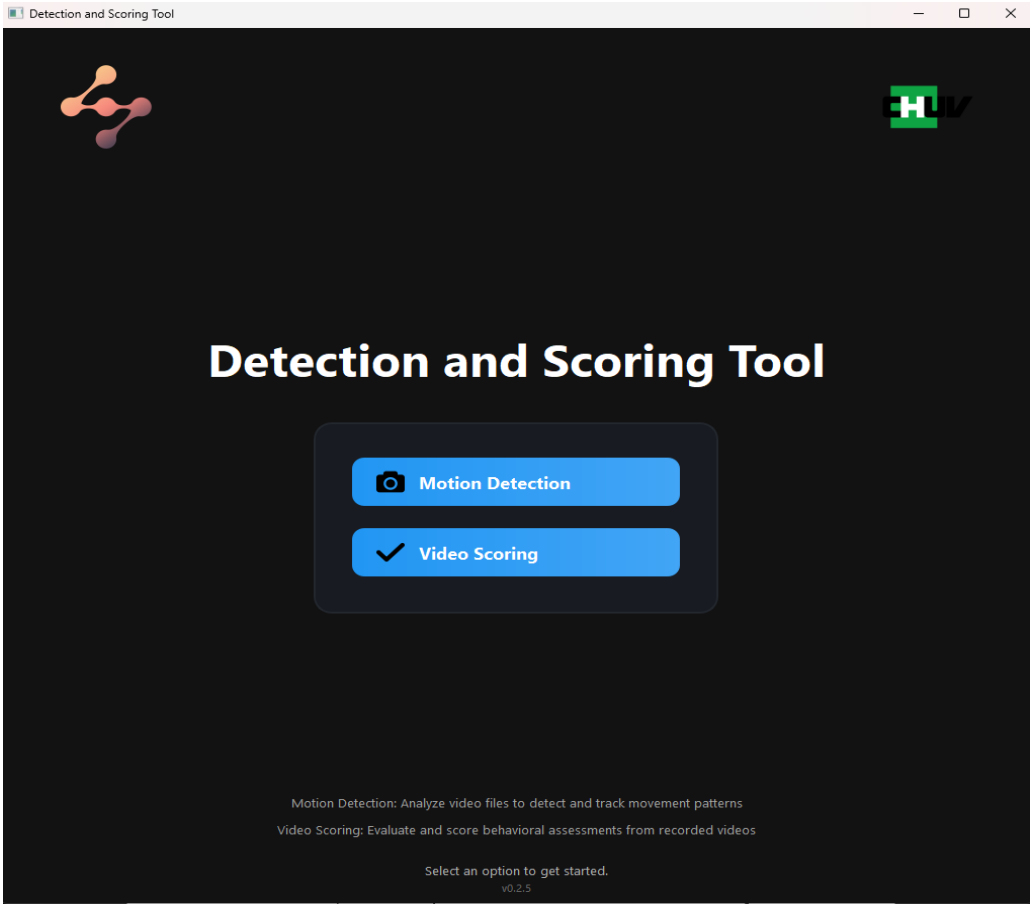
Hes·so VALAIS WALLIS
Haute Ecole de Gestion
Hochschule für Wirtschaft

Filière Informatique de gestion
Studiengang Wirtschaftsinformatik

Bachelor Thesis 2025

Development and improvement of a user-friendly user interface for motion detection analysis from video recordings



Student : Pedro Henrique Gomes de Lima

Advisor : Jean-Paul Calbimonte

Summary

This bachelor thesis was carried out within the Business Information Technology program at HES-SO Valais in collaboration with the University Hospital of Lausanne (CHUV) and supported by The Sense innovation and research center. The project aimed to develop a user-friendly software tool to support the detection and scoring of parasomniac episodes in home-recorded sleep videos, improving standardization and efficiency in clinical research workflows.

Initially focused on creating a graphical interface for manual scoring, the scope expanded after discussions with the clinical stakeholder revealed the need for a complete solution, including automated motion detection. The final application includes two main modules: a configurable motion detection system supporting single and batch processing, and an integrated scoring interface with video playback, structured annotation forms, and conflict management.

The project was developed iteratively with stakeholder feedback, using established tools and workflows to ensure quality, reproducibility, and clinical usability. Testing confirmed that the software meets its objectives by enabling clinicians to detect, annotate, and export parasomnia events in a structured and traceable format, while laying the groundwork for future improvements such as enhanced conflict resolution.

Acknowledgements

I would like to thank Dr. Jean-Paul Calbimonte for his valuable feedback throughout the project and for taking the time to review and correct this document. His guidance really helped keep the work on track.

I also want to give special thanks to Nina Rimorini, the main clinical stakeholder and end user of this project. She was incredibly helpful in making sure I understood the project's needs and context, organizing the visit to the CHUV, and giving important feedback and practical testing. I'm especially grateful for her patience in answering my many questions all along the way, and for being such an understanding and supportive collaborator throughout the development.

Table of Content

1. Introduction 1
1.1 Context 1
1.2 Objective 1
1.3 Motivation 2
2. Work Organization 3
2.1 Management Methodology 3
2.2 Tracking Tools 4
3. State of the Art 5
3.1 NREM parasomnias 5
3.1.1 Introduction to Sleep Architecture 5
3.1.2 NREM Parasomnias: Definition and Clinical Features 6
3.1.3 Management and Hypnosis as a Therapeutic Modality 7
3.2 UI Design Guidelines 8
3.2.1 Foundational Principles in UI Design 8
3.2.2 Current Trends and Practices in UI Design 11
3.3 Video-Based Motion Detection 12
3.3.1 Motion Detection Techniques 13
3.3.2 Software and Libraries 14
3.3.3 Benefits and Limitations 15
4. Methodology 16
4.1 Current State of the Project 16
4.1.1 Overview of the Research Study 16
4.1.2 Manual Scoring Workflow 18
4.1.3 Existing Python Script 19
4.1.4 Clinical Constraints and Ethical Context 22
4.2 Planned Improvements 22
4.2.1 GUI Framework Evaluation and Justification 23
4.2.2 Initial Exploration and Prototype Development 23
4.2.3 Comparative Analysis of Frameworks 24
4.2.4 Functional Objectives of the Final Application 25
4.3 Design Considerations for Integration 26
4.4 User Collaboration and Testing 28
4.5 Development Tools and Workflow 28

5. Results 30
5.1 Solution Overview 30
5.1.1 General Concept 30
5.1.2 User Interface Overview 31
5.1.3 Excel Output and Result Structure 40
5.1.4 Detection Workflow 43
5.2 Architecture and Dependencies 45
5.2.1 Overall Structure and Organization 46
5.2.2 Configuration Management and Data Flow 49
5.2.3 Dependencies and Environment Management 49
5.3 Testing Results 50
5.4 Proof of Concept Validation 51
5.5 Achievement of Objectives 53
6. Discussion 56
6.1 Critical Analysis of the Development Process 56
6.2 Interpretation of Results and Their Clinical Relevance 57
7. Conclusion 58
7.1 Delivered Results 58
7.2 Perspectives 58
8. Bibliography 60
9. Declaration of authorship 62

Table of Figures

Figure 1 Scrum Framework 3
Figure 2 Timeline project diagram 4
Figure 3 Sleep Cycle 6
Figure 4 10 usability heuristics 10
Figure 5 Detection with Frame Differencing 13
Figure 6 Design of the study 17
Figure 7 Example video recorded 18
Figure 8 First prototype in PyQt5 23
Figure 9 Landing Page of the Detection and Scoring Tool 32
Figure 10 Motion Detection – Single Video Processing Interface 33
Figure 11 Motion Detection – Batch Processing Interface 34
Figure 12 Motion Detection – Parameter Configuration View 35
Figure 13 Video Scoring Interface 36
Figure 14 Video Scoring – Segment Annotation Form 38
Figure 15 Conflicts View – Comparison and Resolution Interface 39
Figure 16 Excel Output – Detections Tab 40
Figure 17 Excel Output – Scoring Results Tab 41
Figure 18 Excel Output – Agreements Tab 42
Figure 19 Base Background Substraction Detection Pipeline 44
Figure 20 Architecture schema 46
Figure 21 Motion detection module diagram 47
Figure 22 Scoring module diagram 47
Figure 23 Data flow diagram 49

Table of Tables

Table 1: Framework Comparison: Streamlit, PyQt5, and Tkinter 24
Table 2 Compilation of data from testing period..... 52
Table 3 Comparison between provided scripts and final integration 55

List of Abbreviations

AI	Artificial Intelligence
GUI	Graphical User Interface
ML	Machine Learning
CEMIC	Center for Integrative and Complementary Medicine
CHUV	Centre Hospitalier Universitaire Vaudois
CIRS	Center for Sleep Investigation and Research
NREM	Non-Rapid Eye Movement
JPG	Joint Photographic Experts Group
MP4	MPEG-4 Video File Format
UI	User interface
HCI	Human-Computer Interaction
EEG	Electroencephalography
PSG	Polysomnography
CSV	Comma-Separated Values
FPS	Frames Per Second
MOG2	Mixture of Gaussians 2

1. Introduction

1.1 Context

This bachelor thesis is carried out within the Business Information Technology program at HES-SO Valais-Wallis, in collaboration with the University Hospital of Lausanne (CHUV). The project is supported by the Center for Integrative and Complementary Medicine (CEMIC)[1] and the Sleep Investigation and Research Center (CIRS)[2], both part of the CHUV, as well as the SENSE[3] innovation and research center.

The clinical research project at the core of this thesis focuses on the evaluation of parasomniac episodes in patients suffering from non-rapid eye movement (NREM) parasomnia, including conditions such as night terrors, sleepwalking, and confusional arousals. These episodes are studied in the context of a non-pharmacological treatment. To monitor these phenomena, video recordings are captured over multiple nights using a Tapo C210 (TP-Link) motion detection camera. The output consists of continuous video files (mp4) and snapshot folders (jpg), organized by date and time.

This ongoing study provided a practical and collaborative environment, offering the opportunity to observe clinical processes and contribute directly to a real-world research effort in the field of sleep medicine.

1.2 Objective

The initial phase of the project involved the development of a Python-based tool capable of detecting motion through frame-to-frame statistical analysis (OpenCV, 2025) and generating trigger points for detected events. However, this tool lacks a graphical user interface (GUI), making it difficult for non-technical users such as clinical researchers and medical staff to effectively interpret and label the data.

Originally, the thesis datasheet defined the goal of developing a tool that could:

- Visualize detected motion triggers along a video timeline,
- Assign “correct” or “incorrect” labels to each trigger (with the default set as correct),
- Export relevant video segments based on validated triggers, with user-defined buffer durations before and after each event.

During development, the project scope was expanded in collaboration with the clinical stakeholder to better match actual workflow needs. As a result, the final application includes two fully developed components: a motion detection module and an enhanced scoring system.

The motion detection module supports two algorithms, Frame Difference (OpenCV, 2025) and

[1] Center for Integrative and Complementary Medicine (CEMIC), CHUV: https://www.chuv.ch/fr/cemic
[2] Sleep Investigation and Research Center (CIRS), CHUV. https://www.chuv.ch/fr/sommeil/cirs-home
[3] The Sense Innovation and Research Center. https://www.the-sense.ch/

Background Subtraction (OpenCV, 2025), each with configurable parameters accessible through the interface. Users can analyze single videos or batch process multiple files, defining specific analysis windows and exporting structured Excel files containing detailed detection results and settings for reproducibility.

The scoring module extends far beyond simple correct/incorrect labeling. It provides an interactive form that lets users validate detections, select the episode type and intensity, edit time intervals, record participant and session details, and add optional comments. All annotations are attributed to individual scorers and exported in a structured, human-readable Excel format.

Additionally, the tool includes a conflict detection feature that compares scorer data, identifies disagreements, and allows users to record agreements

This application is designed as an integrated, standalone solution that streamlines the entire workflow for clinicians and researchers working with motion detection data in sleep studies.

1.3 Motivation

The motivation behind this project lies in the need to make the motion analysis tool truly usable and accessible to its primary target users at CHUV. These users are not generic “clinicians and researchers” but a specific clinical research team specializing in sleep medicine and parasomnia studies.

Among them, the main end user and primary point of contact for this project was a PhD student conducting detailed research on parasomniac episodes. She is directly responsible for reviewing, scoring, and analyzing the recorded sleep videos as part of the study. Her role as the primary tester and real-world user of the software made her feedback especially important for shaping the design and functionality of the tool.

More broadly, the application is intended to support the workflow of the entire clinical research team she is part of. This includes other clinicians and research staff who collaborate on scoring, validating, and analyzing parasomniac episodes captured in home-recorded sleep videos. As the existing script operated entirely without a user interface, it presented a significant barrier for these non-technical users without programming backgrounds.

By developing an intuitive and efficient graphical interface (GUI), the project aims to:

- Lower the technical entry barrier for the PhD student and the broader research team at CHUV,
- Increase the accuracy and speed of motion event labeling,
- Streamline and standardize the scoring workflow used in clinical research studies.

Ultimately, providing an accessible interface will not only improve the effectiveness of data processing but also ensure that the tool can be reliably integrated into real clinical workflows, supporting consistent and reproducible research practices at CHUV.

2. Work Organization

2.1 Management Methodology

For this project, the work organization is based primarily on the Scrum methodology, a widely used agile project management framework (Scrum.org, 2025). Scrum is founded on an iterative and incremental approach that allows development to evolve in response to feedback and changing requirements.

Development will be carried out in short iterations, called sprints, during which progress is continuously integrated and evaluated. A product backlog is maintained in an Excel file to guide the project and reflect ongoing priorities.

Although standard Scrum ceremonies are not applied, regular meetings ensure structured communication. Weekly meetings with the academic advisor focus on documentation progress and workflow validation. Additionally, meetings are held every one to two weeks with the PhD student stakeholder to review code updates, raise questions, and adapt requirements when needed.

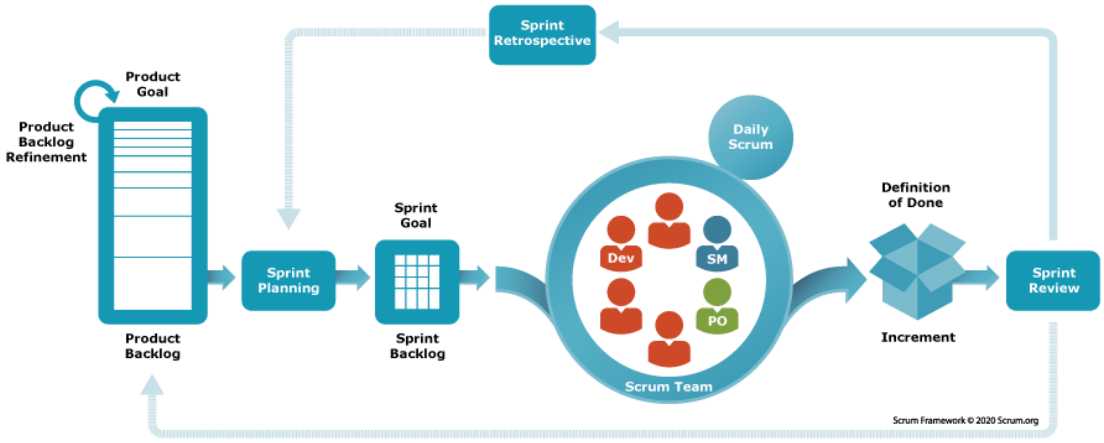


Figure 1 Scrum Framework

Scrum.org. (n.d.). What is Scrum? Retrieved July 4, 2025, from https://www.scrum.org/resources/what-scrum-module

2.2 Tracking Tools

To support this agile, iterative approach in practice, the project used structured tracking tools to maintain clarity and alignment. An Excel-based product backlog served as the main planning document, detailing features, tracking tasks, and recording evolving priorities. This practical artifact ensured that all planned work remained visible and adaptable to changing needs.

In addition to the backlog, meeting records were systematically maintained, capturing dates, participants, agendas, key decisions, and notes. This consistent documentation provided transparency and a clear audit trail of the project's communication and decision-making process.

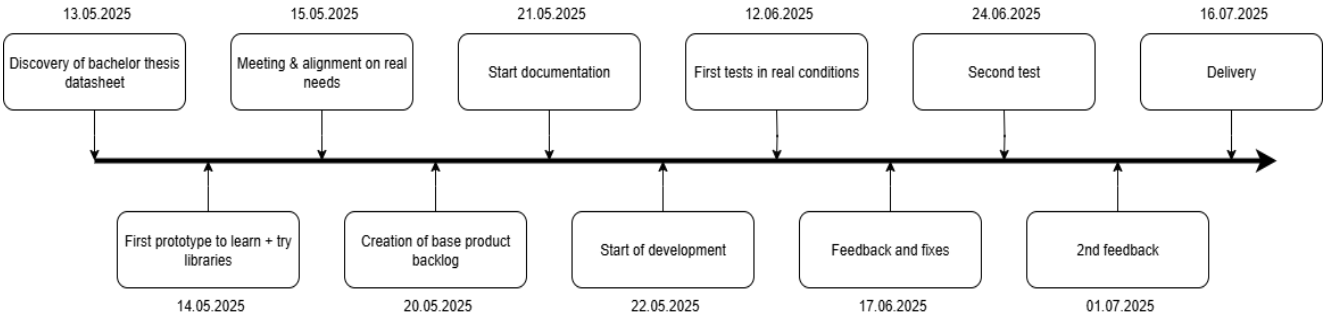


Figure 2 Timeline project diagram

(Source: Author)

3. State of the Art

3.1 NREM parasomnias

3.1.1 Introduction to Sleep Architecture

Sleep is a complex physiological process essential for cognitive function, emotional regulation, and overall health. It is broadly categorized into two distinct phases: non-rapid eye movement (NREM) sleep and rapid eye movement (REM) sleep. These phases alternate cyclically throughout the night, forming what is known as sleep architecture.

A typical sleep cycle lasts approximately 90 to 110 minutes and repeats four to six times per night. Each cycle progresses through NREM stages N1 to N3, followed by REM sleep. The proportion of REM sleep increases in successive cycles, while deep NREM sleep predominates in the earlier part of the night. (Patel et al., 2024)

NREM sleep comprises three stages, each characterized by distinct electroencephalographic (EEG) patterns and physiological features:

Stage N1: This transitional phase marks the onset of sleep, lasting 1 to 7 minutes. It is characterized by a reduction in alpha wave activity and the emergence of theta waves. Muscle activity decreases, and individuals can be easily awakened.

Stage N2: Representing the largest proportion of total sleep time, this stage is marked by the presence of sleep spindles and K-complexes on EEG. These features are associated with memory consolidation and sensory processing. Arousal thresholds increase, making it more challenging to awaken the sleeper.

Stage N3: Also known as slow-wave sleep (SWS) or deep sleep, this stage is characterized by high-amplitude, low-frequency delta waves. It is crucial for physical restoration, immune function, and declarative memory consolidation. Arousal from this stage is difficult, and individuals may experience sleep inertia upon awakening.

(Colten & Altevogt, 2006) (Kryger et al., 2022)

REM sleep is distinguished by rapid eye movements, muscle atonia, and EEG patterns resembling wakefulness. It is the primary stage for vivid dreaming and plays a vital role in emotional regulation and procedural memory consolidation. REM sleep typically constitutes 20–25% of total sleep time in adults. (Sleep Foundation, 2025) (Colten & Altevogt, 2006) (Kryger et al., 2022)

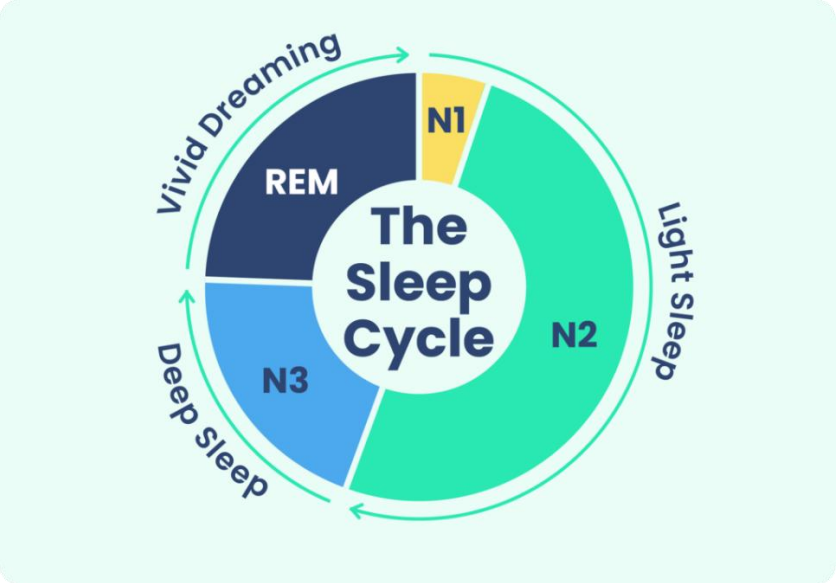


Figure 3 Sleep Cycle

From Stages of Sleep, by Sleep Foundation, 2023, SleepFoundation.org. https://www.sleepfoundation.org/stages-of-sleep

3.1.2 NREM Parasomnias: Definition and Clinical Features

NREM parasomnias are undesirable behaviors or experiences occurring during NREM sleep, particularly during the transition from deep sleep to lighter stages or wakefulness. They are classified under disorders of arousal and include:

- Confusional Arousals: Episodes of mental confusion or disorientation upon awakening, often with minimal motor activity.
- Sleep Terrors: Sudden arousals accompanied by intense fear, screaming, and autonomic activation, with little to no recall of the event.
- Sleepwalking (Somnambulism): Complex behaviors involving leaving the bed and performing routine activities while ambulating, typically with amnesia for the event. (Sateia, 2014)

These parasomnias are most prevalent in childhood but can persist into adulthood. A study reported a lifetime prevalence of 7% for NREM parasomnias in the general population. (Stallman, 2016)

The pathogenesis of NREM parasomnias is multifactorial and results from the interplay of predisposing, priming, and precipitating factors. Predisposing factors include genetic influences, as evidenced by familial aggregation observed in cases of sleepwalking and night terrors. Priming factors affect the depth or stability of NREM sleep; these include sleep deprivation, stress, and the use of certain medications such as sedatives. Finally, precipitating factors whether external, such

as noise or physical contact, or internal, such as obstructive sleep apnea can provoke partial arousals from deep sleep, thereby triggering parasomniac behaviors.

