Abdul-Rahman, et al. “Dashboard Design Patterns”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2023), pp. 342–352. DOI: 10.1109/TVCG.2022.3209448. URL: https://doi.org/10.1109/TVCG.2022.3209448.

[4] 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 (2008), pp. 574–594. DOI: 10.1080/10447310802205776.

[5] Celery Project. *Celery Documentation*. Accessed: 2026-06-25. 2024. URL: https://docs.celeryq.dev.

[6] Xi Chen et al. “Composition and Configuration Patterns in Multiple-View Visualizations”. In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2021), pp. 1514–1524. DOI: 10.1109/TVCG.2020.3030338. URL: https://doi.org/10.1109/TVCG.2020.3030338.

[7] Fred D. Davis. “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology”. In: *MIS Quarterly* 13.3 (1989), pp. 319–340. DOI: 10.2307/249008. URL: https://doi.org/10.2307/249008.

[8] Django Software Foundation. *Django Documentation*. Accessed: 2026-06-25. 2024. URL: https://docs.djangoproject.com.

[9] Docker Inc. *Docker: Containerization Platform*. https://www.docker.com/. Accessed 2026-06-25. 2026.

[10] Dawn Dowding and Jacqueline A. Merrill. “The Development of Heuristics for Evaluation of Dashboard Visualizations”. In: *Applied Clinical Informatics* 9.3 (2018), pp. 511–518. DOI: 10.1055/s-0038-1666842.

[11] Encode OSS Ltd. *Uvicorn Documentation*. Accessed: 2026-06-25. 2024. URL: https://www.uvicorn.org.

[12] European Cooperation for Space Standardization. *ECSS-E-ST-40C Rev.1 – Software (30 April 2025)*. https://ecss.nl/standard/ecss-e-st-40c-rev-1-software-30-april-2025/. Accessed 2025-11-24.

[13] European Southern Observatory. *ESO Science Archive Facility*. https://archive.eso.org/. Accessed 2026-05-19. 2026.

[14] European Space Agency. *Space Safety Programme (S2P): Space Surveillance and Tracking (SST) Core Software and ISOC Suite*. https://www.esa.int/Safety_Security/Space_Safety. Accessed 2026-06-24. 2022.

[15] FastAPI Team. *FastAPI Documentation*. Accessed: 2026-06-25. 2024. URL: https://fastapi.tiangolo.com.

[16] Figma Inc. *Figma: Collaborative Interface Design Tool*. https://www.figma.com/. Accessed 2026-03-18. 2026.

[17] J. Filho et al. "Space Surveillance payload camera breadboard: Star tracking and debris detection algorithms". In: *Advances in Space Research* 72.10 (2023), pp. 4215–4228. DOI: 10.1016/j.asr.2023.08.033. URL: https://doi.org/10.1016/j.asr.2023.08.033.

[18] GitHub, Inc. *GitHub Documentation*. Accessed: 2026-06-25. 2024. URL: https://docs.github.com.

[19] Google. *NotebookLM: AI-Powered Research Assistant*. 2024. URL: https://notebooklm.google.

[20] Google. *The Go Programming Language Documentation*. Accessed: 2026-06-25. 2024. URL: https://go.dev/doc.

[21] Google LLC. *Google Forms*. https://www.google.com/forms/about/. Accessed: 2026-06-05. 2026.

[22] Grafana Labs. *Grafana Design System: Lists of Objects Patterns*. Dashboard UX Best Practices. 2024. URL: https://grafana.com/developers/saga/templates/lists-of-objects/.

[23] Sandra G. Hart and Lowell E. Staveland. "Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research". In: *Human Mental Workload*. Ed. by Peter A. Hancock and Najmedin Meshkati. Amsterdam: North-Holland, 1988, pp. 139–183.

[24] Alan R Hevner et al. "Design science in information systems research". In: *MIS quarterly* (2004), pp. 75–105.

[25] Streamlit Inc. *Streamlit Documentation*. Accessed: 2026-06-25. 2024. URL: https://docs.streamlit.io.

[26] *ISO 8601:2019 — Date and time — Representations for information interchange*. Accessed 2026-06-08. Geneva, Switzerland: International Organization for Standardization, 2019. URL: https://www.iso.org/standard/70907.html.

