

Bachelor Thesis Proposal : Reconstruction of the Tobii I-VT-filter

Graf Cem IM7

Januar 2025

1 Context

Eye tracking technology is applied across a diverse set of fields, including medical diagnostics, psychological research, computer vision, human computer interaction and also in the gaming industry. In the eye tracking technology area, Tobii AB — referred to here simply as “Tobii” — is recognized as the global market leader [1], [2]. Since 2011, Tobii has developed and refined the I-VT (Identification by Velocity Threshold) filter, a classifier designed to differentiate fixation-based eye movements from saccades by applying a velocity threshold approach that considers the three-dimensional (3D) spatial characteristics of eye movements. A saccade is a rapid, jerky, conjugate eye movement that shifts both eyes together, with equal amplitude, from one fixation point to another. Saccades may be initiated voluntarily or occur reflexively [3]. This classifier, primarily integrated into their eye tracking hardware [4], facilitates the application of advanced visualization and analytical methods in eye tracking studies. Despite its widespread adoption, a proper open-source implementation of the I-VT filter remains unavailable, thereby complicating the enhancement and direct comparison of results across different eye-tracking research efforts.

2 Research Questions and Objectives

The primary aim of this thesis is to provide a deterministic, open-source reconstruction of the proprietary Tobii I-VT fixation filter. This work is based on the hypothesis that replication is achievable due to the extensive details provided in Tobii’s published whitepaper.

2.1 Research Questions

- **RQ1:** How closely can the open-source implementation replicate the classification results of the original Tobii I-VT filter?
- **RQ1.1:** What is the percentage agreement between the Tobii I-VT filter and this open source reconstruction?

- **RQ1.2:** How robust is the agreement across different parameter settings?

2.2 Research Objectives

To answer the research questions and ensure structured development, the following objectives are defined.

In this context, the objective is not only to reconstruct the filter deterministically, but also to achieve exact reproduction of the original Tobii I-VT classification results. Full (100%) agreement under identical conditions is considered the ideal benchmark for success. Any deviations will be systematically analyzed and documented.

- **O1:** Reverse-engineer the Tobii I-VT algorithm by comparing its published documentation with behavioral output on controlled datasets.
- **O2:** Apply structured software engineering methods, including:
 - RUP for iterative and phased planning,
 - TDD for continuous quality assurance,
 - CI for automated testing,
 - Docker for reproducible deployment.
- **O3:** Evaluate the reconstructed filter’s classification accuracy through comparison with the original Tobii’s system.
- **O4:** Develop an analysis module to evaluate the behavior of individual filter components.
- **O5:** Analyze performance metrics including Precision, Recall, F1 Score, False Positive Rate, Overall Agreement, and Cohen’s Kappa to assess the classification accuracy, robustness, and statistical reliability of the reconstructed filter.
- **O6:** Release the implementation and documentation as an open-source project to support future research and reproducibility.

3 Motivation

The Software Engineering Laboratory for Safe and Secure Systems LAS³, also operates an eye-tracking laboratory. It plans to develop its own eye-tracking systems and intends to adopt Tobii’s I-VT filter, since it is the standard on Tobii hardware. Making the filter open source allows for continuous development and encourages community contributions. It also improves the replicability of research results.

The open-source reconstruction provides important advantages for the research

community. It enables transparent processing of gaze data and supports consistent evaluation methods across different systems. This makes it easier to compare results from studies using Tobii software with those based on alternative solutions. In addition, researchers with access to raw data can reproduce key metrics using the open-source filter, even without access to proprietary Tobii tools.

4 State of Research and Theoretical Background

The Tobii I-VT filter is a more advanced version of the original I-VT method, which was first introduced through velocity-based algorithms [5]. Tobii refined this approach by adding a number of practical improvements, as outlined in their whitepaper [6]. One key improvement is better handling of missing data: the filter uses gap-fill interpolation to maintain continuity in gaze sequences, even when tracking data is briefly lost. It also includes noise reduction techniques to improve the reliability of fixation detection.

Additionally, the filter can optionally choose the more stable eye signal when both are available, and it filters out very short fixations that are likely just noise. Overall, the Tobii I-VT filter works as a step-by-step processing pipeline. Each phase helps clean up the data and identify meaningful eye movements more accurately.