Diagnosing NREM parasomnias is challenging due to the sporadic nature of episodes and the frequent absence of recall. Polysomnography (PSG) in a sleep laboratory may not capture events, especially if they occur infrequently. Home-based video monitoring has emerged as a valuable tool, allowing for extended observation in the patient's natural sleep environment. (Mainieri, 2023) (Dauvilliers, 2022)

3.1.3 Management and Hypnosis as a Therapeutic Modality

The management of NREM parasomnias relies on a multidimensional approach that integrates safety precautions, behavioral strategies, and, when necessary, pharmacological treatment. Creating a safe sleep environment is essential to minimize the risk of injury during episodes, particularly in individuals prone to complex or potentially harmful behaviors.

On a behavioral level, interventions such as scheduled awakenings, stress reduction techniques, and cognitive-behavioral strategies may help reduce episode frequency in some patients, although evidence remains limited and primarily drawn from case series and clinical experience. (Attarian, 2013) Pharmacotherapy, most commonly involving agents like clonazepam, may be considered in selected cases; however, the evidence supporting its effectiveness is limited. Additionally, concerns regarding side effects and the potential for dependence warrant a cautious and individualized approach to medication use.

Hypnotherapy has been investigated as a non-pharmacological approach to treating NREM parasomnias, based on the induction of a trance-like state intended to enhance suggestibility and promote behavioral modification. While the available evidence is limited, some studies have reported encouraging outcomes. In one study involving 36 patients, 45.4% were either symptom-free or showed significant improvement at a one-month follow-up after just one or two sessions of hypnotherapy. Similarly, a case series reported a 75% efficacy rate in individuals presenting with sleepwalking and night terrors following a brief hypnotherapeutic intervention. Despite these promising findings, the overall quality of evidence remains low, primarily due to methodological shortcomings and small sample sizes. Consequently, further randomized controlled trials are necessary to validate these results and to establish standardized hypnotherapeutic protocols for the management of NREM parasomnias. (Mainieri, 2023) (Hauri, 2007) (Chamine, 2018)

3.2 UI Design Guidelines

User Interface (UI) design is a critical discipline within Human-Computer Interaction (HCI) that focuses on creating interfaces facilitating effective interaction between users and digital systems (Shneiderman, 2005). The primary objective of UI design is to enhance user satisfaction by improving the usability, accessibility, and overall experience during interaction with a system. In domains such as healthcare and scientific research, where precision and clarity are paramount, well-designed interfaces are essential to ensure that users can interact with complex data and systems efficiently and accurately.

3.2.1 Foundational Principles in UI Design

Ben Shneiderman, a leading figure in Human-Computer Interaction (HCI), introduced a set of guiding principles that have become essential in the design of user-friendly digital interfaces (Shneiderman, 2005). These recommendations, first outlined in his foundational text Designing the User Interface, offer a practical framework to enhance user experience and system usability.

- Maintain Consistency: Interfaces should behave predictably across similar situations. This includes using uniform terminology, commands, and layouts throughout the system, which helps users form accurate mental models and reduces the need for re-learning in different contexts.
- Provide Shortcuts for Experienced Users: As users gain familiarity with the system, they should be able to speed up their interaction through mechanisms like keyboard shortcuts, abbreviations, or customizable command sequences. These enhancements improve efficiency without hindering new users.
- Deliver Clear Feedback: Every user action should generate a timely and understandable response from the system. The nature of the feedback should match the significance of the action, minor inputs may need only subtle confirmation, while major operations warrant more explicit indicators.
- Structure Interaction for Completion: Tasks should be designed to have a clear beginning, progression, and conclusion. Completing a sequence should prompt informative feedback, reinforcing that a goal has been achieved and signaling readiness to proceed to the next task.
- Prevent and Handle Errors Gracefully: Interfaces should be designed to minimize the likelihood of errors by constraining inputs and guiding user actions. When mistakes do occur, the system should provide meaningful, non-technical explanations along with options for correction.
- Support Undo and Redo: Users should have the ability to easily reverse or cancel actions. This reduces fear of making mistakes and encourages exploration by giving users confidence that they can recover from unintended consequences.

- Foster User Control: The interface should empower users by making them feel in command of the interaction. Systems that override user input or behave unpredictably undermine confidence and can lead to frustration or misuse.
- Minimize Memory Load: Since human short-term memory is limited, the interface should avoid requiring users to recall information between screens. Important elements should remain visible or be easily retrievable, and workflows should be streamlined to minimize unnecessary steps.

Together, these eight principles form a cohesive strategy for designing intuitive and robust interfaces. They remain highly influential in modern UI/UX practice and are particularly relevant in specialized domains like clinical software, where clarity and reliability are critical. (Shneiderman, 2005)

Jakob Nielsen's usability heuristics, first introduced in 1995, provide a widely respected framework for evaluating the effectiveness and user-friendliness of digital interfaces (Nielsen, 1995). These principles serve as practical guidelines during both design and usability assessment phases.

- System Status Transparency: Interfaces should provide timely and clear feedback to users, ensuring they are continuously informed about the system's current activity or response.
- Real-World Language and Context: The terminology and structure of the interface should mirror familiar, real-world conventions to facilitate intuitive understanding and navigation.
- Freedom to Navigate and Correct: Users should be able to easily reverse unintended actions or escape from unwanted states, promoting a sense of control and reducing frustration.
- Uniformity and Predictability: Consistent design elements such as commands, icons, and layout structures, should be used throughout the system, allowing users to predict behavior and reduce learning time.
- Proactive Error Avoidance: Rather than simply handling errors after they occur, the interface should be designed to prevent common mistakes through thoughtful design choices and constraints.
- Support for Recognition over Recall: To lessen cognitive load, users should not be required to remember information between screens. Instead, key options and actions should always be easily accessible and visible.
- Adaptability for Skill Levels: Systems should cater to both novice and expert users by including mechanisms like shortcuts or customization options that improve efficiency for experienced users without overwhelming beginners.
- Clear and Focused Layout: Visual design should be simple and uncluttered, presenting only relevant content to avoid distractions and improve overall readability.
- Constructive Error Messaging: When errors occur, messages should be written in plain

language, clearly identify the issue, and suggest actionable steps for recovery.

- Accessible Support Resources: Although well-designed systems aim to be self-explanatory, accessible help content or documentation should be available for users who need guidance.

These heuristics are fundamental tools in user interface development, especially in contexts where intuitive operation and error minimization are critical, such as in medical or scientific software. (Nielsen, 1995)

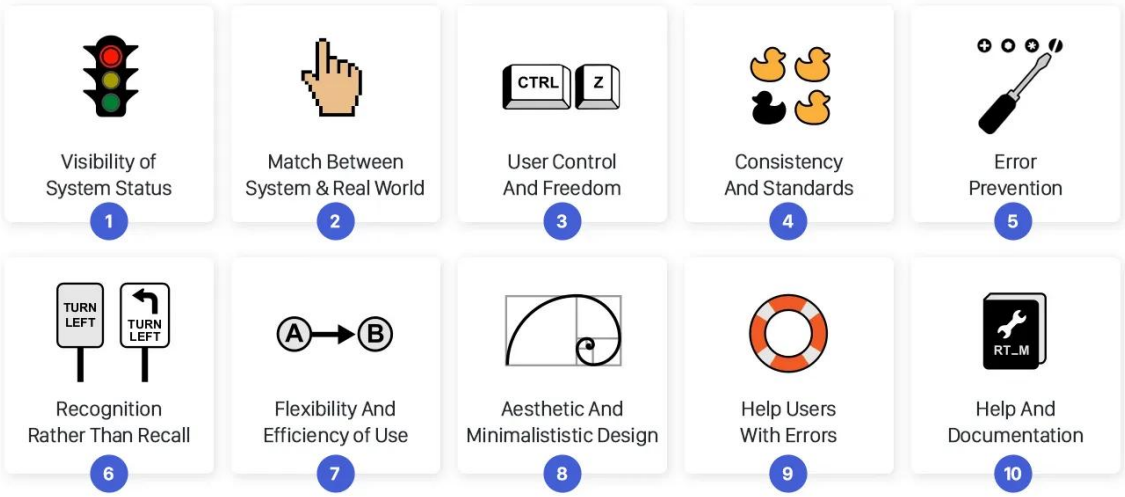


Figure 4 10 usability heuristics

Michal Langmajer, 10 usability heuristics every designer should know. UX Design. Retrieved July 4, 2025, from https://uxdesign.cc/10-usability-heuristics-every-designer-should-know-129b9779ac53

The International Organization for Standardization (ISO) offers a widely recognized framework for designing human-system interactions through its ISO 9241 series. Among its various parts, ISO 9241-11:2018 is specifically dedicated to the concept of usability. This standard defines usability as the extent to which a particular system, service, or product enables specific users to achieve clearly defined goals with a high degree of effectiveness, efficiency, and user satisfaction within a given usage environment.

Effectiveness, in this context, refers to how accurately and completely users can perform intended tasks. Efficiency measures the cost in terms of time, effort, or other resources required to reach those goals, relative to the success achieved. Satisfaction encompasses the user's comfort and subjective approval of the system during and after use.

By outlining these dimensions, ISO 9241-11 does not prescribe rigid design steps but instead provides a structured approach for evaluating and optimizing usability across diverse systems. It encourages developers to ground design decisions in actual user requirements and environmental constraints, thereby ensuring that interfaces are both practical and contextually appropriate.

This standard is particularly valuable in high-stakes domains like healthcare and scientific research, where usability impacts not just productivity but also accuracy, safety, and user trust. Through its emphasis on context-specific performance and user experience, ISO 9241-11 supports

the development of interactive systems that genuinely meet the needs of their intended users. (Standardization, 2018)

3.2.2 Current Trends and Practices in UI Design

Design Systems and Component Libraries

In contemporary user interface (UI) design, the adoption of comprehensive design systems has become pivotal. These systems provide standardized guidelines encompassing visual styles, interaction patterns, and component libraries, ensuring consistency and efficiency across digital products.

Material Design by Google is a prominent example, emphasizing principles such as responsive motion, depth, and adaptive layouts to create intuitive and engaging user experiences. This design system integrates tactile surfaces and natural motions, aiming to make digital interactions feel more physical and grounded. (Google, 2025)

Apple's Human Interface Guidelines (HIG) focus on clarity, deference, and depth, guiding designers to create interfaces that are intuitive and harmonious with the overall user experience on Apple platforms. The HIG emphasizes the importance of content-focused design, where UI elements support the content without overshadowing it. (Apple, 2025)

Microsoft's Fluent Design System introduces elements like light, depth, motion, material, and scale to craft immersive and coherent experiences across devices . Fluent Design aims to create a sense of continuity and fluidity, enhancing user engagement through visually rich and responsive interfaces. (Microsoft, 2025)

These design systems not only streamline the design process but also promote accessibility and inclusivity, ensuring that digital products cater to a diverse user base.

Tools and Frameworks

The evolution of UI design tools has significantly impacted the way designers and developers collaborate and prototype interfaces.

Figma[4] has emerged as a leading browser-based design tool, offering real-time collaboration features that allow multiple stakeholders to work simultaneously on a project. Its cloud-based nature ensures that designs are always up-to-date and accessible from anywhere, facilitating seamless teamwork.

Adobe XD[5] provides a robust platform for designing and prototyping user experiences, integrating features like voice prototyping and auto-animate to create dynamic interactions. Its integration with other Adobe Creative Cloud applications allows for a cohesive workflow for designers familiar with Adobe's ecosystem.

[4] Figma. (2025). Collaborative interface design tool. https://www.figma.com/fr-fr/
[5] Adobe XD. (2025). Adobe XD Platform documentation. https://adobexdplatform.com/

For developers, especially those working with Python, GUI frameworks like Streamlit[6], PyQt[7], and Tkinter[8] offer varying levels of complexity and customization. Streamlit is particularly favored in the data science community for its simplicity and ability to create interactive web applications with minimal code. PyQt provides a comprehensive set of tools for building cross-platform applications with advanced functionalities, while Tkinter serves as a straightforward option for creating basic GUI applications. (Rafalski, 2025)

Usability Testing and UX Research

Usability testing is a critical component in the UI design process, ensuring that interfaces meet user needs and expectations. Various methodologies are employed to gather insights into user behavior and preferences.

Task-based testing involves observing users as they perform specific tasks, identifying any obstacles or confusion encountered during the process. This method provides direct feedback on the usability of particular features or workflows.

The think-aloud protocol encourages users to verbalize their thoughts while interacting with the interface, offering valuable insights into their decision-making processes and highlighting areas of potential improvement.

A/B testing compares two versions of a design to determine which performs better in terms of user engagement or task completion rates. This data-driven approach helps in making informed design decisions.

Standardized surveys like the System Usability Scale (SUS) provide quantifiable measures of usability, enabling designers to benchmark and track improvements over time.

In high-stakes environments such as healthcare and scientific research, rigorous usability testing is paramount. Interfaces in these domains must not only be efficient but also minimize the risk of user error, ensuring safety and reliability in critical applications. (Budiu, 2025) (Soegaard, 2025)

3.3 Video-Based Motion Detection

Motion detection from video sources constitutes a fundamental capability in computer vision, offering practical solutions across disciplines including public security, traffic control, and biomedical research (Sultani et al., 2018). It enables the identification of spatial or temporal variations in pixel data, thereby detecting activity or behavioral changes in recorded environments. In medical and behavioral sciences, especially sleep research, motion detection provides a means to unobtrusively monitor nocturnal activity. This is particularly relevant in the analysis of sleep disorders where abnormal motor behaviors during sleep, such as those seen in parasomnias, are central to diagnosis and therapeutic evaluation.

[6] Streamlit. (2025). Streamlit – The fastest way to build data apps. https://streamlit.io/
[7] The Qt Company. (2025). Qt for Python Documentation. https://doc.qt.io/qtforpython-6/
[8] Python Software Foundation. (2025). Tkinter — Python interface to Tcl/Tk. https://docs.python.org/3/library/tkinter.html

3.3.1 Motion Detection Techniques

In large-scale or data-intensive environments, motion detection systems increasingly rely on machine learning and deep learning architectures to interpret complex spatiotemporal patterns. These methods, often involving convolutional neural networks (CNNs) or recurrent architectures, are trained on massive annotated video datasets to automatically recognize motion-related events or behaviors. Their ability to adapt to diverse lighting conditions, occlusions, and background noise makes them particularly powerful in security, autonomous systems, and large-scale video analytics. However, such systems are often resource-intensive, requiring substantial computational power, labeled data, and ongoing tuning, which may be excessive for targeted clinical or research applications where interpretability and efficiency are prioritized. (Sultani et al., 2018)

In smaller-scale contexts such as clinical video annotation or behavior monitoring where scenes are controlled, cameras are static, and performance must be real-time lighter algorithmic approaches remain more appropriate. This is where OpenCV, an open-source computer vision library for Python, proves especially effective. OpenCV offers a well-maintained suite of motion detection tools optimized for low-latency and real-time applications, making it a practical standard in academic and applied research projects. (Open Source Vision Foundation, 2025)

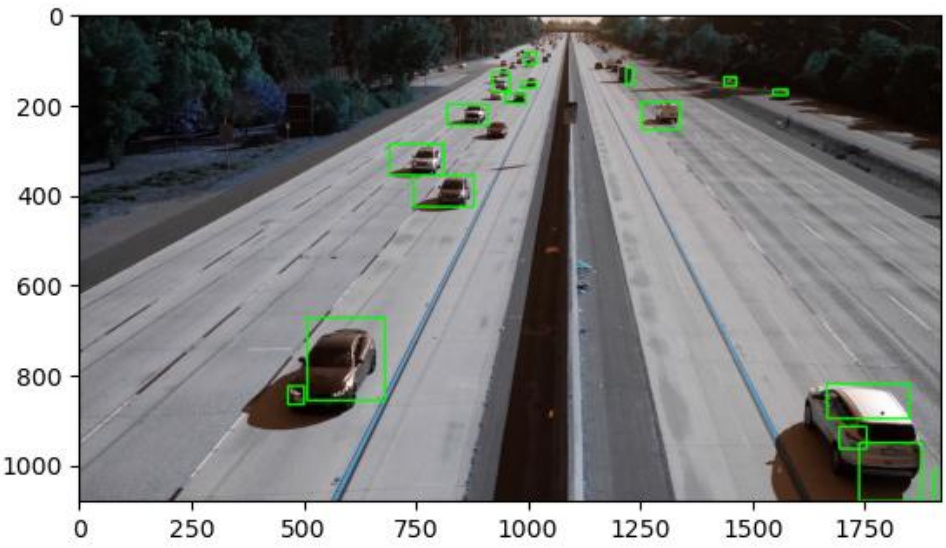


Figure 5 Detection with Frame Differencing

itberrios. (n.d.). detection_with_frame_differencing.ipynb. GitHub. Retrieved July 4, 2025, from https://github.com/itberrios/CV_projects/blob/main/motion_detection/detection_with_frame_differencing.ipynb

Among the motion detection techniques available in OpenCV, three primary categories can be distinguished: background subtraction, optical flow, and frame differencing. Background subtraction maintains a reference model of the scene often an average or probabilistic distribution of previous frames and identifies motion as deviations from this model. It is robust under stable environmental conditions and is implemented in OpenCV through algorithms such as MOG2 and KNN. Optical flow

methods, such as Farneback or Lucas-Kanade, estimate pixel-wise motion vectors by analyzing the temporal evolution of brightness patterns. These are ideal for capturing subtle or continuous motion but impose a significantly higher computational load.

The method implemented in the current system is frame differencing, which relies on computing the pixel-wise absolute difference between consecutive grayscale frames, typically after applying a Gaussian blur to reduce noise. When the difference exceeds a threshold and forms contours of significant size, motion is registered and logged. This approach offers an optimal trade-off between simplicity and performance, particularly in structured environments like sleep laboratories where cameras are static, lighting is constant, and responsiveness is key. Its implementation using OpenCV requires minimal resources and integrates seamlessly with timestamp-based logging and contour analysis, as demonstrated in our application. (Rosebrock, 2025) (Bradski, 2000)

3.3.2 Software and Libraries

The implementation of motion detection systems relies heavily on robust and well-supported software tools. In the context of this project, the software ecosystem was built around Python, leveraging its extensive scientific computing libraries for video processing, data handling, and analysis. Among these, OpenCV[9] (Open Source Computer Vision Library) serves as the central framework for low-level video operations, providing efficient routines for frame capture, grayscale transformation, spatial filtering, and motion contour detection. Its ability to directly manipulate pixel matrices and apply image-processing algorithms in real-time makes it particularly suited to iterative prototyping and experimentation.$

Unlike proprietary platforms, OpenCV's open-source model ensures full transparency over algorithm behavior and reproducibility of results, an essential criterion in research environments. The library also supports interoperability with a wide range of Python modules, which facilitates rapid integration with higher-level data processing tools.

To complement OpenCV, the project also utilized NumPy for optimized array computation and Pandas for structured data manipulation. These libraries enable precise control over motion analysis outputs, such as organizing detection logs, computing frame indices, and exporting event metadata into tabular formats for downstream statistical evaluation. This layered software architecture allows motion data to be processed in real-time and later subjected to systematic review, annotation, or validation steps using interactive graphical interfaces or external analytics platforms.

Collectively, this combination of open-source libraries forms a modular and scalable software stack that balances performance with clarity qualities that are particularly advantageous in academic

[9] Open Source Vision Foundation. (2025). OpenCV 4.x Documentation. https://docs.opencv.org/4.x/

and clinical research scenarios where traceability, automation, and maintainability are crucial. (Open Source Vision Foundation, 2025)

3.3.3 Benefits and Limitations

Using motion detection in sleep studies offers clear advantages for both researchers and participants. Because it relies on video recordings rather than physical sensors, it helps preserve the natural sleep environment and avoids interfering with the participant's behavior. It also makes the analysis process more efficient by automatically highlighting periods of activity, which reduces the amount of time researchers spend manually watching entire video recordings.

However, some limitations and responsibilities come with this approach. Technical factors such as poor lighting, awkward camera angles, or low video resolution can reduce the accuracy of motion detection. It's important to make sure that recording conditions are carefully set up so that the software can detect relevant movements reliably.

In addition, working with video recordings, especially in clinical or sensitive settings, raises important ethical questions. Participants must give informed consent, and the data must be handled with care to protect privacy. Regulations in Switzerland, for example, require strict guidelines when handling personal video data, especially in a medical context.