[27] *ISO/IEC/IEEE 29148:2018 Systems and software engineering – Life cycle processes – Requirements engineering*. Geneva, Switzerland: International Organization for Standardization, 2018.

[28] John A. Kennewell and Ba-Ngu Vo. "An Overview of Space Situational Awareness". In: *Proceedings of the 16th International Conference on Information Fusion (FUSION 2013)*. Accessed 2026-06-24. 2013, pp. 1–8. URL: https://ba-ngu.vo-au.com/vo/KV_SSA_FUSION13.pdf.

[29] Dae-Won Kim. "ASTRiDE: Automated Streak Detection for Astronomical Images". In: *Astronomy and Computing* 15 (2016), pp. 20–29. DOI: 10.1016/j.ascom.2016.02.003.

[30] P. B. Kruchten. “The 4+1 View Model of Architecture”. In: *IEEE Software* 12.6 (1995), pp. 42–50. DOI: 10.1109/52.469759.

[31] Patricia McCormick. “Space Situational Awareness in Europe: The Fractures and the Federative Aspects of European Space Efforts”. In: *Astropolitics* 13.1 (2015), pp. 43–64. DOI: 10.1080/14777622.2015.1012002.

[32] Mendeley. *Mendeley Reference Manager*. 2024. URL: https://www.mendeley.com.

[33] Meta Platforms, Inc. *React Documentation*. Accessed: 2026-06-25. 2024. URL: https://react.dev.

[34] Microsoft Corporation. *Playwright Documentation*. Accessed: 2026-06-25. 2024. URL: https://playwright.dev/docs/intro.

[35] Stephen R. Midway. “Principles of Effective Data Visualization”. In: *Patterns* 1.9 (2020), p. 100141. DOI: 10.1016/j.patter.2020.100141.

[36] A. Moitinho et al. “Gaia Data Release 1: The archive visualisation service”. In: *Astronomy & Astrophysics* 605 (2017). Accessed January 5, 2026, A52. DOI: 10.1051/0004-6361/201731059. URL: https://publons.com/wos-op/publon/18737420/.

[37] Marco Felice Montaruli et al. “An Orbit Determination Software Suite for Space Surveillance and Tracking Applications”. In: *CEAS Space Journal* 16.5 (2024), pp. 619–633. DOI: 10.1007/s12567-024-00535-1. URL: https://doi.org/10.1007/s12567-024-00535-1.

[38] Giacomo Muntoni et al. “Crowded Space: A Review on Radar Measurements for Space Debris Monitoring and Tracking”. In: *Applied Sciences* 11.4 (2021), p. 1364. DOI: 10.3390/app11041364. URL: https://doi.org/10.3390/app11041364.

[39] 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 (2020), p. 113322. DOI: 10.1016/j.dss.2020.113322.

[40] NASA. *Systems Engineering Handbook*. https://www.nasa.gov/reference/systems-engineering-handbook/. Accessed 2026-01-27. 2019.

[41] OpenAI. *Introducing GPT-5.3-Codex*. https://openai.com/index/introducing-gpt-5-3-codex/. Accessed 2026-04-17. Feb. 2026.

[42] OpenJS Foundation. *Node.js Documentation*. Accessed: 2026-06-25. 2024. URL: https://nodejs.org/docs/latest/api/.

[43] Pallets Project. *Flask Documentation*. Accessed: 2026-06-25. 2024. URL: https://flask.palletsprojects.com.

[44] Ken Peffers et al. “A design science research methodology for information systems research”. In: *DESRIST*. Springer. 2007, pp. 83–106.

[45] Pence, W. D. and Chiappetti, L. and Page, C. G. and Shaw, R. A. and Stobie, E. *Definition of the Flexible Image Transport System (FITS), Version 3.0*. Accessed: 2026-07-14. 2010.

[46] Pillow Contributors. *Pillow Documentation*. Accessed: 2026-06-25. 2024. URL: https://pillow.readthedocs.io.

[47] Plotly Technologies Inc. *Dash User Guide and Documentation*. Accessed: 2026-06-25. 2017. URL: https://dash.plotly.com.

[48] PostgreSQL Global Development Group. *PostgreSQL Documentation*. Accessed: 2026-06-25. 2024. URL: https://www.postgresql.org/docs/.