4.1 I-VT Filter Components

1. **Gap Filling:** This initial stage addresses missing data points within the recorded stream. Interpolation techniques are employed to fill in these gaps, ensuring a continuous data set that is critical for subsequent analysis.
2. **Eye Selection:** When available, data from both eyes is evaluated, and the filter selects the most reliable signal. This standardization step optimizes the consistency and quality of the input data.
3. **Noise Reduction:** To refine the raw eye tracking data, noise reduction algorithms are applied. This process effectively minimizes random fluctuations and artifacts, thereby enhancing the fidelity of the subsequent analyses.
4. **Velocity Calculation:** The cleaned data is used to compute the angular velocity of eye movements. By analyzing the speed between consecutive gaze points, the filter generates metrics needed to differentiate various eye movement behaviors.
5. **I-VT Classification:** With velocity information in hand, the algorithm classifies segments of data as either fixations or saccades. This classification is essential for understanding and interpreting the dynamic patterns of eye movements.

6. **Merging Adjacent Fixations:** To reduce data fragmentation, adjacent fixations that occur in rapid succession are merged into a single, continuous fixation. This consolidation better reflects natural gaze behavior.
7. **Discarding Short Fixations:** Finally, any fixations that do not meet a minimum duration threshold are removed. This step ensures that only stable and meaningful fixation events are retained for further analysis.

This systematic approach enables the Tobii I-VT Filter to transform raw eye-tracking data into a precise and actionable format, ensuring robust results that are critical for a wide range of research applications.

4.2 Practical Relevance and Available Implementations

These enhancements contribute to the popularity of the I-VT classifier, as its ease of understanding and implementation makes it one of the most accessible classifiers available [6]. Moreover, variations of the I-VT filter have been proposed to suit different application contexts [7], reflecting its flexibility in accommodating diverse research and real-world scenarios.

An important tool in this domain is the Tobii Pro Lab software, which incorporates its own implementation of the I-VT filter with customizable parameters. Tobii Pro Lab provides a user-friendly interface for data analysis and visualization, making it a preferred choice for practitioners who require reliable and easily adjustable gaze filtering solutions [4].

In addition to Tobii’s proprietary solutions, there are open-source implementations available. For example, the GazeToolkit, released under an open-source license in 2019 [8], offers documentation and executable code. It is important to note that the GazeToolkit has not yet been cited in academic publications. My own tests with the GazeToolkit indicated discrepancies when compared to Tobii’s original I-VT filter, and despite attempts to contact the repository owner for clarification, further insights remain unavailable.

5 Approach

The approach is divided into five key aspects, presented below: Data Acquisition and Processing, System Architecture and Deployment, Requirements Analysis and Design, Implementation and finally the Analysis Component and Validation.

5.1 Data Acquisition and Processing

For this thesis, the LAS³ team has made available a 2GB raw dataset from an eye tracking study of a promotional thesis that applied the Tobii I-VT filter [9]. This allows for experimentation with alternative filter settings within Tobii Lab. Since the existing study does not include data from participants using monocular vision, it is necessary to generate additional testing data using a

Tobii eye tracker. I plan to conduct a simple eye tracking study to record this data, thereby expanding the test base for further evaluation of the I-VT filter. For data preprocessing, a Python script will be employed to transform the acquired data into a format suitable for subsequent analysis and to serve as a valid test set.

5.2 System Architecture and Deployment

The system architecture is designed with flexibility and scalability in mind, ensuring that the entire application can be deployed seamlessly across diverse environments. A key aspect of this strategy is the use of containerization technologies and an automated Continuous Integration and Continuous Deployment (CI/CD) pipeline.

Containerization and Pipeline Setup: To provide a flexible and portable environment, Docker containers will be employed. Docker allows the encapsulation of the application and its dependencies into lightweight, portable containers [10]. This approach ensures that the software operates consistently across different environments, from development and testing stages to production. Each component of the system is containerized, which simplifies scaling, maintenance, and rapid deployment.

In parallel, a CI/CD pipeline will be established on GitHub. This pipeline automates the integration, testing, and deployment processes, significantly reducing the time between code commits and production updates. With continuous integration, every code change is automatically built and tested, ensuring that new updates do not break the existing functionality. Continuous deployment further streamlines the workflow by automating the release of new versions into production, thus promoting a seamless and efficient development cycle.