By combining thoughtful technical setup with responsible data handling, motion detection can be a valuable tool in behavioral research while respecting ethical standards. (Confédération suisse, 2011)

4. Methodology

4.1 Current State of the Project

4.1.1 Overview of the Research Study

The research project at the heart of this work is a clinical investigation led by the CHUV's Center for Integrative and Complementary Medicine (CEMIC), in collaboration with the Sleep Investigation and Research Center (CIRS). The study is designed to evaluate the effectiveness of a non-pharmacological treatment, namely hypnosis, on reducing parasomniac episodes in individuals diagnosed with non-rapid eye movement (NREM) parasomnia. This condition includes sleep disorders such as sleepwalking, sleep terrors, and confusional arousals.

The study compares two groups of patients: one receiving hypnosis-based mental imagery treatment (HYP), and the other receiving standard care involving sleep hygiene and safety education (EDUC). The central hypothesis is that the hypnosis intervention, particularly when tailored with specific mental imagery techniques, will result in a measurable reduction in the frequency of parasomniac episodes compared to the control condition.

To measure this effect, the primary endpoint of the study is defined as the average number of parasomniac episodes per night, recorded over 10 consecutive nights both before and after the treatment period (approximately four weeks apart). These episodes are not measured via subjective reports, but through systematic scoring of infrared video recordings taken in the participants' home environments. Two trained scorers independently evaluate these recordings to ensure reliability and accuracy, under the supervision of a sleep specialist.

Participants are randomly assigned in equal proportions (1:1) to either the HYP or EDUC group, using a computer-generated randomization list. The allocation is concealed in sealed envelopes, which are opened sequentially upon inclusion of each participant.

The clinical protocol is composed of five on-site visits and two virtual follow-ups:

- Visit 1 (Baseline): Informed consent is obtained, eligibility criteria are verified, and an EEG recording is performed during neutral mental imagery tasks, similar to the procedure used at Visit 5. Participants are randomized at the end of this session.
- Visits 2–4 (Treatment Sessions): These visits include either hypnosis sessions for the HYP group or educational interventions for the EDUC group. The hypnosis sessions involve:
    - V2: A guided safe-place imagery session.
    - V3: A session focused on rewriting parasomnia-related mental imagery.

- V4: Techniques for modulation and control of dissociative states.

In parallel, participants in the EDUC group receive personalized guidance on sleep hygiene, environmental control, and risk mitigation strategies. All treatment sessions are conducted by the same physician and last between 45 to 60 minutes.

- Visit 5 (Follow-Up): This session is used to collect post-treatment clinical parameters, conduct an EEG recording during a neutral imagery task, and optionally perform a post-treatment polysomnography (PSG).

The timeline for the treatment phase is structured as follows:

- V2 occurs 15 days (±7) after V1,
- V3 occurs 7 days (±7) after V2,
- V4 occurs 21 days (±7) after V3.

As part of the protocol, video recording plays a central role. Patients are instructed on how to correctly position the camera in their sleeping environment and how to start and stop the recordings according to their nightly routines. The camera, configured to record during specific nighttime intervals, stores footage locally before the data is transferred to CHUV's secure servers. Participants are given the opportunity to request deletion of selected timeframes prior to processing, in order to preserve privacy and comply with ethical guidelines.

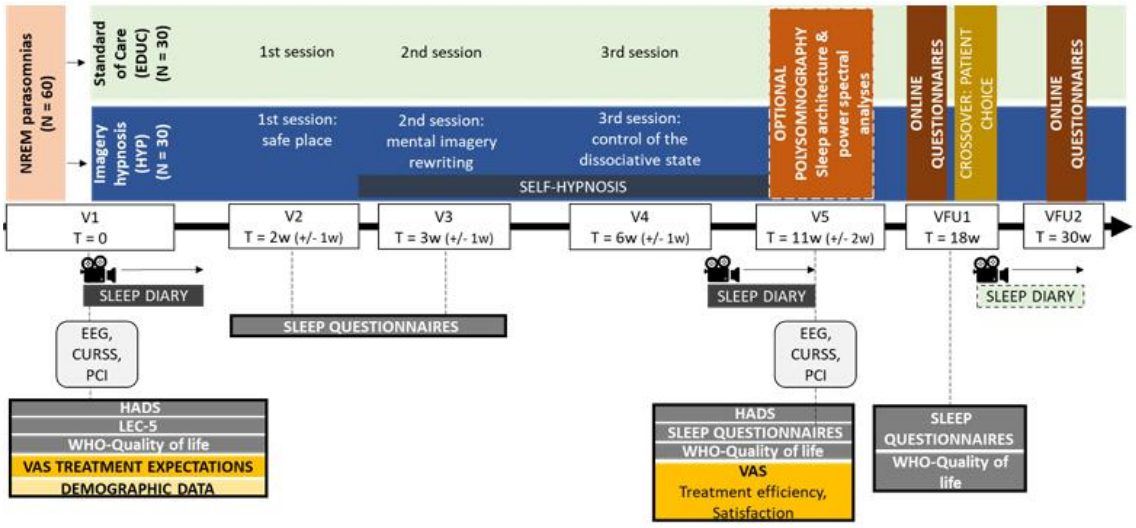


Figure 6 Design of the study

CHUV. (2024). Study protocol for parasomnia video scoring project [Confidential internal document].

The recorded videos are reviewed to identify parasomniac episodes, classify their subtype (e.g., confusional arousals, sleep terrors, sleepwalking, or undetermined), and measure their duration.

Initially, two trained scorers conduct a joint homogenization phase to align their scoring criteria. Afterwards, each scorer independently annotates the recordings. In the event of discrepancies between the two assessments, an initial discussion phase is held between the two scorers to attempt resolving their discordances collaboratively. If they are unable to reach an agreement, a third sleep expert is then called upon to adjudicate the scoring for that night.

This manual procedure, while effective in ensuring expert control, has proven to be highly labor-intensive and unsustainable given the volume of video data produced across multiple nights. The lack of integrated tools forces scorers to rely on generic video players and external files for annotations, leading to fragmented workflows and limited traceability. These constraints have highlighted the need for a dedicated software solution capable of streamlining video review, structuring annotations, and assisting scorers through partial automation. This need is particularly relevant considering the growing size of the dataset and the clinical importance of timely, accurate analysis.

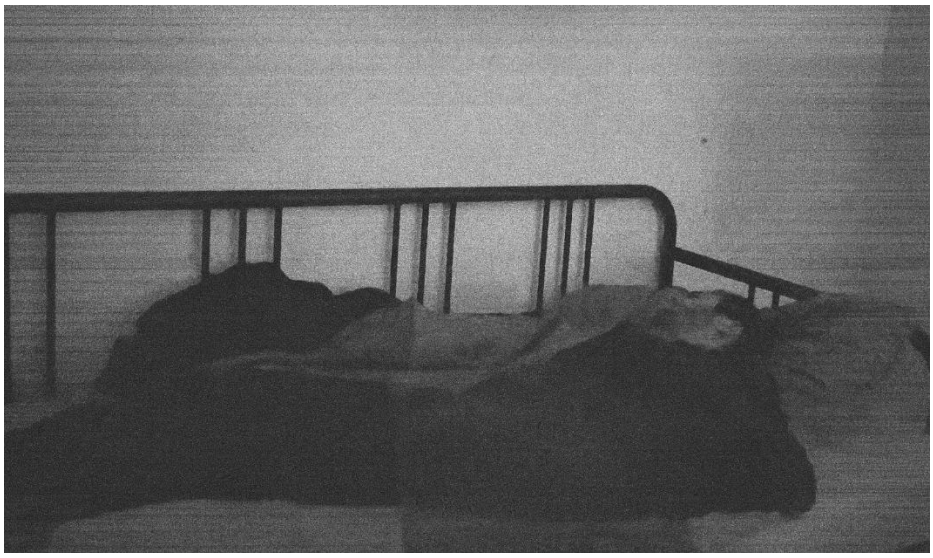


Figure 7 Example video recorded

(Source: Author)

4.1.2 Manual Scoring Workflow

Before the development of any automated system, the identification and annotation of parasomniac episodes were carried out entirely by human scorers. Once the video recordings were exported from the patients' home devices to CHUV's secure servers, two trained evaluators were tasked with reviewing the footage manually. Using conventional video playback tools, they would watch each night's recording, often several hours long, in real time.

During this review, scorers noted the start and end times of each observed episode, assigned a category to the event, and often included written observations or clarifications. All data was entered into structured Excel sheets or plain text formats, entirely by hand.

To ensure consistency in scoring, the two scorers worked independently, reviewing and

annotating the recordings on their own. After several nights had been scored separately, their results were compared to identify any discordances in the number, type, or duration of detected episodes. When disagreements were found, they were discussed and resolved collaboratively, or, if needed, referred to a third sleep expert for final arbitration.

Although this method offers expert-level granularity, it quickly revealed critical limitations:

- Time consumption: Reviewing multiple nights per patient at full length was unsustainable for the team.
- Risk of human error: Manual timestamping and classification left room for inconsistencies or omissions.
- Data fragmentation: With video playback and annotation handled in separate tools, it was difficult to maintain traceability or quickly verify annotations.

These challenges underscored the importance of creating a more efficient, unified interface that could centralize playback, annotation, and data management. This was the starting point for the development of a dedicated motion detection and scoring system.

4.1.3 Existing Python Script

As part of the preliminary development phase, two basic Python scripts were provided to support the motion detection and scoring process. These scripts were assembled from earlier independent experiments and adapted to the current study's context. While they offer a useful starting point, they remain highly limited in functionality and usability.

The scoring_vid_N.py script uses OpenCV to perform basic motion detection on a single video file. The approach relies on a simple frame differencing method with Gaussian blurring to reduce noise, followed by contour detection to identify significant movements. The user is prompted via the command line to enter the start and stop timestamps (hour, minute, second) for the segment they wish to analyze. Once launched, the script processes the video frame-by-frame, skipping intervals of approximately two seconds, and identifies potential motion by comparing each frame to the previous one.

When motion is detected (based on contour area size), the script records the corresponding frame number and timestamp. This information is saved in a .csv file and stored in a local output folder. A basic real-time visual preview is displayed using OpenCV windows, showing both the original video with bounding boxes and the thresholded motion detection mask.

While the core functionality operates as intended, the script suffers from several critical limitations:

- Command-line only: There is no graphical user interface, which makes the tool

inaccessible for non-technical users such as clinicians or research assistants.

- Single-file processing: The script can only process one video at a time, requiring manual input for each file. There is no support for batch analysis or structured input (e.g., a CSV file listing all start and stop times).
- Unstructured output: The results are saved as raw CSV files without formatting, making them difficult to interpret or use directly in clinical workflows. The absence of Excel formatting, column validation, or summary structure further limits the script's utility.
- Lack of metadata integration: The output does not include essential contextual information such as participant ID, night number, or episode type data which is crucial for traceability and aggregation in a clinical setting.

As it stands, this script functions more as a proof of concept than a usable clinical tool. Its value lies in confirming the feasibility of motion detection on the provided video material. However, in order to support real-world workflows, especially in a hospital or research environment, it must be significantly extended.

In addition to the motion detection script, a second prototype was provided to support the manual annotation of parasomniac episodes. The discordances_check.py script, built using Python's Tkinter framework, offers a minimal graphical interface aimed at facilitating the entry of scoring data. The goal is to allow scorers to input structured information about each episode such as start time, end time, intensity, type, and trigger into a local CSV file.

The interface includes several forms to collect scorer initials, participant ID, recording period, and episode-specific parameters. The scorer can manually enter start and end times (formatted in hh:mm:ss), select episode characteristics from dropdown menus, and add a free-text description. The application then stores this information in a local output file (entries.csv) and includes logic to detect conflicting entries based on participant, night, and start time. In such cases, it prompts the user to confirm whether to overwrite the existing data.

A useful feature of this script is the discordance checking function, which compares annotations from multiple scorers and highlights differences in parameters such as duration, type, or intensity. It also proposes corrections by allowing the user to merge or edit entries through a dedicated correction window. This aims to partially support the scoring arbitration process, which is essential when multiple scorers evaluate the same video segment.

Despite these useful elements, the script remains a very early-stage prototype with significant functional limitations:

- No video integration: There is no built-in video playback or synchronization between annotations and the video content. Scorers must refer to external players to determine timestamps.

- No detection visualization: The interface does not show detected motion events or facilitate review of previously flagged segments.
- Manual-only annotation: All data must be entered manually by the user. There is no import or pre-fill mechanism using detection data.
- Partial compliance logic: While some rules are enforced (e.g., unique combinations of participant and start time), others like time overlaps, episode duration limits, or scorer-specific access control, are missing or only superficially implemented.

In its current state, the interface provides a structured but narrow framework for episode annotation. Its development validates the basic structure of a scoring pipeline but leaves many critical user and system needs unmet.

4.1.4 Clinical Constraints and Ethical Context

Given the clinical nature of the study and the involvement of human participants, the development of software tools for video analysis must adhere to strict ethical and operational standards. All video recordings used in this project were collected under an approved clinical protocol by CHUV, in compliance with Swiss data protection regulations and internal research governance policies. (Confédération suisse, 2020)

The primary ethical concern lies in the use of video recordings taken in patients' private home environments. These recordings may contain sensitive visual information, and as such, must be treated with the highest level of confidentiality. Participants are informed in advance of the recording procedure and are given the opportunity to request the deletion of specific time segments before data is transferred to CHUV's secure infrastructure. This ensures that patient autonomy and privacy are respected throughout the process.

From a technical standpoint, the software developed in this project is intended to be used exclusively within the CHUV research ecosystem, on pre-anonymized data. All storage and processing are performed on local or secured hospital servers, with no cloud transmission or external access permitted. The system does not collect or transmit any personal identifiers; all references to participants are managed using coded identifiers defined by the clinical team.

In terms of design, the application must support traceability, consistency, and reproducibility of annotations. This includes proper timestamp formatting, scorer identification, and standardized data exports for clinical audit trails. Usability also plays a role in ethics: the software must reduce the risk of scorer fatigue or error by providing a clear interface, structured forms, and validation feedback.

Finally, the tool must be flexible enough to integrate into existing workflows without compromising clinical procedures or introducing undue complexity. For this reason, the application prioritizes transparency in its logic and output structure, ensuring that expert users maintain full control over the data they review and validate.

4.2 Planned Improvements

Following the review of existing tools and workflows (Section 4.1), it became clear that a more structured and user-friendly solution was needed to support the scoring of parasomniac episodes. The manual annotation process, while thorough, is highly time-consuming and fragmented. The provided scripts, although functional as proofs of concept, lack integration, user-friendliness, and the ability to scale to a larger dataset.

The planned improvements focus on the development of an integrated video scoring platform that enables real-time video playback, intuitive annotation workflows, and structured data export. To support this goal, a critical first step involved the selection of a suitable GUI framework that could meet the technical and usability requirements of a clinical research environment.

4.2.1 GUI Framework Evaluation and Justification

In the initial phase of the project, the objective was to identify a suitable graphical user interface (GUI) framework to support the development of a clinical video annotation tool. The intended application involves reviewing video footage recorded in a medical setting, marking events detected via motion analysis, and enabling expert users to validate or reject those events through an interactive labeling interface. The software had to support video playback custom timeline interaction, user-triggered annotations, and export of structured data.

The selection criteria included (1) support for multimedia playback and control, (2) flexibility in building interactive widgets such as sliders and lists, (3) the ability to integrate with Python-based analysis backends, and (4) capacity for creating a professional and usable interface suitable for non-technical users in a clinical environment.

4.2.2 Initial Exploration and Prototype Development

The evaluation began with an exploratory implementation using Streamlit, a Python-based framework primarily oriented toward the rapid development of web applications for data exploration and visualization. Streamlit enables the creation of user interfaces using a high-level declarative syntax and integrates seamlessly with popular data science tools.

However, the exploration revealed that Streamlit's architecture is not designed for complex, real-time multimedia interaction. Video playback is limited to basic HTML5 rendering, with no support for advanced controls such as synchronized triggers, interactive timeline markers, or frame-accurate navigation. Furthermore, the inability to dynamically manipulate interface components outside of a page-reload cycle hindered the creation of a smooth and responsive user experience.

Based on these limitations, attention shifted toward PyQt5, a well-established Python binding for the Qt framework. A prototype was developed using PyQt5, this time focusing on desktop-based interaction.

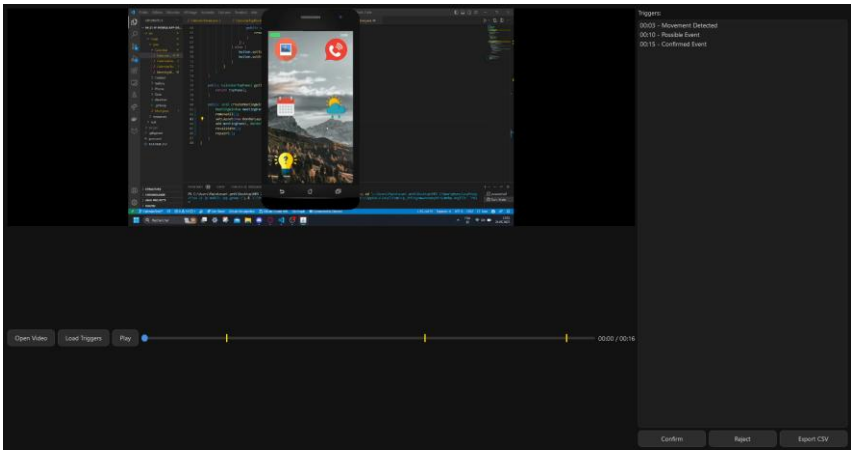


Figure 8 First prototype in PyQt5

(Source: Author)

Although the initial prototype was simplified and not directly aligned with the final clinical use case, it served to evaluate core functionalities including layout control, widget customization, video integration via the VLC backend, and the rendering of timeline-based trigger markers. This investigation confirmed the feasibility of implementing a feature-rich application with a responsive and customizable interface using PyQt5.

In addition, Tkinter was also considered during this evaluation. As Python's standard GUI library, it offers the advantage of simplicity and broad compatibility. However, it was ultimately not selected because its styling capabilities are outdated and it lacks robust support for advanced multimedia features such as integrated video playback with custom sliders or synchronized event triggers. Implementing such features in Tkinter would have required complex, low-level workarounds unsuitable for developing a modern, user-friendly clinical tool.

4.2.3 Comparative Analysis of Frameworks

To support the decision-making process, a comparative analysis of three GUI frameworks—Streamlit, PyQt5, and Tkinter was conducted. The analysis focused on technical criteria relevant to the specific demands of the project.

Feature	Streamlit	PyQt5	Tkinter
Application type	Web-based (browser)	Desktop-native	Desktop-native
Multimedia playback support	Limited (HTML5, no timeline control)	Full support via VLC	Not natively supported
Timeline and custom slider support	Not supported	Fully supported (custom widgets)	Possible but complex to implement
User interaction complexity	Simple inputs only (forms, buttons)	Rich interaction with full widget set	Limited interactivity, low-level
Styling and visual customization	Limited (HTML/CSS only)	Extensive via QSS	Very limited, use of other versions
Learning curve and documentation	Minimal	Moderate	Basic, but outdated in some cases

Table 1: Framework Comparison: Streamlit, PyQt5, and Tkinter

Table created by the author.

The decision to proceed with PyQt5 was based on a combination of technical capabilities and practical considerations. PyQt5 provides access to an extensive widget toolkit, including the ability to build custom slider components, integrate with external video backends (such as VLC), and create complex multi-pane interfaces. Its support for Qt Style Sheets (QSS) allows for the design of professional-grade user interfaces, a critical feature given the tool's intended deployment in a clinical research context.

Moreover, early experimentation demonstrated that PyQt5 could effectively support the project's advanced requirements, including:

- Real-time video playback with .mkv and .mp4 formats
- Visualization of motion triggers as timeline markers
- Trigger-by-trigger validation via interactive buttons
- Data export to structured formats (e.g., CSV)

This confirmed PyQt5 as a technically robust and strategically appropriate choice for the development of the video annotation interface.

4.2.4 Functional Objectives of the Final Application

To meet the requirements identified during the evaluation phase, the final software solution will integrate multiple components into a unified, user-friendly interface. The tool will be designed as a desktop application built with PyQt5, offering a structured workflow for both motion detection and manual scoring of parasomniac episodes. The following features are planned:

Main Interface

- Landing page with clear navigation between core features:
  - Motion Detection
  - Video Scoring
- Clean, intuitive, and professional visual design adapted for clinical use.

Motion Detection (Single Video)

- Select a single video file for detection.
- Input fields for start time and end time of the analysis.
- Adjustable detection parameters (e.g., sensitivity, blur, frame skip).
- Console or on-screen logging of detection events and status.

Batch Processing

- Dedicated tab for batch motion detection.
- Upload of an Excel file listing video names and start7end times for each segment.
- Automatic processing of multiple videos with result summaries.

Video Scoring Interface

- Embedded video player for direct in-app visualization.
- Interactive timeline with a draggable slider for navigation.
- Volume control and playback options.
- Keyboard shortcuts for efficiency in video analysis.
- Option to automatically load motion detection results based on the selected video.
- Display of detection segments in a list for quick access and scoring.

Scoring Form

- Dynamic form that adapts based on validation status (detailed fields for correct segments, minimal input for incorrect).
- Key fields for scorer, participant details, detection times, validation status, episode type, intensity, and description.
- Automatic creation of a video clip from the scored segment if it does not already exist.