[49] Pydantic Team. *Pydantic Documentation*. Accessed: 2026-06-25. 2024. URL: https://docs.pydantic.dev.

[50] Python Software Foundation. *Python Documentation*. Accessed: 2026-06-25. 2024. URL: https://docs.python.org/3.

[51] Redis Ltd. *Redis Documentation*. Accessed: 2026-06-25. 2024. URL: https://redis.io/docs/latest/.

[52] Alper Sarikaya et al. "What Do We Talk About When We Talk About Dashboards?" In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), pp. 682–692. DOI: 10.1109/TVCG.2018.2864903. URL: https://doi.org/10.1109/TVCG.2018.2864903.

[53] Martin Schrepp, Andreas Hinderks, and Jörg Thomaschewski. "Construction of a Short Version of the User Experience Questionnaire (UEQ-S)". In: *International Journal of Interactive Multimedia and Artificial Intelligence* 4.6 (2017), pp. 103–108. DOI: 10.9781/ijimai.2017.09.001. URL: https://doi.org/10.9781/ijimai.2017.09.001.

[54] DeepL SE. *DeepL Write: AI-powered writing assistant*. 2026. URL: https://www.deepl.com/en/write.

[55] Vidya Setlur et al. "Heuristics for Supporting Cooperative Dashboard Design". In: *IEEE Transactions on Visualization and Computer Graphics* 30.1 (2024), pp. 370–380. DOI: 10.1109/TVCG.2023.3327158. URL: https://doi.org/10.1109/TVCG.2023.3327158.

[56] Smithsonian Astrophysical Observatory. *SAOImage DS9*. https://sites.google.com/cfa.harvard.edu/saoimageds9. Accessed 2026-05-12. 2026.

[57] Antonella Vallenari, Anthony G. A. Brown, Timo Prusti, et al. "Gaia Data Release 3: Summary of the Content and Survey Properties". In: *Astronomy Astrophysics* 674 (2023), A1. DOI: 10.1051/0004-6361/202243940.

[58] Andrea Vazquez-Ingelmo, Francisco J. Garcia-Penalvo, and Roberto Theron. "Information Dashboards and Tailoring Capabilities – A Systematic Literature Review". In: *IEEE Access* 7 (2019), pp. 109673–109688. DOI: 10.1109/ACCESS.2019.2933472.

[59] Viswanath Venkatesh et al. "User Acceptance of Information Technology: Toward a Unified View". In: *MIS Quarterly* 27.3 (2003), pp. 425–478. DOI: 10.2307/30036540. URL: https://doi.org/10.2307/30036540.

[60] Vite Team. *Vite Documentation*. Accessed: 2026-06-25. 2024. URL: https://vitejs.dev.

[61] Jagoda Walny, Christian Frisson, Mieka West, et al. "Data Changes Everything: Challenges and Opportunities in Data Visualization Design Handoff". In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2020), pp. 12–22. DOI: 10.1109/TVCG.2019.2934538.

[62] 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 (2007), pp. 1224–1231. DOI: 10.1109/TVCG.2007.70515.

[63] Zotero. *Zotero: Reference Management Software*. 2024. URL: https://www.zotero.org.

Appendix A

Metadata structure for .fits files

To ensure architectural transparency and data traceability within the Spot Debris pipeline, Table A.1 defines the core astrometric metadata extracted from the .fits image headers. The header also includes the World Coordinate System (WCS), which enables the transformation of celestial coordinates (RA/DEC) into pixel coordinates.

Table A.1: Core .fits header keywords utilized by the Spot Debris backend.

Keyword	**Example value**	**Description**
BITPIX	16	Data type format (bits per pixel).
NAXIS1	2048	Image width in pixels.
NAXIS2	2048	Image height in pixels.
DATE-OBS	'2026-07-07T12:00:00'	UTC timestamp of exposure start.
EXPTIME	1.3000	Exposure duration in seconds.
SITELAT	'32.73622'	Latitude of the observation site.
SITELONG	'-16.92576'	Longitude of the observation site.
CRVAL1	-16.9257	WCS reference right ascension (RA).
CRVAL2	32.7362	WCS reference declination (DEC).
RA	'21 18 23.00'	Target RA (center of FOV for Gaia queries).
DEC	'-12 53 10.0'	Target DEC (center of FOV for Gaia queries).