Together, the use of Docker containers and the implementation of a CI/CD pipeline provide a robust framework that minimizes deployment errors, enhances scalability, and fosters an agile development environment.

5.3 Requirements Analysis and Design

This phase applies the Rational Unified Process (RUP) methodology following Larman's book [11] to systematically plan and structure the project. Functional and non-functional requirements are defined based on Tobii's paper, with detailed use cases clarifying system interactions and workflows. Additionally, a domain model, class model, and a System Sequence Diagram (SSD) are developed to document the system's structure and communication processes.

5.4 Implementation

Based on the principles of Test-Driven Development (TDD) as advocated by Jeff Langr [12], I am using this approach to implement the I-VT filter. In TDD, you begin by writing a test that describes the desired functionality, which initially

fails. This failing test then serves as a guide for your implementation following the Red–Green–Refactor cycle. Unlike conventional unit testing, where tests are typically added after the code is written, TDD ensures from the very beginning that the code meets the defined requirements. Based on Mäkinen and Münch [13], TDD significantly reduces defect densities and achieves higher test coverage, which translates into more maintainable and robust code. Although the initial development effort may be higher, the long-term benefits in reduced maintenance costs and improved software quality justify using TDD in this work.

5.5 Analysis Component and Validation

Initially, a dedicated analysis component will be developed incrementally. The primary function of this component is to structure and annotate the output of the reconstructed I-VT filter at each processing stage.

This approach is inspired by the step-by-step methods outlined in the Tobii whitepaper, aiming to provide transparency and reproducibility. The structured outputs will facilitate later external visualization and evaluation but will not include direct graphical rendering within the analysis component itself.

Following the implementation phase, a Python-based script will compare the outputs of the reconstructed I-VT filter against the original Tobii I-VT filter, which serves as the accuracy baseline. The validation will align the classified events to allow direct comparison. The evaluation considers Fixation, Saccade, and Unknown as possible classes, treating the classification task as a multi-class problem. A confusion matrix will then be generated to capture the classification performance across all classes. Based on this, key metrics such as Precision, Recall, F1 Score, False Positive Rate, Overall Agreement, and Cohen’s Kappa will be calculated. This approach aims to ensure an objective and reproducible evaluation of the reconstruction.

6 Project Schedule

Weeks 1–2: Inception Phase – Problem Understanding and Feasibility

- Define project vision, primary research question, and success criteria
- Identify stakeholders and document expectations
- Targeted literature review on eye-tracking and the Tobii I-VT filter (focus on implementation-relevant details)
- Initial exploration of the 2GB raw Tobii dataset (data quality, missing data patterns, export formats)

Weeks 3–4: Elaboration Phase – Requirements, Risks, and Architecture Baseline

- Derive functional and non-functional requirements from Tobii literature and early data findings

- Identify key technical risks (data validity handling, windowing, monocular/bino logic, reproducibility)
- Model system behavior (use cases, domain model, system sequence diagrams) and draft core class responsibilities
- Set up Docker-based development environment and CI (tests, linting, reproducible runs)
- Implement preprocessing scripts to harmonize datasets and exports
- Plan and (if feasible) start a small additional eye-tracking study to capture monocular test data

Weeks 5–10: Construction Phase – Iterative Implementation and Testing

- Build an executable I-VT pipeline baseline (velocity computation, classification, basic data flow)
- Implement core processing steps iteratively (e.g., gap handling, eye selection strategy, noise reduction, merging/discard rules)
- Develop and integrate the analysis module for step-wise inspection and debugging
- Strengthen automated testing (unit + regression tests against reference outputs) and improve Docker/CI integration
- Continuous refactoring, documentation of design decisions, and incremental thesis writing (methods + implementation chapters)

Weeks 11–12: Transition Phase – Validation and Finalization

- Validate the implementation using Python-based evaluation scripts and compare against the Tobii reference
- Analyze discrepancies and finalize parameter tuning / edge-case handling
- Finalize thesis (results, discussion, limitations, references) and polish figures/tables
- Package and release as open-source (README, reproducibility guide, usage examples)

7 Outline

1. Introduction
 - (a) Theoretical Background
 - (b) Motivation

2. Research Question
3. Literature Review and Related Work
4. Requirement Analysis
5. Architecture
 - (a) Rough Design
 - (b) Detailed Design
6. Implementation
 - (a) Development Environment
 - (b) Concrete implementation
7. Test
8. Deployment
9. Analysis and Validation
10. Conclusion, limitations, future work
11. Acknowledgments