Data Export and Storage

- Export of all detection and scoring data into a single, well-formatted Excel file.
- Human-readable layout with proper column names, time formatting, and grouping by scorer, participant, and night.
- Files stored in organized folders for traceability and ease of use.

Conflict Resolution

- Conflict-checking tool to identify discordances between scorers.
- Highlights mismatched entries (e.g., different episode types or timings).
- Enables users to resolve discordances manually and save corrected versions.

Visual and UX Design

- Responsive and modern UI using PyQt5 with custom styling.
- Organized layouts, icons, consistent fonts, and color coding to enhance clarity and reduce cognitive load.

4.3 Design Considerations for Integration

To ensure long-term usability, transparency, and adaptability, the development of the clinical video annotation tool is guided by several design principles. These considerations aim to support both the immediate project goals and future adaptations, whether by technical staff or research collaborators with limited programming experience.

A key expectation for such a tool is to maintain a clean separation between the user interface

and the application logic. This allows the interface, built for clinicians or researchers, to evolve independently of the motion detection or data processing modules. Structuring the code this way improves maintainability and facilitates the integration of additional features, such as new detection methods or scoring workflows, without rewriting existing components.

Rather than relying on complex design patterns, the logic should remain as modular and explicit as possible. For instance, introducing a new motion detection algorithm should ideally require modifying or adding a single, self-contained file. This approach helps keep the learning curve low for contributors who may not have a software engineering background.

The configuration of detection parameters, thresholds, and processing behavior should be managed through a dedicated configuration file (e.g., JSON format). This allows the application to remain flexible and adaptable without requiring users to modify the underlying code. Adjusting detection sensitivity or switching between detection strategies should be a matter of editing well-documented values in this file.

From a structural perspective, the application is expected to follow a logical directory organization, with folders separating the interface components, core logic (e.g., motion analysis, scoring utilities), and output data. This helps maintain clarity and encourages consistent development practices.

To support reproducibility and ease of deployment, the application should run within a Python virtual environment. This ensures that dependencies are isolated and controlled, and that the software can be installed or migrated reliably across different systems using a requirements.txt file. Such an environment is particularly important in research settings where tool longevity and compatibility matter.

Overall, the architecture should reflect a balance between simplicity and extensibility: simple enough to be approachable and maintainable but structured enough to support the evolving needs of a clinical study. While some of these structural elements have already been explored and partially implemented, they represent guiding principles for the full development of the tool.

4.4 User Collaboration and Testing

An essential part of the development methodology for this project was ongoing collaboration with the main stakeholder at CHUV, who represents the primary user and clinical expert for the application. This ensured that the design and features of the tool were closely aligned with real clinical needs and workflows.

At the outset of the project, an in-person visit to the CHUV provided an opportunity to discuss the overall goals of the study, review existing manual scoring practices, and clarify the requirements for the planned software. During this visit, detailed conversations covered essential aspects such as the necessary features for motion detection and scoring, the logical folder structure to facilitate data management, and the preferred organization of the application's user interface.

Throughout development, regular meetings and discussions were held to maintain alignment with these requirements. These interactions allowed the stakeholder to clarify expectations, prioritize features, and provide critical input on workflow design. This continuous feedback loop was essential in shaping the planned functionality of the application, ensuring it would be usable and relevant in the clinical context.

To validate the practical usability of the tool, informal testing phases were also conducted during development. The application was installed and tested on CHUV computer to check compatibility and to identify and resolve environment-specific issues. In addition, the stakeholder was given the opportunity to test the application directly, exploring its features independently and providing feedback on both functionality and interface design. This feedback informed several refinements, such as clarifying form layouts, adjusting detection parameter settings, and improving navigation flow.

By involving the primary user throughout the project, the development process embraced user-centered design principles, aiming to deliver a solution that meets the clinical team's needs while remaining straightforward to maintain and extend in future research projects. This collaborative approach also helped ensure that the final software design would be realistic, practical, and suited to integration into the existing research environment at CHUV.

4.5 Development Tools and Workflow

To support the development of the clinical video scoring application, a set of established software tools and workflows was adopted. The choice of tools was guided by the need to maintain code quality, ensure version control, facilitate clear communication with stakeholders, and promote reproducibility within a clinical research environment.

The main tools and practices included:

- Visual Studio Code (VSCode): Used as the primary development environment, offering strong Python support, integrated terminal, linting, and extensions for efficient navigation and debugging. It was especially helpful for managing the project's multiple components, including the GUI, detection logic, and data export routines.

- Git and GitHub: Version control was handled using Git, with GitHub serving as the remote repository for secure backups and change tracking. The GitHub Desktop app was used to simplify the management of commits and pushes, providing a more visual and accessible interface. For branching strategy, the project employed a main-only approach suitable for solo development. However, the repository structure and commit history were maintained in a way that would support feature branching in the future, if multiple contributors were to be involved.

- Microsoft Excel: Used in two key ways:
  - Product Backlog Management: A spreadsheet tracked planned features, priority levels, and development progress to organize tasks systematically.
  - Meeting Tracker: This tracker ensured that requirements discussions and stakeholder decisions were documented, traceable, and accessible throughout development.

- Microsoft Teams: Employed for direct communication with the main clinical stakeholder, enabling messaging, file sharing, and meeting scheduling. Teams supported rapid feedback cycles and maintained alignment with clinical requirements during development.

- Python Virtual Environments: The application was designed to run within a dedicated virtual environment. This approach ensured that all dependencies (defined in requirements.txt) were isolated from the system's global packages, promoting consistent behavior across different installations and simplifying deployment within the hospital environment.

By using these tools and practices, the project aimed to maintain a professional, reproducible, and well-documented development process, while ensuring accessibility and usability for a clinical research context involving interdisciplinary teams.

Additionally, part of the development was informed by adapting code from an existing open-source repository, which provided a basic implementation of frame differencing for motion detection in Python.

This repository served as a starting point for developing the detection logic, which was then significantly expanded, refactored, and integrated into the broader application to meet clinical research requirements. (itberrios, s.d.)

5. Results

5.1 Solution Overview

This section provides an overview of the software solution developed to support the analysis and scoring of parasomniac episodes in clinical video recordings. The primary objective of the tool is to streamline a process that was previously fully manual, by offering integrated motion detection capabilities and a user-friendly interface for expert scoring and annotation.

The application was designed to address specific needs identified in collaboration with clinical stakeholders, including the ability to process home-recorded sleep videos, detect potential movement events, and enable scorers to efficiently review, validate, and annotate these events. By integrating these functions into a single platform, the software aims to reduce the time and effort required for manual review, improve consistency between scorers, and produce structured, well-formatted outputs suitable for clinical research.

The following subsections describe the overall concept behind the solution, provide an overview of the user interface and workflows, and highlight the main features and components developed during the project.

5.1.1 General Concept

The main concept behind the developed application is to provide a centralized, user-friendly platform for the detection and scoring of parasomniac episodes captured in clinical video recordings. Traditionally, this process has relied on fully manual review: scorers watch hours of infrared footage, identify events of interest, and record start and end times, types, and other metadata entirely by hand. This approach is both time-consuming and prone to variability between scorers.

According to the initial project description in the bachelor thesis data sheet, the planned scope was limited to developing a user interface that would connect to and use an existing motion detection application essentially focusing only on supporting the scoring part. However, after detailed discussions with stakeholders at CHUV, it became clear that the existing resources were extremely limited, consisting only of very simple exploratory scripts without any production-ready logic or integration.

As a result, the scope of the project expanded significantly. It was necessary not only to design the user interface for scoring but also to develop the entire underlying logic from scratch, including motion detection routines, batch processing capability, data management, and exporting features. This ensured that the final tool would be genuinely usable in a clinical research context, supporting the full workflow from raw video input to structured, analyzed output.

The software is thus structured around two tightly integrated but clearly defined modules:

- Motion Detection Module: Automates the identification of movement events in video footage. It applies frame-based analysis to detect periods of significant motion, generates logs of these events with timestamps, and allows for both single-video processing and batch analysis using preconfigured schedules.
- Video Scoring Module: Provides scorers with an interactive interface to review recorded videos, navigate to detected segments, validate or correct motion events, and annotate episodes with standardized metadata such as type, intensity, and triggers. When a form is saved, the system also automatically creates a video clip of the annotated segment if it does not already exist. This module also supports exporting all annotations in structured Excel files optimized for clinical research and subsequent analysis.

The overarching goal is to streamline and standardize the scoring workflow, reducing manual workload while improving accuracy, traceability, and data quality.

5.1.2 User Interface Overview

This subsection presents the visual layout and design choices made for the application's user interface. The goal was to create an intuitive, professional-looking desktop application that would be accessible to clinicians and researchers with varying levels of technical expertise.

A major design priority was to clearly separate the two main functional areas of the tool: motion detection and video scoring. This separation is reflected in the interface structure, which guides users logically through the steps of detection, review, and annotation without unnecessary complexity. Design considerations focused on ensuring that all controls and forms are clearly labeled and grouped by function, maintaining consistent styling and color schemes to convey a professional, research-grade appearance, and providing clean layouts with minimal visual clutter to reduce cognitive load during long scoring sessions. Attention was also given to including feedback elements such as logs and confirmations to build user confidence and support error reduction.

The following images and captions illustrate the application's main screens and design elements, explaining why specific interface decisions were made to improve usability and align with the clinical workflow requirements.

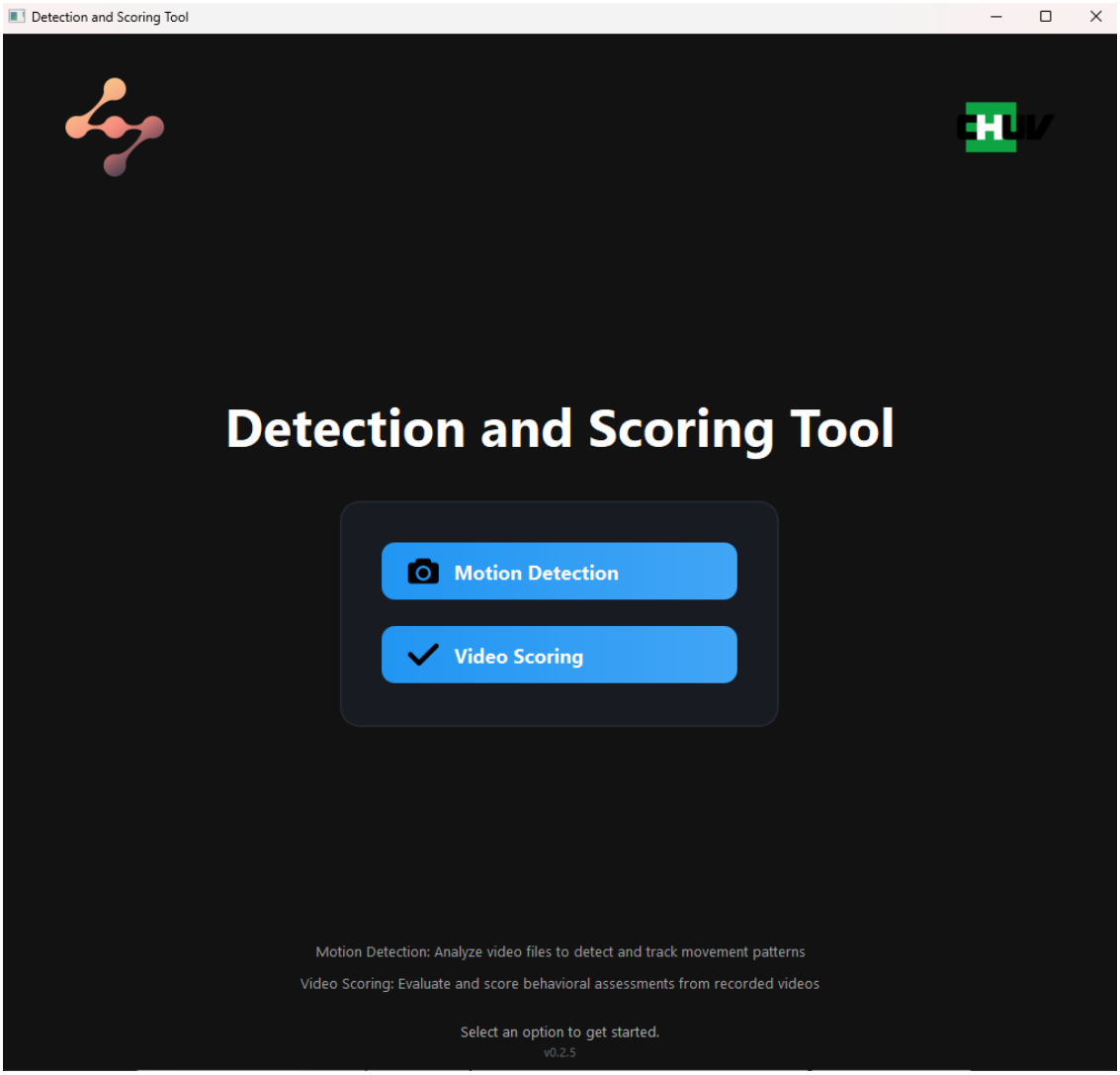


Figure 9 Landing Page of the Detection and Scoring Tool

(Source: Author)

The landing page of the application serves as a clear and focused entry point, separating the two main functional areas, Motion Detection and Video Scoring, into distinct, visually prominent buttons. This design choice ensures users can immediately recognize the scope of the application and select their desired workflow without confusion.

The inclusion of the CHUV and Sense logos reinforces institutional identity and signals that the tool is intended for professional clinical use. The screen layout follows Shneiderman's "Consistency and Standards" heuristic by using uniform button styles, colors, and typography to communicate function clearly and reduce ambiguity. Additionally, the short explanatory text under each button supports "Recognition Rather Than Recall," guiding users to understand each section's purpose at a glance without requiring them to remember internal details or previous instructions.

By offering a simple, well-organized starting point, the landing page reduces cognitive load and establishes a professional, user-friendly introduction to the application, setting expectations for the clear, guided workflows that follow in the tool.

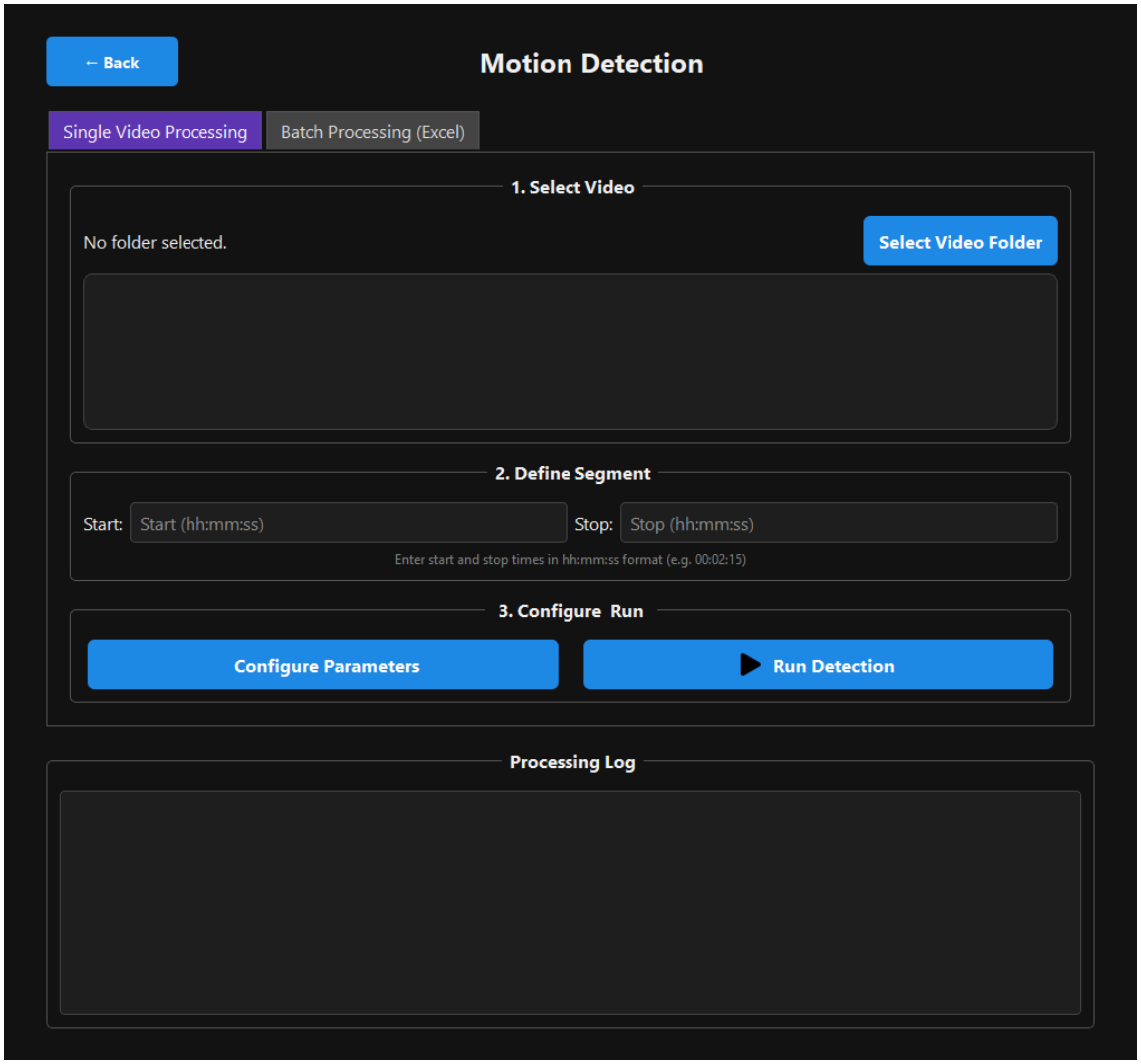


Figure 10 Motion Detection – Single Video Processing Interface

(Source: Author)

This screen demonstrates the design of the single video processing tab within the Motion Detection module. The layout is structured into clear, numbered sections: Select Video, Define Segment, Configure and Run, to guide the user through each step in an intuitive, sequential manner. This approach follows Shneiderman's "Design Dialogs to Yield Closure" heuristic, ensuring users always know where they are in the process and what the next action will be.

The interface also separates configuration from execution, encouraging careful parameter review before running detection. Button labels and form placeholders use plain, descriptive language to minimize confusion, supporting Recognition Rather Than Recall by making expected inputs (like start and stop times in hh:mm:ss format) visible at the point of use.

Additionally, if a video has already been analyzed before, the application prompts the user with a confirmation pop-up that compares the existing analysis configuration to the current parameters. This message explains whether the settings match or differ, allowing users to make an informed

choice about whether to continue and re-run the detection.

The design also includes a Processing Log area to provide real-time feedback during detection, increasing transparency and user confidence in the system's operations. This structured and clearly labeled interface aims to reduce errors and support clinicians in performing accurate, repeatable analyses without unnecessary technical complexity.

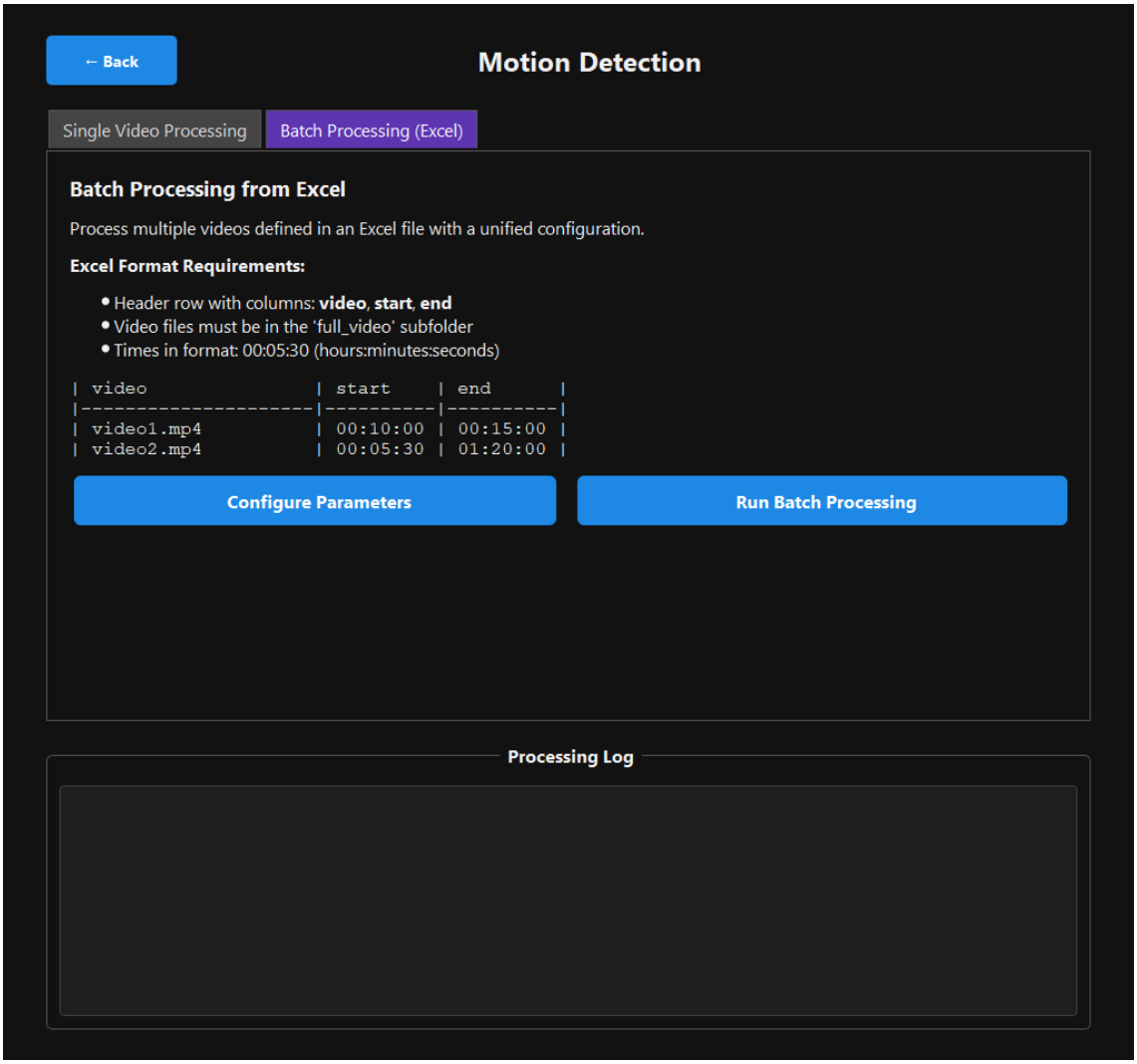


Figure 11 Motion Detection – Batch Processing Interface

(Source: Author)

This figure shows the Batch Processing tab within the Motion Detection module, designed to enable efficient, large-scale analysis of multiple video files in a single run. The interface clearly explains the required Excel format, specifying column headers (video, start, end) and time formatting standards. By providing a direct example within the screen, the design supports Recognition Rather Than Recall, helping users understand how to structure their input data without consulting external documentation.