References

- [1] *Eye Tracking Market Share*. [Online]. Available: <https://www.gminsights.com/industry-analysis/eye-tracking-market> (visited on 04/23/2025).
- [2] Tobii, *Net sales of Tobii from 2014 to 2020 (in million SEK) [Graph]*, Feb. 2021. [Online]. Available: <https://www.statista.com/statistics/1003717/net-sales-tobii/> (visited on 04/23/2025).
- [3] R. Brandes, F. Lang, and R. F. Schmidt, Eds., *Physiologie des Menschen: mit Pathophysiologie* (Springer-Lehrbuch), de. Berlin, Heidelberg: Springer Berlin Heidelberg, 2019, ISBN: 978-3-662-56467-7 978-3-662-56468-4. DOI: 10.1007/978-3-662-56468-4. [Online]. Available: <http://link.springer.com/10.1007/978-3-662-56468-4> (visited on 04/25/2025).
- [4] Tobii AB, *Eye movement classification*, Nov. 2022. [Online]. Available: https://connect.tobii.com/s/article/eye-movement-classification?language=en_US (visited on 04/23/2025).
- [5] D. D. Salvucci and J. H. Goldberg, “Identifying fixations and saccades in eye-tracking protocols,” in *Proceedings of the symposium on Eye tracking research & applications - ETRA '00*, Palm Beach Gardens, Florida, United States: ACM Press, 2000, pp. 71–78, ISBN: 978-1-58113-280-9. DOI: 10.1145/355017.355028. [Online]. Available: <http://portal.acm.org/citation.cfm?doid=355017.355028> (visited on 04/23/2025).

- [6] A. Olsen, “The Tobii I-VT Fixation Filter,” p. 21, Mar. 2012.
- [7] J. Trabulsi, K. Norouzi, S. Suurmets, M. Storm, and T. Z. Ramsøy, “Optimizing Fixation Filters for Eye-Tracking on Small Screens,” *Frontiers in Neuroscience*, vol. 15, p. 578439, Nov. 2021, issn: 1662-453X. DOI: 10.3389/fnins.2021.578439. [Online]. Available: <https://www.frontiersin.org/articles/10.3389/fnins.2021.578439/full> (visited on 04/23/2025).
- [8] Martin Konopka, *GazeToolkit [GitHub repository]*, 2019. [Online]. Available: <https://github.com/uxifiit/GazeToolkit> (visited on 04/23/2025).
- [9] F. Hauser, “Visuelle Expertise bei Code Reviews,” 2024, Publisher: Universität Regensburg. DOI: 10.5283/EPUB.59753. [Online]. Available: <https://epub.uni-regensburg.de/id/eprint/59753> (visited on 04/23/2025).
- [10] Docker Inc, *What is Docker?* en. [Online]. Available: <https://docs.docker.com/get-started/docker-overview/> (visited on 04/23/2025).
- [11] C. Larman, *UML 2 und Patterns angewendet - objektorientierte Softwareentwicklung: use cases, "Gang of Four"-Patterns und GRASP; umfangreiche Fallstudien und zahlreiche Praxistipps* (Software-Entwicklung), 1. Aufl., [Nachdr.] Frechen: Mitp, 2009, ISBN: 978-3-8266-1453-8.
- [12] J. Langr, *Modern C++ Programming with Test-Driven Development: Code Better, Sleep Better*, 1st ed. Raleigh: Pragmatic Programmers, LLC, The, 2013, ISBN: 978-1-68050-402-6.
- [13] S. Mäkinen and J. Münch, “Effects of Test-Driven Development: A Comparative Analysis of Empirical Studies,” in *Software Quality. Model-Based Approaches for Advanced Software and Systems Engineering*, D. Winkler, S. Biffel, and J. Bergsmann, Eds., vol. 166, Series Title: Lecture Notes in Business Information Processing, Cham: Springer International Publishing, 2014, pp. 155–169, ISBN: 978-3-319-03601-4. DOI: 10.1007/978-3-319-03602-1_10. [Online]. Available: http://link.springer.com/10.1007/978-3-319-03602-1_10 (visited on 04/23/2025).