The batch processing option was added in response to the clinical need to analyze entire folders

of videos, reducing repetitive manual work and the potential for input errors when handling many files. The clean layout separates configuration and execution steps, reinforcing Shneiderman's "Error Prevention" principle by encouraging users to verify parameters before starting analysis.

The inclusion of a Processing Log area ensures that users receive feedback on the batch operation's progress and any issues encountered, promoting transparency and supporting traceability of results in a research context. Overall, this design prioritizes usability and clarity, ensuring the batch detection process remains accessible even to non-technical users.

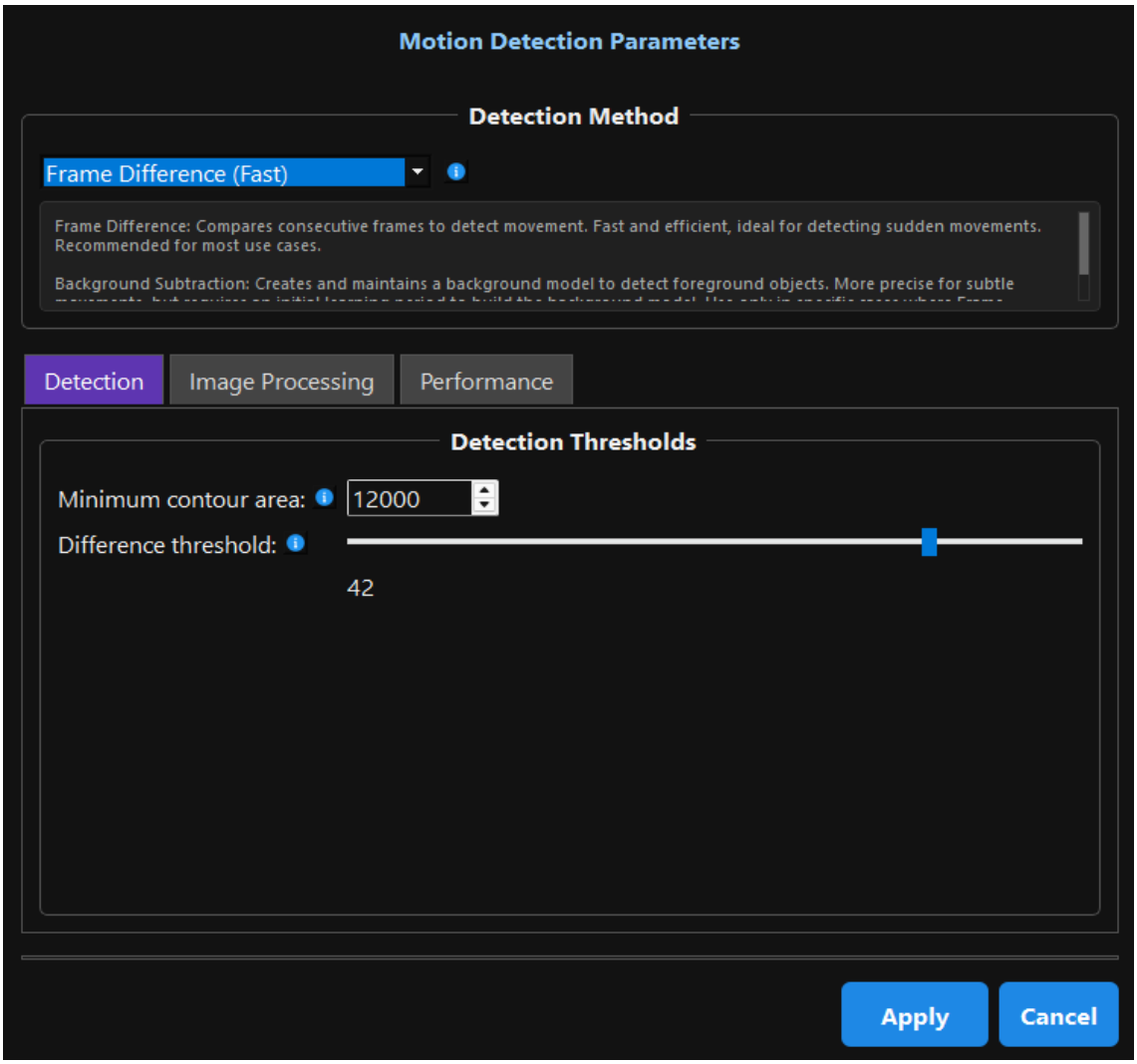


Figure 12 Motion Detection – Parameter Configuration View

(Source: Author)

The configuration window is accessible via the "Configure Parameters" button in the Motion Detection module. The top section allows users to select the detection method, offering a choice between Frame Difference and Background Subtraction. Frame Difference is set as the default option because it best fits the typical use case of detecting clear, sudden movements in home-recorded sleep videos. An explanatory text box below describes the differences between the two

methods in simple terms, supporting informed decision-making even for non-technical users.

Below, parameters are organized into three dedicated tabs: Detection, Image Processing, and Performance. This grouping helps users navigate settings logically, with the Detection tab presented first because it contains the most critical thresholds that directly impact motion detection results such as minimum contour area and difference threshold. The other tabs still offer important controls for users who want to fine-tune the analysis for specific scenarios.

Additionally, the interface includes hoverable information icons next to each parameter, providing contextual explanations and recommended ranges. This design choice ensures that users do not need to memorize parameter meanings or acceptable values, thereby reducing errors and making the configuration process more accessible to clinical researchers.

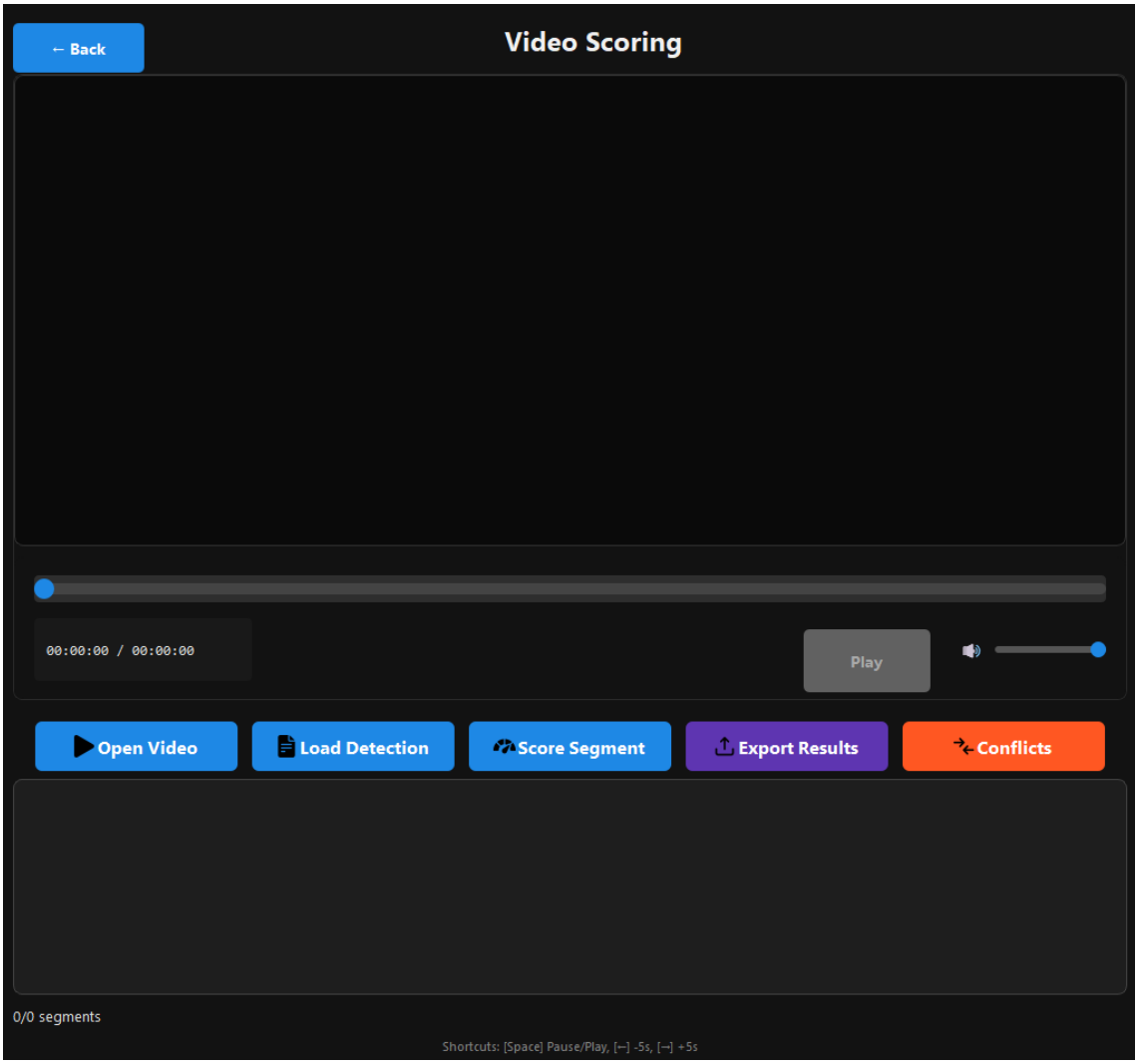


Figure 13 Video Scoring Interface

(Source: Author)

This screen shows the Video Scoring module, designed to support structured, interactive annotation of movement episodes detected in clinical video recordings. At the top, the integrated video player allows users to review footage directly within the application, with controls for playback, pause, time navigation via a slider, precise timestamp display, and volume adjustment. These elements enable scorers to analyze recordings carefully without needing external players or tools.

Below the player, the interface arranges all necessary scoring controls in a clear left-to-right order to support intuitive workflows. The first button lets users select the video file they wish to analyze, while the Load Detection button offers the option to import precomputed detection data. When this button is used, the app checks for a matching detection file in the expected location and prompts the user with a pop-up dialog to confirm whether to load it automatically or skip it.

The Score Segment button (explained in more detail in a dedicated section) opens the annotation form for marking detections as correct, incorrect, or uncertain. Export Results enables users to save their completed scoring in a structured Excel file with clear formatting. The final Conflicts button is reserved for reviewing scorer agreement and highlighting potential discordances across annotations; this functionality is described separately to maintain clarity.

Beneath the main controls is the segments list, which populates automatically when detection data is loaded. Each segment's position is also overlaid onto the video slider as a color-coded marker: yellow by default (awaiting scoring), green for confirmed correct detections, red for incorrect detections, and orange for uncertain cases. This visualization ensures scorers can quickly see the state of analysis at a glance.

At the bottom of the screen, the interface displays a running count of analyzed segments versus the total loaded, along with keyboard shortcuts to improve efficiency (e.g., spacebar to play/pause, arrow keys to skip ±5 seconds).

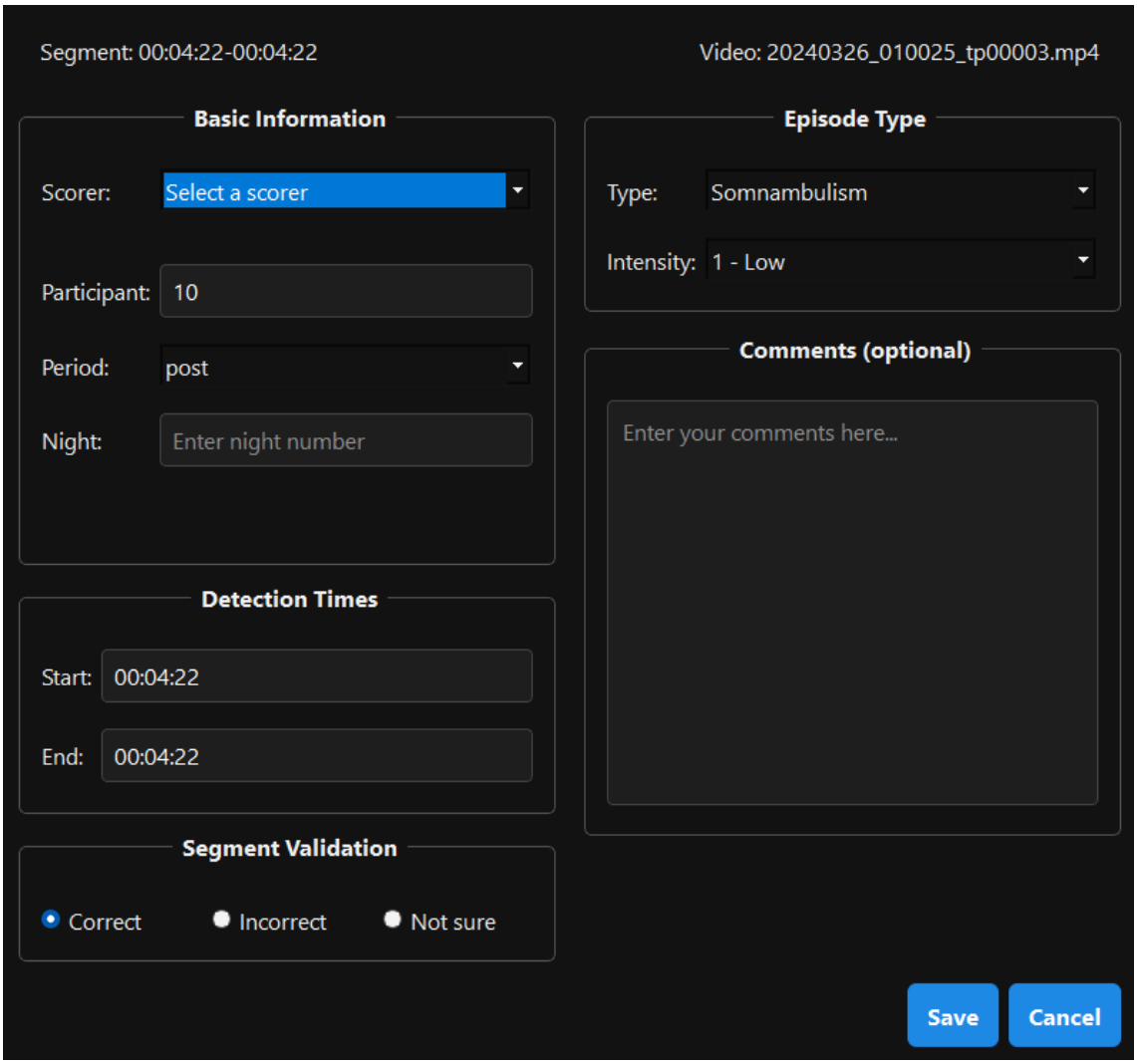


Figure 14 Video Scoring – Segment Annotation Form

(Source: Author)

This is the scoring form that opens when the user clicks Score Segment in the Video Scoring module. At the top, the form displays the segment's start and end times along with the video filename, giving clear context for the episode being annotated.

The Basic Information section allows the user to select the scorer's name from a predefined list, with an option to enter custom initials if needed. Fields for Participant and Period are automatically populated when the video is selected, reducing repetitive data entry and supporting consistency across annotations. The Night field can be entered manually to match the specific recording session.

The Detection Times section pre-fills with the selected segment's start and end times but remains editable so scorers can fine-tune timing to match the precise moment the event is observed during playback. This ensures annotations reflect clinical judgment rather than relying blindly on automated detection.

The Segment Validation options let the user specify whether the detection is Correct, Incorrect, or Not sure. Selecting Incorrect triggers a dynamic form behavior that hides the subsequent fields,

preventing the user from entering episode details that would be irrelevant for false detections. This design choice reduces cognitive load and minimizes error, aligning with Jakob Nielsen's "Error Prevention" heuristic, by ensuring users only see inputs that are relevant to the validation choice.

For correctly detected segments, the Episode Type and Intensity dropdowns allow detailed classification, covering common parasomniac behaviors such as somnambulism, night terrors, and confusional arousals along with the episode's severity. Finally, an optional Comments box provides flexibility for scorers to record any additional qualitative observations.

After the form is saved, a clip of the selected segment is created and saved in a clip folder.

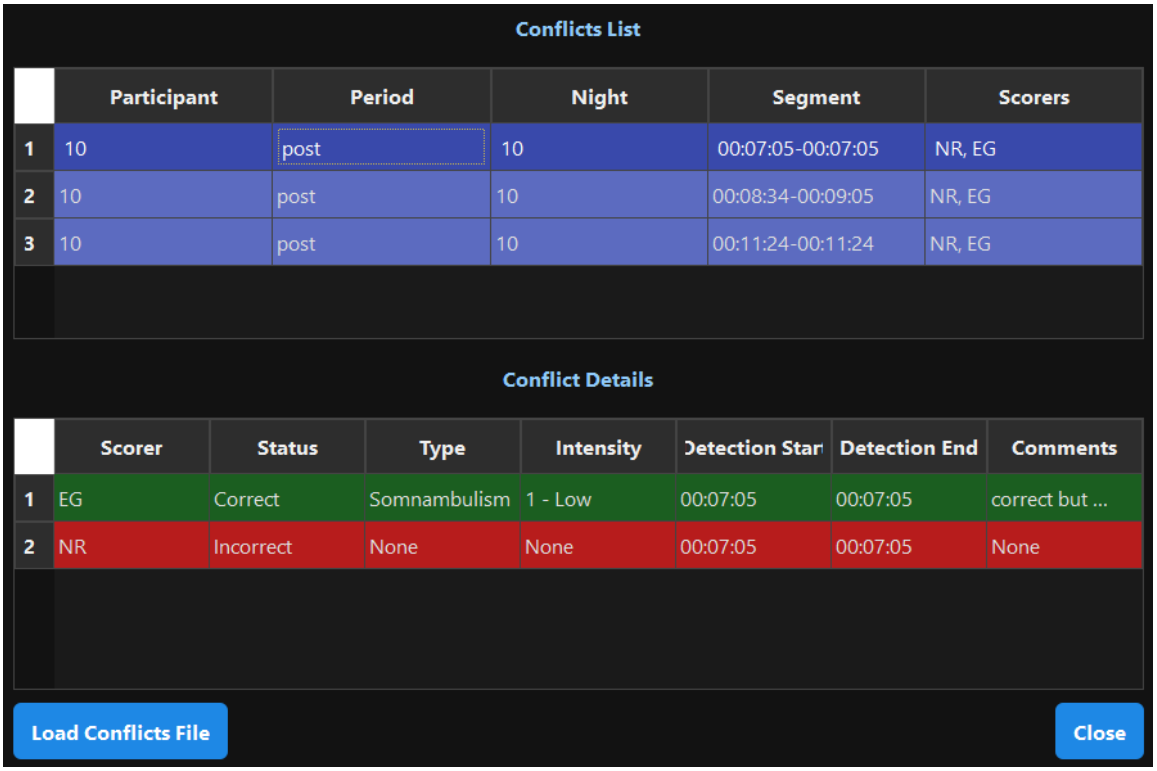


Figure 15 Conflicts View – Comparison and Resolution Interface

(Source: Author)

This screen shows the Conflict List and Conflict Details interface, which opens when the user clicks the Conflicts button in the Video Scoring module. Its purpose is to help scorers and supervisors identify and resolve discordances between multiple annotations of the same video segment, ensuring consistent, high-quality data for research.

In the lower-left corner, users can load the scoring file to automatically detect potential discordances. The system analyzes all scored segments, identifying differences in status (e.g., Correct vs. Incorrect), episode type, intensity, or significant time discrepancies (more than 10 seconds difference in start or end time). Detected agreements are automatically saved in a new worksheet within the same Excel file, making them accessible for clean reporting.

The upper section displays the Conflict List, summarizing all detected discordances with columns for participant, period, night, segment times, and involved scorers. This overview allows users to

quickly scan where discordances exist across the dataset. When a user selects a conflict from this list, the detailed comparison is shown below in the Conflict Details panel.

This lower section presents the two scorers' entries side by side, highlighting key differences in status, type, intensity, times, and any entered comments. Color coding enhances readability: green indicates Correct, red indicates Incorrect, following the same color code as the segment scoring interface.

5.1.3 Excel Output and Result Structure

Beyond its interactive detection and scoring interface, the application is designed to generate a single structured Excel file that consolidates all results from the annotation process. This file serves as the final, shareable output of the workflow, supporting the clinical team's need for traceable, consistent, and human-readable records.

The Excel export is structured into multiple tabs, each dedicated to a specific type of information gathered during analysis. By organizing data this way, the system ensures that scorers, clinicians, and researchers can easily navigate between raw annotation records, validated agreements, and detected discordances, without confusion or manual reformatting.

Column headers are standardized to reduce ambiguity and support direct integration with statistical tools or reporting templates. The overall formatting is designed to be clear and professional, aligning with the needs of a clinical research environment that values both precision and usability.

Participant	Periode	Video Name	Video Duration	Date	Start Hour	End Hour	Detections	Detection Config
10	post	20240326_010025_tp00003.mp4	02:00:51	26-03-2024	01:00:25	03:01:16	00:04:22-00:04:22; 00:07:05-00:07:05; 00:07:57-00:08:01; 00:08:34-00:09:05; 00:11:24-00:11:24; 00:14:36-00:15:40; 00:16:26-00:16:26; 00:17:40-00:17:55; 00:22:00-00:22:15; 00:26:50-00:26:50;	{ "detection_strategy": "frameDiff", "min_contour_area": 12000, "blur_kernel": 7, "frame_skip_seconds": 2, "merge_gap_sec": 30, "margin_seconds": 5, "diff_threshold": 42}
10	post	20240325_233517_tp00002.mp4	01:25:08	25-03-2024	23:35:17	01:00:25	00:31:02-00:31:04; 00:56:34-00:58:07; 00:58:38-00:58:40; 00:59:23-00:59:52; 01:01:36-01:02:09; 01:10:39-01:10:58; 01:19:37-01:19:37	{ "detection_strategy": "frameDiff", "min_contour_area": 12000, "blur_kernel": 7, "frame_skip_seconds": 2, "merge_gap_sec": 30, "margin_seconds": 5, "diff_threshold": 42}
10	post	20240326_030116_tp00004.mp4	01:57:33	26-03-2024	03:01:16	04:58:49	00:26:03-00:27:56; 00:28:27-00:28:27; 00:28:58-00:28:58; 00:29:48-00:30:29; 00:31:54-00:32:27; 01:02:12-01:02:12; 01:05:32-01:05:32; 01:24:58-01:25:06; 01:27:46-01:28:29; 01:33:14-01:33:23;	{ "detection_strategy": "frameDiff", "min_contour_area": 12000, "blur_kernel": 7, "frame_skip_seconds": 2, "merge_gap_sec": 30, "margin_seconds": 5, "diff_threshold": 42}

Figure 16 Excel Output – Detections Tab

(Source: Author)

This tab of the exported Excel file summarizes all detected motion segments for the analyzed videos. Each row corresponds to a single processed video file, with columns capturing essential contextual information such as Participant ID, Period, Video Name, Video Duration, Recording Date, Start Hour, and End Hour. This structure ensures traceability by linking detection results back to specific participants, sessions, and time intervals.

The Detections column lists all identified movement segments for each video in standardized start–end timestamp format (hh:mm:ss), enabling scorers to see exactly when motion was detected without having to replay the entire video manually. To further support reproducibility and methodological transparency, the Detection Config column records all parameters used during

analysis including detection strategy, contour size thresholds, frame skipping intervals, and margins.

By capturing both raw detection outputs and the precise configuration settings in the same worksheet, this design supports clinical research requirements for auditability and reproducibility, while also offering a human-readable, easily shareable format for clinical teams and researchers reviewing the motion analysis results.

Video Name	Participant	Period	Night	Segment	Detection Start	Detection End	Type	Intensity	Status	Comments	Scorer
20240326_010025_tp00003.mp4	10	post	10	00:04:22-00:04:22	00:04:22	00:04:22			Incorrect		EG
20240326_010025_tp00003.mp4	10	post	10	00:04:22-00:04:22	00:04:22	00:04:22			Incorrect		NR
20240326_010025_tp00003.mp4	10	post	10	00:07:05-00:07:05	00:07:05	00:07:05	Somnambulism	1 - Low	Correct	correct but other told correct	EG
20240326_010025_tp00003.mp4	10	post	10	00:07:05-00:07:05	00:07:05	00:07:05			Incorrect		NR
20240326_010025_tp00003.mp4	10	post	10	00:07:57-00:08:01	00:07:57	00:08:01	Somnambulism	1 - Low	Correct	yes 2	EG
20240326_010025_tp00003.mp4	10	post	10	00:07:57-00:08:01	00:07:57	00:08:01	Somnambulism	1 - Low	Correct	yes	NR
20240326_010025_tp00003.mp4	10	post	10	00:08:34-00:09:05	00:08:34	00:09:05	Somnambulism	1 - Low	Correct	time diff higher than 10 sec	EG
20240326_010025_tp00003.mp4	10	post	10	00:08:34-00:09:05	00:08:05	00:09:05	Somnambulism	1 - Low	Correct	lower start	NR
20240326_010025_tp00003.mp4	10	post	10	00:11:24-00:11:24	00:11:24	00:11:24	Somnambulism	1 - Low	Correct	Sleep Token	EG
20240326_010025_tp00003.mp4	10	post	10	00:11:24-00:11:24	00:11:24	00:11:24	Somnambulism	1 - Low	Not sure		NR

Figure 17 Excel Output – Scoring Results Tab

(Source: Author)

This tab of the exported Excel file captures the scoring results produced by expert reviewers using the scoring form. Each row corresponds to a single annotated segment, with columns that systematically record all relevant details needed for clinical analysis and research audit trails.

Key fields include the Video Name, Participant ID, Period, and Night, linking each annotation back to the specific recording session and participant. The Segment column shows the original detected start and end times, while Detection Start and Detection End allow scorers to adjust timing to match their visual review of the event precisely. This design choice supports clinical judgment by enabling corrections to automated detection output.

Episode characteristics are also recorded in structured fields: Type (e.g., Somnambulism, Night Terror), Intensity (with standardized levels), and Status indicating whether the detection was deemed Correct, Incorrect, or Not Sure. An optional Comments column allows scorers to provide qualitative context, such as explaining disagreements or special considerations. Finally, the Scorer field ensures all annotations are clearly attributed, supporting traceability and enabling analysis of inter-rater agreement.

To further improve usability, each tab in the Excel file is color-coded, making it easy for users to immediately recognize which section they are working in and navigate between detections, scoring results, agreements, or discordances with minimal confusion.

Video Name	Participant	Period	Night	Segment	Detection Start	Detection End	Type	Intensity	Status	Comments	Scorers
20240326_010025_tp00003.mp4	10	post	10	00:04:22-00:04:22	00:04:22	00:04:22	None	None	Incorrect		NR, EG
20240326_010025_tp00003.mp4	10	post	10	00:07:57-00:08:01	00:07:57	00:08:01	Somnambulism	1 - Low	Correct	yes 2 + yes	NR, EG

Figure 18 Excel Output – Agreements Tab

(Source: Author)

This tab of the exported Excel file records all scoring segments where complete agreement was reached between multiple scorers. Each row represents a segment where reviewers assigned the same validation status, episode type, intensity, and closely matching detection times. By listing only these agreed annotations, the Agreements tab serves as a validated dataset suitable for direct clinical review or statistical analysis.

Compared to the detailed Scoring Results tab, this view consolidates the information to focus specifically on consensus data. In particular, Comments and Scorers are grouped together to show shared notes and explicitly list which reviewers contributed to the agreement. This format emphasizes transparency while minimizing redundant detail, making it easier for researchers to quickly identify validated events.

5.1.4 Detection Workflow

The motion detection module of the application was designed to automate the identification of potential movement events in home-recorded sleep videos while maintaining a user-friendly and clinically appropriate workflow. The process is structured into three clear stages that balance ease of use for non-technical users with the technical depth and flexibility required in clinical research.

The first step involves input selection and parameter configuration. Users begin by choosing the specific video file to analyze and defining the start and end times of the segment they wish to process. This targeted approach ensures that only the relevant portion of the recording is included, which both respects privacy considerations and improves computational efficiency. A dedicated configuration window allows users to adjust key detection parameters, such as thresholds, minimum contour area, and blur kernel size. These settings are stored in a centralized configuration file, promoting consistency across analyses and supporting reproducibility for research teams.

The second step is the detection processing itself, where the core analysis of the video frames is performed using a structured pipeline. This pipeline transforms raw video footage into structured, time-stamped segments of detected movement.

The process begins with initialization and configuration, where the application loads the selected video file and extracts metadata such as frame rate, duration, and resolution. Users specify the analysis window by providing start and end times, while detection parameters, including algorithm choice and thresholds, are loaded from the configuration file to ensure consistency.

During data preparation, frame numbers corresponding to the defined timestamps are calculated based on the video's frame rate. A masking step removes timestamp overlays in the video frames to prevent false positives. The system also applies frame-skipping logic to sample the video at configurable intervals, balancing detection accuracy with processing speed.

Motion detection itself supports two selectable strategies:

- Frame Difference, the default and recommended method for stable home-recorded videos, captures pairs of consecutive frames, converts them to grayscale, applies Gaussian blur to reduce noise, and computes the absolute difference between frames. A binary threshold isolates regions of significant change, followed by morphological operations to clean the mask. Contours are detected and filtered by a minimum area threshold to eliminate noise.
- Background Subtraction (MOG2) offers an alternative for more dynamic environments. It initializes a statistical background model using a warm-up phase to learn static regions. Each frame is then processed to segment foreground movement using the model, with configurable parameters such as variance threshold, learning rate, and history length. Masks are cleaned and contours are extracted similarly to the Frame Difference approach.

After detection, post-processing filters out small or spurious contours, converts frame numbers

back to standardized hh:mm:ss timestamps, and merges temporally close events into single segments based on a configurable merging gap. Optional time margins can be automatically added before and after each detected segment to ensure complete coverage of the motion event.

The system also handles export and conflict detection. Metadata such as participant ID, period, and recording date are extracted from filenames and included in the output. When existing analyses are detected for a video, the app compares parameter configurations and prompts the user if differences exist, ensuring traceable, consistent processing.

For batch processing, the application reads an Excel input file listing multiple videos and their analysis windows. It validates paths and formats, launches parallel workers to process videos simultaneously, and aggregates all results into a single Excel file with clearly labeled columns. This supports efficient processing of large datasets while maintaining consistent detection logic and output structure.

Key configurable parameters include the frame skip interval (balancing speed and detection granularity), merging gap (to unify close detections), added margins (to avoid cutting events too tightly), and minimum contour area (to exclude noise). Algorithm-specific settings such as difference thresholds or background model learning rates allow adaptation to different recording conditions.

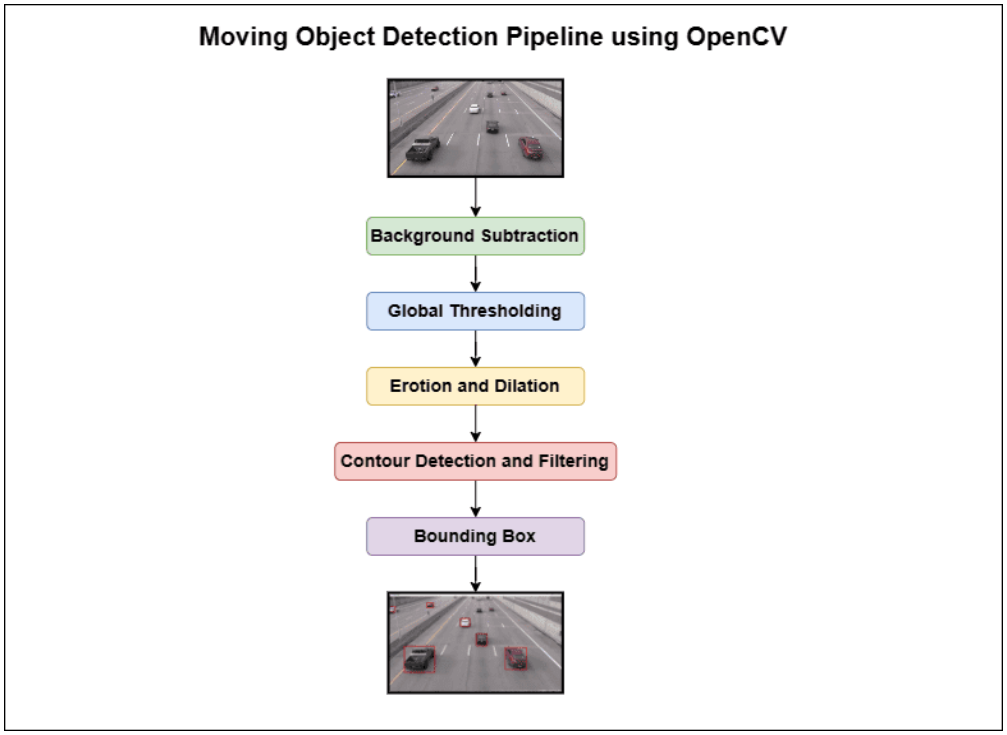


Figure 19 Base Background Substraction Detection Pipeline

Ghosh, A., (2024, 2 décembre). Moving Object Detection using OpenCV. LearnOpenCV – Learn OpenCV, PyTorch, Keras, Tensorflow With Code, & Tutorials. https://learnopencv.com/moving-object-detection-with-opencv/

The third step focuses on result logging and export. Once detection is complete, the application generates structured logs listing all identified movement segments with precise start and end timestamps in hh:mm:ss format. For both single video analysis and batch processing, these results are saved in a standardized Excel file with clearly labeled columns capturing participant IDs, video names, time intervals, and detection parameters. This output format is designed to be human-readable and integrates seamlessly with clinical documentation workflows, allowing scorers and researchers to immediately use the results to guide expert review and scoring processes.

5.2 Architecture and Dependencies

This section describes the technical structure and key dependencies of the developed application. The software was designed to balance professional engineering practices with the need for readability and maintainability in a clinical research environment.

The architecture was intentionally designed to prioritize clarity, maintainability, and ease of use for non-programmer clinical researchers. By separating the interface from the detection logic and maintaining structured configuration and output formats, the tool supports reliable, traceable analyses that can be integrated into existing research workflows.

5.2.1 Overall Structure and Organization

The application follows a modular design with clear separation between the graphical user interface (GUI) and the underlying processing logic. This structure ensures that interface elements can evolve independently of detection algorithms or data export routines, supporting both maintainability and potential future expansion.

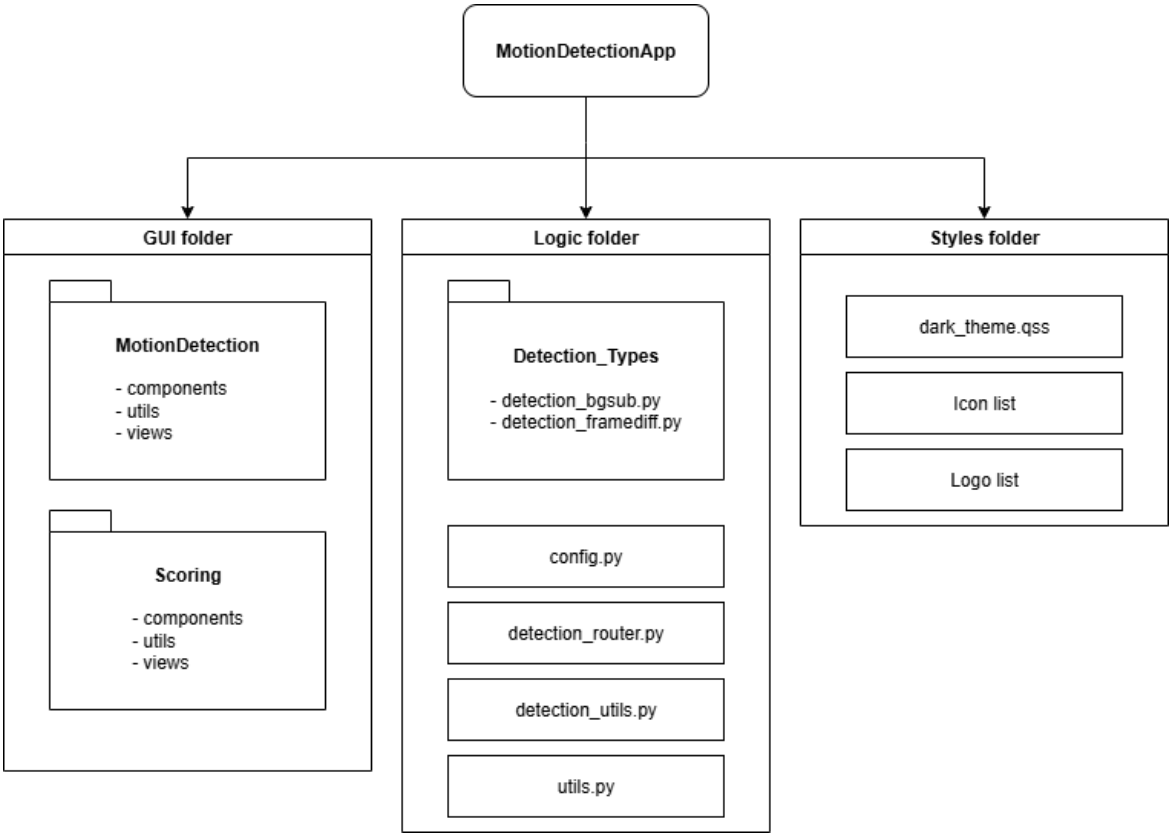


Figure 20 Architecture schema

(Source: Author)

At the top level, the MotionDetectionApp directory serves as the root of the project, containing the primary entry point (main.py), default configuration file (motion_config.json), and a requirements.txt file listing all Python dependencies to ensure reproducible environment setup across installations.

The GUI directory encapsulates all user interface (frontend) components, built using the PyQt5 framework. This layer is subdivided into two main modules reflecting the core functionalities of the software:

Motion Detection Module: Responsible for video analysis tasks. It includes views for single and batch detection workflows, parameter configuration dialogs, reusable UI components (e.g., video list management, time controls), and utility classes supporting threaded batch processing.

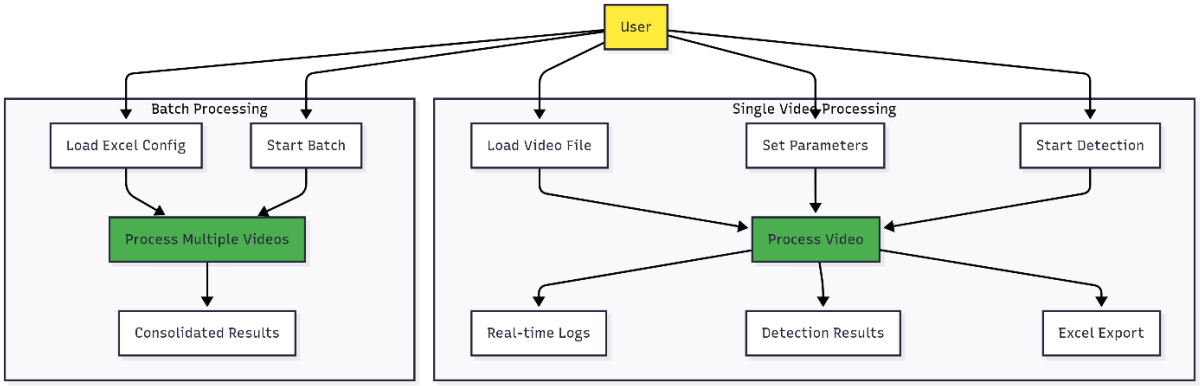


Figure 21 Motion detection module diagram

(Source: Author)

Scoring Module: Dedicated to manual review and annotation of detected segments. It contains interfaces for video playback, structured scoring forms, and conflict resolution views. The scoring module is designed with reusable components like the embedded video player and segment trigger list, alongside utilities for data management and clip creation.

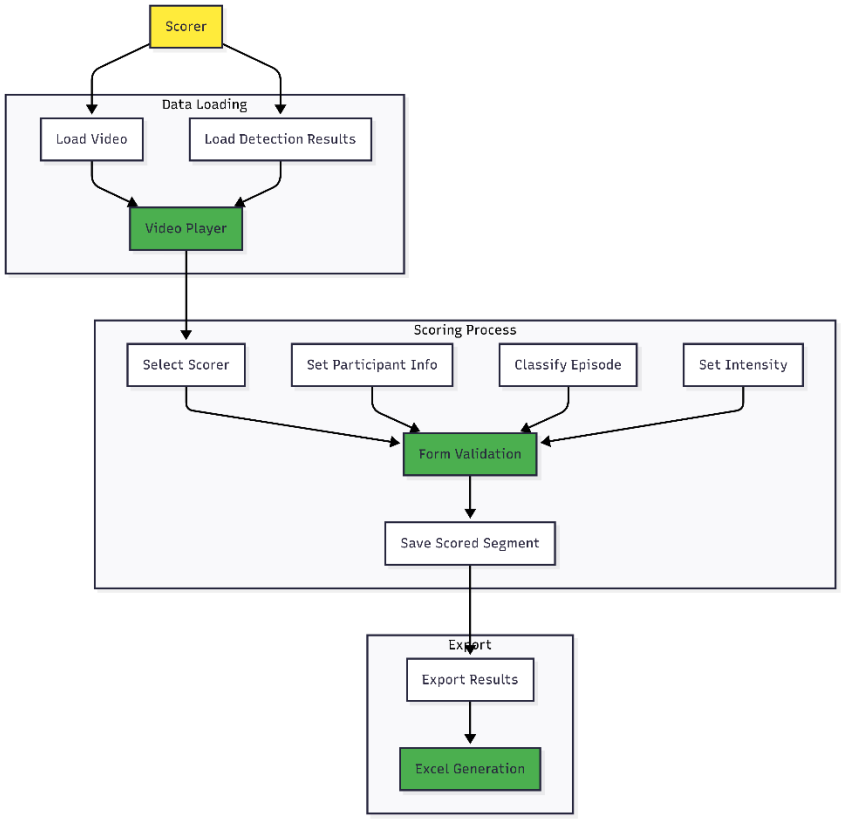


Figure 22 Scoring module diagram

(Source: Author)

This separation between MotionDetection and Scoring ensures that each workflow remains logically self-contained while sharing a consistent design language and user experience.

The Logic directory implements the application's backend. It manages configuration loading, detection routing, and the implementation of detection algorithms. A dedicated Detection_Types submodule contains algorithm-specific implementations such as frame differencing and background subtraction designed for easy extensibility. This modular backend structure makes it straightforward to integrate additional detection methods in the future.

Finally, the styles directory holds all visual resources needed for a polished user experience, including the CSS-based theme file and application branding assets.

5.2.2 Configuration Management and Data Flow

Central to the application's design is its motion_config.json file, which stores all adjustable parameters in a single, human-readable format. This approach allows clinicians and researchers to adjust analysis settings such as frame skipping intervals, contour thresholds, or algorithm-specific parameters without modifying source code. Centralizing configuration also promotes consistent, reproducible analyses across sessions or between team members.

Input data is designed to follow standard formats familiar to clinical teams. Videos are processed in widely compatible .mp4 or .mkv formats. For batch analyses, users provide an Excel input file (.xlsx) listing video paths and analysis windows in a structured table. This supports large-scale processing without requiring repeated manual configuration.

Output data is similarly standardized. All results are saved in a single Excel file with clearly separated, color-coded tabs for Detections, Scoring Results, Agreements, and Discordances. This organization improves usability by allowing scorers and researchers to quickly navigate between different stages of analysis while maintaining a consistent structure suitable for integration with existing documentation workflows. Columns are clearly labeled to support audit trails and data analysis.

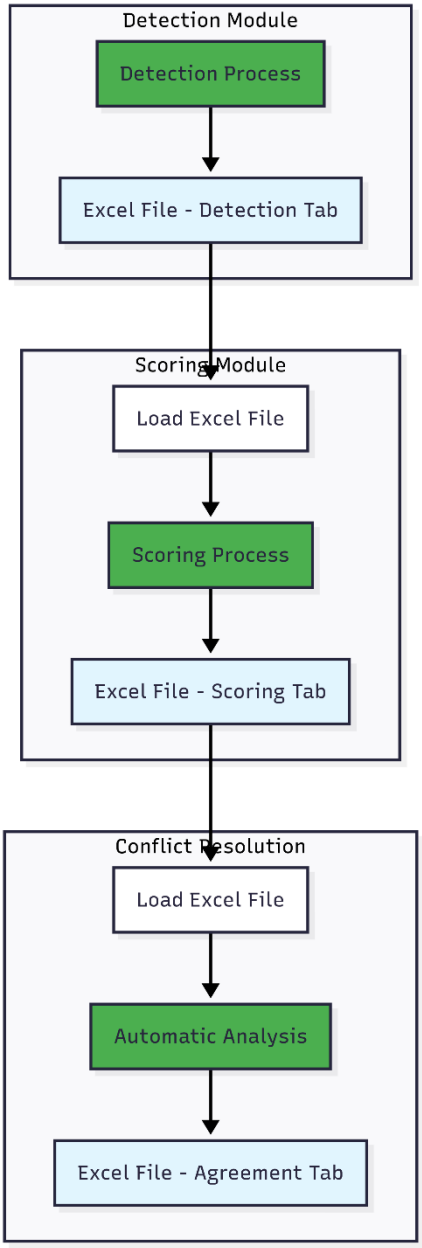


Figure 23 Data flow diagram

(Source: Author)

5.2.3 Dependencies and Environment Management

The software relies on several well-supported Python packages to deliver a professional, user-friendly experience:

- PyQt5: For the desktop GUI, providing advanced widgets and layout management suitable for clinical use.
- OpenCV: For efficient video frame processing, supporting Frame Difference and Background Subtraction detection methods.

- NumPy and Pandas: For fast numerical operations and structured data management, enabling clean integration with Excel exports.
- openpyxl: To read and write Excel files with multiple tabs and formatted outputs.
- python-vlc: For reliable video playback within the scoring module, supporting common video formats with timeline control.
- pywin32: For compatibility with Windows-specific file dialogs and integration features.

All dependencies are specified in the requirements.txt file, supporting reproducible installations across different systems. The application is designed to run inside a Python virtual environment, which isolates dependencies from the Python installation system and ensures consistent behavior during deployment. This approach simplifies installation for new users and supports collaborative development by providing a clearly defined, version-locked environment.

5.3 Testing Results

To ensure the technical stability, portability, and functional completeness of the application before any user validation, a structured program of internal testing was conducted during development. This testing process focused not only on identifying bugs but also on verifying that the entire workflow matched the requirements of the clinical research environment at CHUV.

Testing began on the developer's primary workstation, which served as the main environment for iterative development. Here, all individual components were validated in detail. This included checking the video input selection, motion detection processing in both single and batch modes, parameter configuration windows, scoring form behaviors, and the conflict resolution interface. Logs and error messages were reviewed to confirm they were clear and helpful. User interface consistency was also examined to ensure that forms, buttons, and labels used standardized language and layouts.

To ensure cross-environment compatibility, the software was then installed and tested on a second personal computer with different hardware and system settings. This step verified that installation instructions, especially the creation and activation of a dedicated Python virtual environment, were correctly documented, that dependencies installed reliably from the requirements.txt file, and that no hard-coded paths or environment-specific assumptions existed in the codebase.

A final critical test was performed on a workstation within the CHUV environment itself. This validated that the software could be deployed on hospital-managed systems, confirmed that no conflicts existed with CHUV security or IT policies, and ensured that local data storage practices met privacy requirements for handling anonymized clinical video files.

Throughout this process, functional end-to-end tests were performed to simulate typical clinical workflows. For example, single and batch processing runs were configured and executed, scoring results were exported to Excel, and scorer disagreement files were loaded and reconciled. This

holistic testing approach ensured that the software reliably supported all intended use cases from start to finish.

During these cycles, several issues were identified, particularly in the user interface, where inconsistent layouts or confusing labels could slow down scorers. Error handling was also refined to better guide users when entering incorrect parameters or attempting unsupported actions. Each issue was logged and addressed systematically through iterative development, with repeated retesting to verify fixes.

Ultimately, this thorough internal testing phase ensured that the application was robust, reproducible, and fully functional across different environments. By delivering a validated and documented installation process, consistent user experience, and reliable feature set, the project established a solid foundation for subsequent stakeholder testing and clinical feedback.

5.4 Proof of Concept Validation

After completing internal development and technical testing, the application was deployed for proof of concept validation with the clinical stakeholder at CHUV. This phase aimed to verify that the tool functioned reliably in its intended environment and matched the workflow needs of real-world clinical use.

Deployment began with a collaborative remote session to install the software on a workstation at CHUV. Together, the developer and stakeholder ensured that the virtual environment was correctly set up, all dependencies were installed successfully, and any environment-specific considerations were addressed immediately.

Following installation, the stakeholder performed independent testing over approximately two weeks, processing real clinical video recordings. This testing covered all core features, including single and batch motion detection, parameter configuration, integrated scoring with video playback, conflict management, and structured Excel export.

To illustrate actual usage during this phase, the following table summarizes key metrics collected across all runs:

Metric	Value
Number of single-video detection runs	14
Number of videos in batch detection runs	151
Total detection runs	165
Number of participants (unique IDs)	2
Total segments detected (all modes)	~ 934
Testing duration	~ 2 weeks
Example batch run (1-hour, 26 videos, Frame Difference)	213 segments detected, 3 missed real episodes at diff_threshold=25
Note on threshold tuning	Lower threshold values increase sensitivity (fewer missed real episodes) but also increase false positives

Table 2 Compilation of data from testing period

Table created by the author.

This data demonstrates that the tool was used extensively under realistic conditions, particularly in batch mode to handle large volumes of video efficiently. The example batch run highlights how parameter tuning directly affects detection performance: using a lower diff_threshold of 25 resulted in 213 detected segments over 26 videos while missing only 3 real episodes.

Such results underline an important design trade-off: lower thresholds improve sensitivity and reduce the chance of missing true episodes but inevitably increase the number of false positives. In the clinical research context, this approach is intentional and acceptable, as it prioritizes ensuring that no clinically relevant events are missed even if it means scorers must later filter out extra detections during review.

User feedback collected during this phase also provided valuable insights. The stakeholder described the tool as a very good base for identifying episodes, while noting variability in detection quality depending on recording conditions such as lighting or subtlety of movements. In particular, tests with Participant 12 revealed that even with good lighting, certain episodes with notable head movements were missed, emphasizing the challenge of creating a universally optimal parameter set.

However, the baseline parameters generally performed well for high-quality recordings, and the interface made adjusting parameters quick and intuitive between videos. To avoid missing any true episodes, the user indicated a preference for running detections with very low thresholds to capture all possible micro-movements even at the cost of reviewing more false positives. This strategy reinforces the importance of having a robust second-scorer workflow to validate and refine results.

Overall, the stakeholder found the interface very user-friendly and meeting initial expectations.

Feedback also suggested refinements to make forms clearer and workflows more intuitive, as well as expanding the conflict management feature to better support scorer reconciliation. No critical bugs or technical barriers were encountered, confirming the system's stability and reliability in its intended clinical context.

Based on this collaborative validation, iterative updates were implemented to improve usability and expand functionality in line with stakeholder needs. This approach ensured that the final tool not only met its technical requirements but genuinely supported the practical demands of clinical researchers working with sleep study video data.

5.5 Achievement of Objectives

At the outset of this bachelor project, the planned scope was defined in the official thesis datasheet, which described creating a graphical user interface to support manual scoring of parasomniac episodes. The initial idea was to develop a user-friendly front end that would simply connect to an existing detection tool, allowing scorers to review video segments, validate events, and record standardized annotations.

However, through early meetings and detailed discussions with the clinical stakeholder, it became clear that the real needs of the project were significantly broader than the original description suggested. The existing detection code was minimal, more a prototype or demonstration script than a production-ready system, and lacked batch processing, parameter management, logging, and structured export. As a result, the scope naturally expanded to include not only the scoring interface but also a complete motion detection module.

During this phase, it was acknowledged that if the additional detection features proved too difficult to implement fully, the project could still deliver value by focusing on the scoring part alone. However, through iterative development, testing, and close stakeholder collaboration, both major components were successfully implemented:

- Motion Detection, with single-video and batch modes, parameter configuration, two selectable algorithms, structured Excel outputs, and logs.
- Scoring, with integrated video playback, detection segment overlay, interactive annotation forms, clips creation, conflict checking, and export of standardized scoring results.

It is worth noting that the scope of certain features grew over time, as discussions with the stakeholder clarified real clinical workflows. For example, the Conflicts button was originally planned to simply flag differences between scorers. Based on feedback, it was expanded to load scorer files, identify and list discordances in a structured view, and save confirmed agreements back to the Excel export.

That said, the conflict resolution logic remains partially manual. While the system can highlight where scorers disagree and store agreed corrections, there is not yet an automated method for merging and tracking all corrections seamlessly. Currently, the tool serves as an assisted reconciliation interface that clearly shows where scorers align or differ and supports them in discussing and recording final agreements.

In summary, all core objectives were achieved:

- The software enables clinicians to detect, score, and export parasomniac episodes from video recordings in a structured, reproducible manner.
- Both detection and scoring modules are fully operational, with interfaces tailored to clinical needs.
- Outputs are formatted for easy review and research analysis.

The project ultimately delivered more than originally planned, responding to the stakeholder's real-world needs while also defining clear directions for future enhancements, such as more automated conflict resolution, user management, and potential database integration for multi-user research teams.

Feature / Aspect	Initial Scripts	Final Integrated Application
User Interface	Command-line only (detection) and minimal Tkinter form (scoring)	PyQt5 GUI with dedicated views for detection and scoring
Detection Modes	Single video only	Single-video and batch processing with Excel-based batch input
Detection Algorithms	Frame Difference only	Frame Difference and Background Subtraction (MOG2) with configurable parameters
Parameter Configuration	Hard-coded or minimal CLI input	Full GUI-based configuration window with tabs and tooltips
Output Format	Raw CSV with unstructured data	Structured Excel export with color-coded tabs for Detections, Scoring Results, Agreements, and Discordances
Metadata Integration	None	Participant ID, period, night number, scorer initials automatically included
Video Integration	No video playback in scoring	Integrated video player with timeline, playback controls, and detection segment overlays
Scoring Workflow	Manual form entry only	Interactive scoring form with dynamic fields, validation, and segment-specific annotation
Conflict Management	Simple manual discordance check	Interface comparing scorer annotations and saving agreements
Logs and Feedback	Minimal real-time feedback	Integrated processing logs and user feedback in the GUI

Table 3 Comparison between provided scripts and final integration

Table created by the author.

6. Discussion

6.1 Critical Analysis of the Development Process

Overall, the development process for this project was relatively straightforward and well-organized, but it also required careful adaptation to evolving needs and collaborative expectations.

One of the first challenges was adapting to a scope that changed over time. Initially, the project was defined in the bachelor thesis datasheet as focusing primarily on developing a scoring interface to support existing detection capabilities. However, early meetings with the stakeholder revealed that the existing detection code was extremely limited, prompting a clear choice: either deliver only the scoring component, or expand the project to include a fully functional detection module. This decision had to be made early, and clear communication with the stakeholder ensured that expectations were aligned and realistic about what could be achieved within the timeframe.

Because development started early in the project timeline, I was able to manage this change in scope effectively. Having an initial prototype of the scoring component from the first stakeholder meeting helped me quickly understand what the libraries could offer and how the interface might be structured. This early experimentation was essential in selecting the appropriate technologies for the user interface, motion detection logic, and Excel export features. It also gave me a clearer vision of how to organize the codebase and ensure maintainability.

Collaboration with the stakeholder was a strong point throughout the project. Communication was frequent and clear, allowing us to clarify expectations at every stage. Stakeholders understood and supported the technical decisions I had to make, including explaining the feasibility of new requests or suggesting alternative solutions when needed. Even as the stakeholder's requirements evolved, especially when the complexity of certain features increased, we maintained an open dialogue to assess what could realistically be delivered. This transparency and shared understanding were critical to managing scope creep effectively.

Of course, development was not without challenges. Although the overall design and planning were clear, unexpected bugs emerged, particularly related to user interface behaviors, that consumed more time than anticipated. Managing these issues required careful debugging and sometimes revisiting earlier decisions. This meant that in the final stages of testing, I had to communicate honestly with the stakeholder about minor delays and negotiate the possibility of delivering final refinements after the main project deadline.

Ultimately, the development process highlighted the importance of early planning, clear communication, and adaptability when working on collaborative, user-centered software in a clinical research context.

6.2 Interpretation of Results and Their Clinical Relevance

The results of this project can be understood most meaningfully by considering how the developed tool addresses the practical needs of clinical research and improves existing workflows.

At its core, the application delivers a substantial improvement over the manual, time-consuming methods previously used to identify and annotate parasomniac episodes in home-recorded sleep videos. By integrating automated motion detection with an intuitive scoring interface, it helps clinicians and researchers focus their time and expertise on validating relevant events, rather than watching entire nights of footage frame by frame. This shift has clear practical implications for reducing workload and improving efficiency in sleep research studies.

The tool was explicitly designed to fit the real workflows in use at CHUV, not as a generic video annotation platform. Throughout development, features were selected and refined to match the clinic's actual needs: configurable time windows for privacy, support for their video formats, batch processing of large study datasets, and structured Excel exports formatted for their documentation and analysis protocols. The integration of conflict detection and agreement management between scorers directly supports inter-rater consistency, a critical requirement in clinical research to ensure objective, standardized data collection.

From a clinical perspective, the tool adds value by enabling more objective, reproducible, and traceable scoring of parasomniac events. By providing clear logs of detection parameters and structured annotation exports, it improves auditability and transparency, both of which are important for clinical trials and retrospective studies. The detection parameters and scoring forms are configurable enough to support different study designs or patient populations, demonstrating a balance between usability and flexibility.

Another key aspect of its clinical relevance lies in its stakeholder-driven design. The software was not developed in isolation, but in close collaboration with the end users who will rely on it in their research practice. This ensured that the final tool not only works technically but is also meaningful and usable in real-world clinical contexts, aligning with the workflows, expectations, and constraints of the researchers who will apply it.

7. Conclusion

The primary objective of this bachelor project was to develop a user-friendly, clinically appropriate software tool to support the analysis and scoring of parasomniac episodes in home-recorded sleep videos. Motivated by the need to streamline and standardize what was previously a fully manual and time-consuming process, the project aimed to provide clinicians and researchers with an integrated platform that automates motion detection, facilitates consistent scoring, and generates structured outputs suitable for clinical documentation and research analysis.

7.1 Delivered Results

To meet this objective, the project delivered a complete, integrated software solution that combines both automated motion detection and structured scoring workflows. The application includes a dedicated motion detection module capable of processing single videos or batch datasets, with configurable parameters and two selectable detection algorithms tailored for clinical sleep recordings. Results are exported in standardized Excel formats with clearly labeled, color-coded tabs to ensure traceability and compatibility with clinical documentation practices.

The scoring module provides an intuitive interface for reviewing detected segments, visualizing video data with integrated playback controls, and completing structured annotation forms. It also incorporates a conflict detection and resolution feature that helps scorers identify disagreements and document agreements consistently.

Throughout development, the tool was refined in close collaboration with the clinical stakeholder, ensuring that the final design matched real-world workflows and expectations at CHUV. Despite an initially narrower scope defined in the thesis datasheet, the project successfully adapted to incorporate the full detection pipeline, producing a robust, professional-quality application ready for use in clinical research settings.

7.2 Perspectives

While the current version of the application delivers a usable solution for CHUV's clinical research needs, there remain several clear directions for future development and enhancement.

One immediate area for improvement is the conflict detection and resolution feature. Although it currently supports comparing scorer files, identifying discordances, and saving confirmed agreements, this functionality remains partially manual. Future work could focus on fully automating conflict resolution, integrating logic that proposes unified annotations or automatically flags high-priority discrepancies. Additionally, this feature could be moved from within the scoring module to its own dedicated section in the main menu, with a redesigned interface specifically for managing scorer comparisons and agreements. Such changes would improve usability and clarify the workflow for users working collaboratively.

From a development perspective, the codebase could also benefit from further refactoring to achieve a clearer separation of concerns. While the current structure, with parts of the front-end interface tightly coupled to interface-specific logic and detection processes, was chosen to accelerate development and deliver a working solution within project timelines, the idea of refactoring was already considered from the outset. Structuring the code to more distinctly separate the user interface, application logic, and detection algorithms would enhance maintainability, simplify future updates, and better support the integration of new detection methods or scoring features.

Beyond conflict management, the motion detection module itself could be adapted for broader uses. While it was designed with sleep-related parasomnias in mind, the underlying detection logic is general enough to be reused or extended for other video analysis applications. With further development, parts of the codebase could be integrated into larger research tools, repurposed for different types of behavioral or clinical monitoring, or optimized further to handle larger datasets or more complex detection needs.

8. Bibliography

Apple. (2025, may 25). *Human interface guidelines*. Apple Developer. https://developer.apple.com/design/human-interface-guidelines

Attarian, H. &. (2013, Sep). Treatment options for disorders of arousal: A case series. *International Journal of Neuroscience*, pp. 623–625. https://doi.org/10.3109/00207454.2013.783579

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Budiu, R. (2025, may 25). *Glossary of UX research methods*. Nielsen Norman Group. https://www.nngroup.com/articles/research-methods-glossary/

Chamine, I. A. (2018, Feb 15). Hypnosis intervention effects on sleep outcomes: A systematic review. *Journal of Clinical Sleep Medicine*, pp. 271–283. https://doi.org/10.5664/jcsm.6952

Colten, H. R., & Altevogt, B. M. (2006). *Sleep Disorders and Sleep Deprivation: An Unmet Public Health Problem.* National Academies Press.

Confédération suisse. (2011, sept 30). *Loi fédérale du 30 septembre 2011 relative à la recherche sur l'être humain (LRH)*. Consulté le may 26, 2025, sur Fedlex. https://www.fedlex.admin.ch/eli/cc/2013/617/fr

Confédération suisse. (2020). *Loi fédérale du 25 septembre 2020 sur la protection des données (LPD)*. Consulté le 07 14, 2025, sur Fedlex. https://www.fedlex.admin.ch/eli/cc/2022/491/fr

Dauvilliers, Y. (2022). Parasomnias: Epidemiology, clinical features, and diagnostic approach. *Journal of Sleep Research, 31*(4). https://doi.org/10.1111/jsr.13732

Google. (2025, may 25). *Material Design 3*. Material.io. https://m3.material.io/

Hauri, P. J. (2007, jun 15). The treatment of parasomnias with hypnosis: A 5-year follow-up study. *Journal of Clinical Sleep Medicine*, pp. 369–373. https://pmc.ncbi.nlm.nih.gov/articles/PMC1978312/

itberrios. (s.d.). *Detection with frame differencing*. Consulté le Jully 08, 2025, sur GitHub. https://github.com/itberrios/CV_projects/blob/main/motion_detection/detection_with_frame_differencing.ipynb

Kryger, M. H., Roth, T., & Dement, W. C. (2022). *Principles and Practice of Sleep Medicine.* Elsevier.

Mainieri, G. L. (2023, mars 27). Diagnosis and management of NREM sleep parasomnias in children and adults. *Diagnostics*, p. 1261. https://doi.org/10.3390/diagnostics13071261

Microsoft. (2025, may 25). *Design principles*. Fluent 2. https://fluent2.microsoft.design/design-principles

Nielsen, J. (1995, January 1). *10 usability heuristics for user interface design.* Nielsen Norman Group. https://www.nngroup.com/articles/ten-usability-heuristics/

Open Source Vision Foundation. (2025, may 26). *OpenCV – Open Source Computer Vision Library*. Consulté le may 26, 2025, sur OpenCV. https://opencv.org/

OpenCV. (2025). *Getting Started with Videos*. Consulté le July 13, 2025, sur Open Source Computer Vision. https://docs.opencv.org/4.x/dd/d43/tutorial_py_video_display.html

OpenCV. (2025). *How to Use Background Subtraction Methods*. Consulté le July 13, 2025, sur Open Source Computer Vision. https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html

Patel, A. K., Reddy, V., Shumway, K. R., & Araujo., J. F. (2024). *Physiology, Sleep Stages.* https://doi.org/NCBI Bookshelf. https://www.ncbi.nlm.nih.gov/books/NBK526132/

Rafalski, K. (2025, may 25). *Top Python GUI libraries: A comprehensive overview*. Netguru. https://www.netguru.com/blog/python-gui-libraries

Rosebrock, A. (2025, may 26). *Basic motion detection and tracking with Python and OpenCV*. PyImageSearch. https://pyimagesearch.com/2015/05/25/basic-motion-detection-and-tracking-with-python-and-opencv/

Sateia, M. J. (2014). International classification of sleep disorders—third edition: highlights and modifications. *Chest, 146*(5), 1387–1394. https://doi.org/10.1378/chest.14-0970

Scrum.org. (2025). *What is Scrum?* Consulté le July 13, 2025, sur Scrum.org. https://www.scrum.org/resources/what-scrum-module

Shneiderman, B. &. (2005). *Designing the user interface: Strategies for effective human-computer interaction.* Pearson Education, Inc.

Sleep Foundation. (2025, may 21). *Stages of sleep*. SleepFoundation.org. https://www.sleepfoundation.org/stages-of-sleep

Soegaard, M. (2025, may 25). *System Usability Scale for Data-Driven UX*. Interaction Design Foundation. https://www.interaction-design.org/literature/article/system-usability-scale

Stallman, H. M. (2016, Nov 10). Prevalence of sleepwalking: A systematic review and meta-analysis. *PLoS ONE*, p. 20. https://doi.org/10.1371/journal.pone.0164769

Standardization, I. O. (2018). *ISO 9241-210:2010 Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems.* ISO.org. https://www.iso.org/standard/63500.html

Sultani, W., Chen, C., & Shah, M. (2018). Real-world anomaly detection in surveillance videos. *arXiv preprint*. https://doi.org/10.48550/arXiv.1801.04264

9. Declaration of authorship

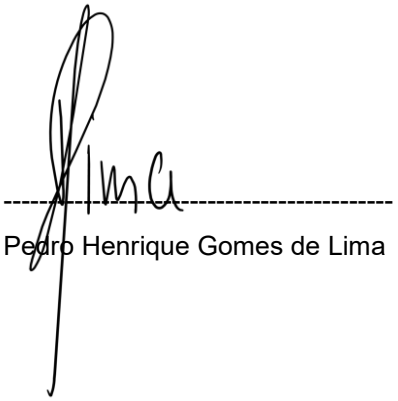
I hereby confirm that I have completed the present Bachelor thesis independently and using only the specified resources. I have exclusively used the sources and materials cited in this work, with the exception of the individuals who provided me with essential information for the writing of this report and who are explicitly acknowledged below:

- Dr. Jean-Paul Calbimonte
- Nina Rimorini

I further declare that, in addition to the specified academic and personal resources, I have used generative Artificial Intelligence (AI) tools (including OpenAI's ChatGPT) as supportive aids during the thesis work. These tools were employed to:

- Develop ideas and explore implementation strategies for the software code.
- Improve the structure, clarity, and professional quality of the written text.
- Review and standardize documentation in alignment with academic conventions.

All final decisions regarding content, wording, code design, and analysis presented in this thesis remain my own responsibility. The use of AI tools was limited to assistance and did not replace my own critical thinking or original contributions.



Pedro Henrique Gomes de Lima

14.07.2025

Date

Annex I: Product Backlog

Numbe	Theme	As a...	I want to...	So that I can...	Acceptance Criteria	Priorit	Status	MoSCoV
1.1	Motion Detection Single Video	User	Design detection UI	Set start/stop times and parameters	Screen sections clear; fields validated	High	Done	Must
1.2	Motion Detection Single Video	User	View processing logs	Track progress	Real-time logs visible	High	Done	Must
1.3	Motion Detection Single Video	User	Run frame differencing detection	Identify motion events	Configurable parameters work	High	Done	Must
1.4	Motion Detection Single Video	User	Mask timestamp overlays	Avoid detection errors	Overlays blacked out correctly	High	Done	Must
1.5	Motion Detection Single Video	User	Select background subtraction	Adapt to dynamic scenes	MOG2 option works	High	Done	Must
1.6	Motion Detection Single Video	User	Export detection results	Review in Excel	Excel includes participant, period, night	High	Done	Must
2.1	Batch Processing	User	Design batch UI	Process multiple videos	Upload instructions clear	High	Done	Must
2.2	Batch Processing	User	Show example input	Prepare files correctly	Sample format displayed	High	Done	Must
2.3	Batch Processing	User	View batch progress logs	Track batch processing	Logs show progress/errors	High	Done	Must
3.1	Video Scoring	Scorer	Use video player with timeline	Navigate footage	Slider and controls present	High	Done	Must
3.2	Video Scoring	Scorer	See segment markers	Identify detections	Color-coded markers visible	High	Done	Must
3.3	Video Scoring	Scorer	Browse detection list	Quickly find segments	List links to playback	High	Done	Must
3.4	Video Scoring	Scorer	Enter episode details	Annotate precisely	Type, intensity, trigger captured	High	Done	Must
3.5	Video Scoring	Scorer	Export annotations to Excel	Share results	Structured output with all details	High	Done	Must
3.6	Video Scoring	Scorer	Color-code output tabs	Improve clarity	Excel tabs consistently colored	Medium	Done	Must
4.1	Conflict Resolution	Scorer	View discordance list	See all conflicts	Selection interface	High	Done	Must
4.2	Conflict Resolution	Scorer	Compare scorer details	Analyze differences	Side-by-side comparison shown	Medium	Done	Must
4.3	Conflict Resolution	Scorer	Highlight differences	Resolve efficiently	Color-coded differences	High	Done	Must
5.1	Data Export	User	Format Detections tab	Review motion events	Includes participant and timing	High	Done	Must
5.2	Data Export	User	Log config parameters	Audit detection	All settings recorded	High	Done	Must
5.3	Data Export	User	Add Scoring Results tab	Store annotations	Complete scorer details logged	High	Done	Must
5.4	Data Export	User	Color-code tabs	Improve readability	Tabs distinguish data types	Low	Done	Must
5.5	Data Export	User	Create Agreements tab	Store consensus data	Only agreed entries included	High	Done	Must
6.1	UI/UX Design	User	Use clear landing page	Choose workflow	Navigation to detection and scoring	High	Done	Must
6.2	UI/UX Design	User	Ensure consistent UI styling	Avoid confusion	Uniform colors, fonts, buttons	High	Done	Must
6.3	UI/UX Design	User	View processing logs	Monitor operations	Clear logs in Detection modules	Medium	Done	Must
7.1	Configuration & Maintenance	Developer	Store parameters in JSON	Edit easily	Readable config file used	High	Done	Must
7.2	Configuration & Maintenance	Developer	Separate UI and logic	Maintain code	Modular design enforced	High	Done	Should
7.3	Configuration & Maintenance	Developer	Organize directories	Find files easily	Logical folder structure	High	Done	Must
7.4	Configuration & Maintenance	Developer	Use requirements.txt	Install dependencies	All packages listed	High	Done	Must
7.5	Configuration & Maintenance	Developer	Support virtualenv	Isolate environment	Install guide uses venv	High	Done	Should
7.6	Configuration & Maintenance	Developer	Document setup steps	Install easily	Instructions clear	High	Done	Must
7.7	Configuration & Maintenance	Developer	Have an easier way to start the app	Quickly launch the app	Script or executable that launches the app	Low	To Do	Could

Annex II: Meeting Tracker

Date	Time	Meeting Title	Participants	Agenda Topics	Decisions Made	Notes
13.05.2025	15:00	First Meeting	CAJ,GOP	Project introduction	Schedule meeting with Nina	-
15.05.2025	10:00	Hands on	RIN,GOP	Overview of code structure and logic	Grant code access and define analysis scope	OneNote
20.05.2025	11:00	Weekly Meeting	CAJ,GOP	Backlog review and early documentation planning	Rework backlog and begin state-of-the-art documentation	-
21.05.2025	Full day	CHUV Visit	RIN,GOP	On-site visit to EEG and hypnosis facilities	Discuss general app functionality, intended usage, and design logic	OneNote
26.05.2025	10:00	Weekly Meeting	CAJ,GOP	Initiate state-of-the-art research	Continue using confidential references, prepare to start development for CHUV server testing	-
02.06.2025	10:00	Weekly Meeting	CAJ,GOP	First coding steps	Proceed with coding, ensure modular design for detection features	-
04.06.2025	18:00	Choices of development and preferences	RIN,GOP	Design and test snapshot and detection types	Finalize detection code; create snapshots only during scoring	OneNote
09.06.2025	10:00	Weekly Meeting	CAJ,GOP	Code development progress	Continue coding work	-
12.06.2025	15:00	Installation	RIN,GOP	First deployment test on CHUV PC	Apply minor code changes, bug fixes; gather follow-up feedback	-
16.06.2025	10:00	Weekly Meeting	CAJ,GOP	Ongoing code development	Assess feasibility of adding more features; finalize and test latest additions	-
23.06.2025	10:00	Weekly Meeting	CAJ,GOP	Documentation planning	Delay final feature, focus on documentation work	-
24.06.2025	15:00	Testing v2	RIN,GOP	Evaluate latest features; discuss remaining development limits	Review feedback on v2; if approved, shift focus entirely to documentation	-
01.07.2025	10:00	Weekly Meeting	CAJ,GOP	Documentation development	Continue writing, add references and create architecture diagrams	-
07.07.2025	10:00	Weekly Meeting	CAJ,GOP	Documentation updates	Include images for human detection, refine project diagrams, plan detailed review for further feedback	-
14.07.2025	10:00	Weekly Meeting	CAJ,GOP	Final documentation review	Apply corrections and complete final edits	-

Motion Detection GUI

A PyQt5-based application designed for clinical video motion detection and scoring. This GUI simplifies the analysis workflow for clinicians and researchers, providing objective tools to annotate, score, and compare movement patterns in video recordings.

Purpose

This tool was developed to:

- Objectively assess motion in sleep-related clinical recordings
- Provide an intuitive and accessible interface for scorers
- Standardize scoring criteria across evaluators
- Support batch processing for high-throughput analysis
- Detect and resolve inter-scorer conflicts effectively

Key Features

- **Modern Interface**: Clean, dark-themed PyQt5 interface with intuitive navigation
- **Flexible Video Processing**: Supports both individual and batch video analysis
- **Multiple Detection Algorithms**:
  - **Frame Difference**: Fast processing, suitable for clear motion detection
  - **Background Subtraction (MOG2)**: More precise detection with background modeling
- **Comprehensive Analysis**: Real-time processing logs and graphical result summaries
- **Export Options**: Excel output with detailed motion event timestamps
- **Integrated Scoring System**: Manual annotation and validation with conflict detection
- **Batch Processing**: Process multiple videos automatically using Excel configuration
- **User-Friendly**: Extensive tooltip system and parameter guidance
- **Configurable**: Adjustable detection parameters for different use cases
- **Preview Mode**: Real-time visualization of detection results during processing

Installation

1. Clone the Repository

```
git clone https://github.com/PHGDL/MotionDetectionGUI.git
cd MotionDetectionGUI
```

2. Set Up Virtual Environment

```
python -m venv venv
# Windows
venv\Scripts\activate
# macOS/Linux
source venv/bin/activate
```

3. Install Dependencies

```
pip install -r requirements.txt
```

4. Launch Application

```
cd MotionDetectionApp
python main.py
```

5. Reset Installation (if needed)

```
deactivate
rm -rf venv             # macOS/Linux
rmdir /s /q venv        # Windows
python -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Usage Guide

Landing Page

The main interface provides clear navigation between modules:

- **Motion Detection**: Analyze videos for movement events
- **Scoring**: Manually review and annotate detected segments

Motion Detection Module

Single Video Analysis

1. **Load Video**: Select your video file using the file browser
2. **Set Parameters**: Configure detection algorithm and sensitivity
3. **Define Analysis Range**: Set start and end times for processing
4. **Run Detection**: Execute analysis with real-time log feedback
5. **Review Results**: View detected motion events and export to Excel

Batch Processing Mode

1. **Prepare Excel File**: Create a configuration file with video list
2. **Load Configuration**: Import the Excel file with video parameters
3. **Configure Detection**: Set global parameters for all videos
4. **Start Batch Process**: Automatic processing of all videos
5. **Export Results**: Consolidated results exported automatically

Batch Excel Format Requirements

Create an `.xlsx` file with this structure:

video	start_time	end_time
video1.mp4	00:00:10	00:01:00
video2.mp4	00:02:30	00:03:45

Requirements:

- **Time Format**: `HH:MM:SS` or float seconds
- **Video Location**: Videos must be in a `full_video` folder at the same level as the Excel file
- **Column Names**: Exactly match the column headers above
- **File Extension**: Must be `.xlsx` format

Example Directory Structure:

```
project_folder/
├── batch_config.xlsx
└── full_video/
    ├── video1.mp4
    └── video2.mp4
```

Scoring Module

Manual Annotation Workflow

1. **Load Video**: Import video file for annotation
2. **Select Motion Results**: Choose folder containing motion detection results
3. **Setup Session**: Select scorer, participant, and analysis period
4. **Annotate Episodes**: Add detailed annotations with:
    - **Time Range**: Precise start and end times
    - **Event Type**: Classification of motion event
    - **Intensity**: Subjective intensity rating
    - **Trigger**: Causative factors or triggers
5. **Conflict Detection**: Automatic identification of scoring discrepancies

Configuration

Parameters can be edited via the GUI or directly in `motion_config.json`:

```
{
    "common": {
        "frame_skip_seconds": 2,
        "timestamp_region": [0, 0, 1000, 120],
        "merge_gap_sec": 30,
        "margin_seconds": 5,
        "detection_strategy": "framediff"
    },
    "framediff": {
        "blur_kernel": 7,
        "min_contour_area": 12000,
        "dilate_iterations": 2,
        "morph_kernel_size": 5,
        "morph_opening": true,
        "morph_closing": true,
        "diff_threshold": 42
    },
    "bgsub": {
        "blur_kernel": 7,
        "min_contour_area": 6000,
        "varThreshold": 65,
        "history": 100,
        "dilate_iterations": 2,
        "morph_kernel_size": 5,
        "morph_opening": true,
        "morph_closing": true,
        "learning_rate": 0.005,
        "warmup_frames": 30
    }
}
```

Project Structure

```
MotionDetectionApp/
├── main.py                      # Entry point for launching the app
├── motion_config.json           # Default configuration parameters
├── requirements.txt             # List of dependencies
├── GUI/                         # Frontend interface components
│   ├── __init__.py
│   ├── landing_page.py          # Main navigation page
│   ├── MotionDetection/         # Motion detection module
│   │   ├── views/               # User interface screens
│   │   │   ├── main_detection_view.py
│   │   │   └── motion_parameters_dialog.py
│   │   ├── components/          # Reusable UI components
│   │   │   ├── video_list.py
│   │   │   └── time_controls.py
│   │   └── utils/               # GUI utilities
│   │       ├── batch_processor.py
│   │       └── motion_worker.py
│   └── Scoring/                 # Scoring and validation module
│       ├── views/               # User interface screens
│       │   ├── main_scoring_view.py
│       │   ├── scoring_form_view.py
│       │   └── conflicts_comparison_view.py
│       ├── components/          # Reusable UI components
│       │   ├── video_player.py
│       │   ├── trigger_list.py
│       │   └── trigger_slider.py
│       └── utils/               # GUI utilities
│           ├── scoring_utils.py
│           ├── scoring_data_utils.py
│           └── clip_creator.py
├── Logic/                       # Core detection algorithms and utilities
│   ├── __init__.py
│   ├── config.py                # Configuration management
│   ├── detection_router.py      # Algorithm routing system
│   ├── detection_utils.py       # Common detection utilities
│   ├── utils.py                 # General utilities
│   └── Detection_Types/         # Detection algorithm implementations
│       ├── detection_framediff.py    # Frame difference algorithm
│       └── detection_bgsub.py        # Background subtraction algorithm
└── styles/                      # Visual resources and themes
    ├── dark_theme.qss           # Application stylesheet
    ├── logo.png                 # Application logo
    ├── logo_CHUV.png            # CHUV logo
    └── [various]_icon.png       # UI icons
```

Developer Guide: Adding New Detection Types

Step-by-Step Implementation

Step 1: Create the Detection Algorithm

Create a new file in `Logic/Detection_Types/` following the naming convention:

```
# Logic/Detection_Types/detection_newtype.py
import cv2
from pathlib import Path
from ..detection_utils import apply_morphology, postprocess_and_export

def detect_motion_newtype(video_path, start_time, stop_time, output_dir, config,
display=False, log_func=None):
    """Motion detection using your new algorithm."""

    # Extract algorithm-specific parameters
    your_param1 = config["your_param1"]
    your_param2 = config["your_param2"]

    # Common parameters
    frame_skip_seconds = config["frame_skip_seconds"]
    min_contour_area = config["min_contour_area"]

    # Initialize video capture
    cap = cv2.VideoCapture(video_path)

    # Your detection algorithm implementation here
    # ... (implement your detection logic)

    # Use postprocess_and_export for consistent output
    return postprocess_and_export(motion_events, video_path, output_dir, config,
log_func)
```

Step 2: Add Configuration Parameters

Update `motion_config.json`:

```
{
    "common": {
        "detection_strategy": "framediff"
        // ... existing common parameters
    },
    "newtype": {
        "your_param1": 50,
        "your_param2": 0.8,
        "blur_kernel": 5,
```

```
        "min_contour_area": 8000
    }
}
```

Step 3: Register in Detection Router

Update `Logic/detection_router.py`:

```
# Add import
from .Detection_Types.detection_newtype import detect_motion_newtype

# Add case in routing logic
elif method == "newtype":
    return detect_motion_newtype(video_path, start_time, stop_time, output_dir,
full_config, display, log_func)
```

Step 4: Update Configuration Management

Update `Logic/config.py`:

```
if strategy in ["framediff", "bgsub", "newtype"]:  # Add your strategy
    return {**config.get("common", {}), **config.get(strategy, {})}
```

Step 5: Add GUI Support

Update `GUI/MotionDetection/views/motion_parameters_dialog.py`:

```
# Add to strategy combo box
self.strategy_box.addItem("Your New Algorithm", "newtype")

# Add parameter descriptions
self.tooltips = {
    "your_param1": "Description of your first parameter",
    "your_param2": "Description of your second parameter"
}

# Add parameter controls
if self.current_strategy == "newtype":
    self.add_parameter_control("your_param1", "Your Parameter 1:", "spin")
    self.add_parameter_control("your_param2", "Your Parameter 2:", "double_spin")
```

Available Utility Functions

- `apply_morphology()`: Apply morphological operations
- `postprocess_and_export()`: Process and export results
- `time_to_seconds()`: Convert time formats
- `seconds_to_time()`: Convert seconds to HH:MM:SS

Technical Notes

Processing Information

- **Log System**: Comprehensive log window tracks all activity during execution
- **Output Structure**: Results folder `motion_results/` is created at participant level
- **File Naming**: Output files include timestamps and detection parameters for traceability

Performance Considerations

- **Frame Skipping**: Configurable frame skip reduces processing time
- **Multi-threading**: Background processing doesn't block the user interface

User Experience

- **Tooltips**: Extensive tooltip system provides guidance for all parameters
- **Progress Tracking**: Real-time progress updates during processing
- **Error Handling**: Graceful error handling with informative messages

Technical Details

- **Video Formats**: Supports common video formats (MP4, AVI, MOV)
- **Resolution**: Handles various video resolutions automatically
- **Timestamp Processing**: Configurable timestamp region extraction
- **Export Format**: Excel files with detailed motion event information